A Database Supported Modeling Environment for Pandemic Planning and Course of Action Analysis

Yifei Ma

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Jiangzhuo Chen, Co-chair
Madhav V. Marathe, Co-chair
Keith R. Bisset
Stephen G. Eubank
Edward A. Fox
Anil Kumar S. Vullikanti

May 15, 2013

Blacksburg, Virginia

Keywords: epidemic simulation, database system, distributed system

A Database Supported Modeling Environment for Pandemic Planning and Course of Action Analysis

Yifei Ma

ABSTRACT

Pandemics such as the 2009 H1N1 and the 2003 SARS events can significantly impact public health and society. In addition to analyzing historic epidemic data, computational simulation of epidemic propagation processes and disease control strategies can help us understand in the laboratory the spatio-temporal dynamics of epidemics. Consequently, the public can be better prepared and the government can control future epidemic outbreaks more effectively. Epidemic propagation simulation systems which use high performance computing technology have been proposed and developed to understand disease propagation processes. The proposed systems, however, do not strongly support two important steps in modeling disease propagation: run-time infection situation assessment and intervention adjustment. In addition, while these simulation systems are computationally efficient in their simulations, most of them have limited capabilities in terms of modeling interventions in realistic scenarios.

In this dissertation, we focus on building a modeling and simulation environment for epidemic propagation and propagation control strategy. The objective of this work is to design such a modeling environment supporting the previously missing functions while still performing well in terms of the expected features such as modeling fidelity, computational efficiency, modeling capability, etc. Our proposed methodologies to build such a modeling environment are: 1) a loosely coupled and co-evolving model for disease propagation, situation assessment, and propagation control strategy, and 2) using relational databases to assess situations and simulate control strategies. Our motivations are: 1) a loosely coupled and co-evolving model allows us to design modules for each function separately and reduces design complexity for the modeling system, and 2) simulating propagation control strategies using relational databases improves the modeling capability and the human productivity of using this environment. To evaluate our proposed methodologies, we have designed and built a loosely coupled and database supported epidemic modeling and simulation environment. With detailed experimental results and realistic case studies, we demonstrate that our modeling environment provides the missing functions and greatly enhances many expected features, such as modeling capability, without significantly sacrificing computational efficiency and scalability.

Contents

List of Figures

List of Tables

Chapter 1

Opening

1.1 Introduction

Pandemics significantly impact public health and society. They also reaffirm an emergent need to develop timely and effective methods for controlling ongoing epidemic outbreaks and planning for those in the future. Although public health authorities around the world did a tremendous job of coordinating and responding to the H1N1 outbreak, the total number of H1N1 cases reported worldwide was still high. Between April 2009 and April 2010, there were about 60 million reported H1N1 cases in the United States. The 2009 H1N1 pandemic led to thousands of deaths in the world, according to the estimates provided by the Centers for Disease Control and Prevention (CDC) [19]. The 2009 H1N1 pandemic also had a great impact on socioeconomics. During its outbreak, schools were closed in many states, and international travelers from infected countries were isolated for diagnosis. Fortunately, H1N1 was not very virulent, allowing a globally-coordinated response by world and national public health authorities to reduce overall casualties. Controlling future pandemics and reducing their economic and social burden will be challenging due to a number of societal trends, which include increased and denser urbanization, increased local and global travel, and a generally older and immuno-compromised population.

Analysis of the spread of historic epidemics [58] and study of measures for pandemic preparedness and mitigation [64, 28] can help us understand past epidemics to some extent, but this approach has limitations for forecasting future epidemics in an ever-changing world. Computational models play an important role in representing the spatio-temporal dynamics of epidemics. The H1N1 pandemic reaffirmed the need for analytical tools and methods to detect, assess, and respond to future pandemics. By building disease propagation models and their simulation systems, public health authorities extend their knowledge about the risk of future infectious diseases to human health, the impact on our daily lives, the spatio-temporal patterns of disease propagation [58], and the effectiveness of response strategies in

controlling disease propagation.

Computational models can assist us in evaluating diverse response strategies aimed at preventive and mitigating pandemics [37, 65, 11]. The most common response strategies include individual adaptive behaviors and public health policies. During an epidemic outbreak, individuals usually take self-protection precautionary measures, and public health policy makers typically enforce or adjust their epidemic preventing and controlling policies. As examples: wearing face masks and seeking vaccination are widely-adopted self-protection behaviors when public health authorities warn the public of an ongoing epidemic outbreak, and many Asian countries executed school closure response strategies during the 2003 SARS (Severe Acute Respiratory Syndrome). These behaviors and policies can improve human immunity or reduce the chance of spreading disease through social contact. Their roles in disease propagation control in previous epidemic outbreaks now are being studied in public health institutes [27, 33, 57]. In this dissertation, we use the term *intervention* to represent individual adaptive behaviors and public health policies.

Why does intervention need to be modeled with the disease propagation model? First, as we elaborated above, individual adaptive behaviors and public health policies can significantly control disease propagation, impact our daily lives, and effect the local economy [9, 56]. Second, understanding the effectiveness of controlling disease propagation and the side effects of interventions on public health and economy, allows public health authorities to make educated decisions facing future epidemic outbreaks [46]. Third, some interventions are not testable in the real world. For example, closing schools can interrupt the daily lives of school-age children and their families; it is not reasonable to enforce in the real world only for a scientific experiment. Computer simulation is a better methodology for the study of these interventions [37]. A high-fidelity computer simulation of disease propagation with intervention enriches our knowledge about the effectiveness and cost of intervention strategies within different scenarios. In a laboratory we can compare the disease controlling effectiveness of a school closure strategy at the early stage of an epidemic versus at the outbreak stage [33], and we can see the disease controlling effectiveness differences between government policies and individual adaptive behaviors in the same outbreak [33, 44]. Along with examining the intervention effectiveness and cost, we are also interested in knowing the causes of effectiveness differences. From a high-fidelity computer simulation, we can obtain the propagation dynamics under the studied interventions. The dynamical disease propagation process could help us interpret the co-evolution of disease dynamics, public health policy, and individual behavior. Modeling and simulating epidemic propagation processes with interventions better prepares us to control future epidemic outbreaks efficiently and wisely.

Until recently, computational modeling of epidemics has focused on aggregate models [38, 5, 36, 59]. Aggregate computational epidemiology models usually assume that the studied population can be partitioned into a few subpopulations based on their demographic features or their health states. These models also assume the people in the subpopulations are homogeneous and, that people follow a regular interaction structure within and between subpopulations, and transmission rates among subpopulations are the same. Although useful

for obtaining analytical expressions for a number of interesting parameters such as the total numbers of infections, the assumptions made by aggregate models are not very realistic, and these models ignore human diversity in transmitting diseases and the complexity of human social interaction that serves as a major transmission mechanism for infectious disease.

Over the past several years, high-resolution, individual-based and disaggregate computational models have been developed to support planning and response to epidemics [24, 17, 20, 54, 7, 14]. These new models use an endogenous representation of individuals with detailed demographic features and their daily activities, and disease propagation is modeled through the explicit interactions between these individuals.

Designing and building a reliable, scalable, high performance modeling environment using the individual-based model to represent the disease propagation is a complex and challenging problem. Barrett, et al [10] proposed an approach to design and construct such a modeling environment to support epidemic planning in public health authorities and epidemic propagation studies for epidemiologists. The steps to construct this environment include using modeling to: 1) create a set of synthetic individuals, 2) generate (time varying) contact networks, 3) simulate the epidemic process, and 4) represent and evaluate interventions.

Several models of disease propagation over synthetic populations and social contact networks have been proposed and their disease propagation simulation engines based on high-performance computing technologies are currently in use to support public health policy studies [23, 14, 7, 20]. However, most of these models are only able to express a small set of interventions and some of the expressible interventions have been simplified to fit their models of the social contact network. Thus, a model which represents and evaluates interventions to support policy making is still not completely developed.

1.2 Research Questions

Our ultimate objective is to build a modeling environment for pandemic planning and course of action analysis. The expected environment is not just an epidemic propagation simulation system; it should also act as a policy making support system. What are the expected features of such a modeling environment? Based on suggestions from public health experts and our experience with modeling epidemic propagation and building its simulation system, the expected features include but are not limited to:

- **Fidelity**. The models of epidemic propagation, individual behavior, and public policy are realistic and similar to events happening in the real world.
- **Capability**. The modeling environment is capable of supporting a large scope of critical pandemic planning and action analysis studies.

- **Efficiency**. The simulation systems in this modeling environment run fast. Computational efficiency is critical here. The earlier study conclusions can be reported to public health authorities, the better chance authorities have to control an ongoing epidemic.
- **Human productivity**. Computational efficiency is not the only measure. Human productivity also determines the study period. The convenience of designing and performing simulation experiments in this environment is another key to success.
- **Scalability**. Pandemic is a global health issue. Its prevention and control require coordination and collaboration between different regions and countries. Thus, the modeling environment is expected to be able to simulate epidemic propagation in regions from small, local communities to the global community.
- **Modularity**. Epidemic propagation is a co-evolving process, and this environment has to model several co-evolving subsystems. Better modularity makes the design and maintenance of the modeling environment easier.
- **Extensibility**. The modeling environment may serve as the core simulation engine and provide general simulation services. Many special purpose applications can be extended from this core simulation engine.

Due to the modeling environment expected features listed, there exist two research questions: 1) How to design and build a modeling environment that performs well in the terms of the expected features? 2) If the expected features contradict each other (for example, better modularity may cause computational efficiency loss), how to balance the expected features?

1.3 Research Challenges

The first challenge in building this modeling environment concerns intervention modeling. Although we have made significant progress in social contact network and disease propagation modeling, the application of realistic interventions with the existing disease propagation models is still challenging. The adaptive behaviors and the public health policies applied in historic epidemic outbreaks as well as those currently under study are diverse and numerous. For example, there are diverse adaptive behaviors such as wearing face masks and reducing social activities [64]. Policy makers may apply stategies on different targeted subpopulations under different situations [64, 33, 44]. For example, when the vaccine stockpile is insufficient during the early stage of an emergent epidemic outbreak, we could explore the optimal use of the limited vaccines by simulating different distribution strategies. Due to the diversity of interventions, it is difficult to represent all of the possibilities using a few pre-defined functions.

Second, the intervention strategies and their relevant information vary widely in different intervention studies. The simulation of these diverse interventions may require a large scope

of supplemental information other than the epidemic propagation relevant information: demographics are needed to simulate a school closure policy, while geographic information is needed to simulate a region-based quarantine policy. These reasons present a capability challenge in modeling and simulating interventions. During the design period of previous simulation systems, the designers predicted the intervention strategies that could be studied in the future and their relevant information. These simulation systems are only capable of supporting intervention studies within the scope of their predicted interventions. Those beyond scope are either not supported or need considerable redesign and rebuilding efforts. Public health experts anticipate a convenient and user-friendly simulation environment capable of modeling and simulating diverse interventions efficiently. Third and finally, from a system design perspective a scalable epidemic modeling environment must take into account system performance, capability in simulating realistic problems, modularity, and extensibility, expected features which may contradict each other. Take the epidemic simulation systems [7, 14] based on the interaction-based model as examples. These systems use more realistic models to represent the propagation process and can efficiently simulate epidemic outbreaks in large populations by employing high-performance computing technologies, they have limited capabilities in terms of modeling interventions in realistic scenarios. Designing and building an epidemic propagation and intervention simulation system that performs and outperforms in all the expected features is an excellent challenge.

1.4 Hypotheses

As elaborated above, there still exist many challenges in building a modeling environment that can efficiently support real-time epidemic planning and policy making. In order to explore solutions to the proposed research challenges in this dissertation we make the following hypotheses:

- **The hypothesis of system decoupling**

  Disease propagation processes with interventions can be modeled as two loosely coupled subsystems, where these subsystems interact with each other to share their system states. As a result, disease propagation process and intervention process can be modeled separately, and their processes can be simulated locally by sharing their system states.

  Our motivation for attempting to decouple the propagation model and the intervention model is the realization that these two co-evolving systems are distinct from a computing perspective. Disease propagation simulation using the interaction-based model is a network dynamics process. The simulation of a network dynamics process is computationally intensive if the network is large. Its model is relatively generic and does not change frequently in different studies (only the parameters of the model change to represent different disease propagations). In contrast, the intervention simulation is

data-intensive and intervention strategies are diverse and change frequently. The disease propagation simulation requires computational efficiency, while the intervention simulation expects efficiency and flexibility in data searching. Utilizing a single computing technology in these two simulation systems is unlikely to satisfy their distinct expectations. Decoupling and modularizing these two systems and applying different technologies could be the right direction. Moreover, a loosely coupled simulation system is easier to design and more easily reused.

- **The hypothesis of utilizing a relational database system for the intervention simulation**

  In a loosely coupled system most of the recently studied interventions can be modeled by relational calculus and expressed using relational query languages. As a result, interventions can be simulated using a relational database system.

  If the disease propagation model and the intervention model can be loosely coupled, then designing their simulation systems separately and applying different computing technologies in these two systems becomes feasible. From our previous experience with building epidemic simulation systems, we realize high-performance computing technologies are more suitable to disease propagation simulation using individual-based models, but lack flexibility in simulating diverse interventions. We are searching for other computing technologies that are able to express diverse interventions flexibly and simulate them efficiently. We are exploring the capability and flexibility of expressing epidemic interventions in relational query languages and the computational efficiency of simulating interventions in relational database systems.

- **The hypothesis of high efficiency in simulating complex interventions**

  The interactive and database supported epidemic simulation system has acceptable levels of computational performance.

  We are exploring a loosely coupled architecture for this modeling environment and relational database utilization for intervention simulation. A new simulation system based on the loosely coupled architecture may cause performance loss. We need to examine its computational efficiency.

- **The hypothesis of high capability of modeling and simulating realistic epidemic problems**

  Using relational databases for intervention simulation improves the efficiency of performing intervention studies and the capability of simulating realistic interventions.

  The motivation for exploring loosely coupled architectures and utilizing relational databases is a desire to resolve capability limitations in modeling and simulating realistic interventions. We need to demonstrate that a loosely coupled and database supported simulation system is able to improve these capabilities.

1.5 Research Contributions

This dissertation on the topic of a loosely coupled and database supported simulation environment for epidemic propagation and its intervention has made the following research contributions:

- **We propose an abstract and loosely coupled model for epidemic propagation simulation and its intervention simulation. The loosely coupled model enables us to utilize different computing technologies in simulating these subsystems.**

  We propose a set of abstractions that allow us to decouple the primary components of an epidemic simulation: 1) the data-intensive and computationally intensive intervention component and infection situation assessment component; and 2) the relatively generic but computationally intensive disease propagation component. The first two components often demand flexibility while the last requires substantial computational efficiency. The abstractions also formalize the communication and data transfer that takes place when these components interact. It requires a delicate balance between the amount of data that is exchanged and the exchange frequency. Our abstractions are based on a formal model called Co-evolving Graphical Discrete Dynamical System (CGDDS). CGDDS is used to formally model the process of interaction-based epidemic propagation and corresponding intervention. In addition, we overlay a Partially Observable Markov Decision Process (POMDP) over CGDDS to model interventions.

- **We study the feasibility of modeling and expressing epidemic interventions using a relational query language.**

  Utilizing relational databases significantly improves human productivity while performing intervention studies and the capability of intervention modeling, critical limitations in previous systems. Database supported simulation also allows run-time interactions between the simulation system and its users, so they may assess the infection situation and adjust their interventions. An intervention selection support system could be extended from this environment.

  The diversity of epidemic interventions presents a challenge to utilize the existing epidemic simulation systems to support real-time epidemic intervention making. Most of the current epidemic simulation systems enumerate a set of interventions and implicitly implement them. They do not have a generic and expressible interface for their users to customize intervention. As a result, new interventions have to be modeled and simulated in an ad hoc manner. This development work typically is the bottleneck in performing real-time intervention studies. We propose an intervention model based on relations and relational calculus in order to simulate intervention using a relational database. By employing relational calculus and a relational database, the development

effort for new interventions has been largely reduced. Thus, the efficiency of performing intervention studies using this database supported modeling environment could fit the requirements of real-time intervention studies. The relational calculus is able to model a very large set of interventions. From a computation optimization perspective, the simulation of complex intervention usually is both computationally intensive and data-intensive. Full-fledged relational database systems with state of the art of query optimization techniques provide high performance. Finally, our work on modeling and simulating epidemic intervention using relational databases extends the ongoing research of behavior simulation using relational databases [62, 63]. In contrast to other simulation systems, which simulate behaviors totally in a database, our work demonstrates high performance computing technology and data management technology can be combined together for complex simulation problems.

- **We built a simulation environment for epidemiologists and public health policy makers to easily and efficiently perform their studies.**

  The ultimate goal of our effort on an epidemic modeling environment is to offer epidemiologists and public health policy makers a platform to effortlessly and efficiently perform their epidemic propagation and intervention studies. Based on our proposed system design, we have successfully built a simulation framework named Indemics. Several complex interventions that are very difficult to be simulated before have been modeled and simulated quickly using Indemics. The simulation of such complex interventions in the previous systems either requires considerable development effort or is too complex to develop.

- **We studied and analyzed the performance of this simulation environment in detail. We have also optimized its efficiency in simulating epidemic propagation in large populations with complex interventions.**

  High performance in simulation experiments is one of the keys to success for the intervention selection support system. In this dissertation, we examine the performance of Indemics in complex intervention studies. We compare the experimental time of intervention studies using the previous non-database-supported simulation system and using Indemics to show the performance overhead. In order to analyze the source of performance costs, we abstract the primary computation tasks in Indemics and examine the performances of these tasks in different intervention simulations. We introduce a set of expected database features, which are distinct from traditional database applications. We examine the performance of two relational databases in Indemics with different query optimization technologies and hardware architectures. We present our analysis and discussions of their performance differences.

1.6 Dissertation Organization

The rest of this dissertation is organized as follows:

In Chapter 2, we introduce related work on modeling and simulation of epidemic propagation and intervention.

In Chapter 3, we introduce our approach to achieving the research objectives. We present our proposed solutions: a loosely coupled simulation system design and simulating interventions using relational databases. We start with the motivations of these two ideas, and then discuss these two ideas in detail. Finally in this chapter, we present a system formalization of this loosely coupled and database supported simulation system.

In Chapter 4, we present our implementation of the proposed system design, Indemics. An introduction to the system architecture and the primary modules is given first in this chapter. After the architecture introduction, we present our database design in Indemics for intervention simulation. Finally in this chapter, we introduce the API design of Indemics.

In Chapter 5 and Chapter 6, we evaluate our approach based on the proposed research hypotheses. We study the performance of Indemics regarding the following aspects. In Chapter 5, we discuss the intervention implementation efficiency of Indemics, and present our performance prediction methodology for case study planning. In Chapter 6, we focus on the system performance of Indemics in real intervention case studies. We present the performance overhead caused by the system decoupling and using databases, the performance cost analysis, and finally a performance comparison between different database systems in intervention simulations.

In Chapter 7, we introduce two real case studies of intervention strategies, whose simulation experiments are performed using Indemics. We give a detailed introduction to intervention strategies and explain our modeling and implementation of these interventions. The simulation results of each of the case studies are provided.

In Chapter 8, we conclude our work on this loosely coupled and database supported epidemic simulation environment and present further work that could be extended from this dissertation.

Chapter 2

Literature Review

This dissertation focuses on building an environment for person-to-person epidemic propagation modeling and simulation. Our proposed ideas of building this environment involve a loosely coupled system architectures and utilizing relational databases for intervention simulation. Thus, in this chapter we first introduce the early work on modeling and simulating epidemic propagation and its intervention. We then present recent research progress in database supported scientific simulation.

2.1 Epidemic Propagation Modeling and Simulation

First, we introduce the early work on epidemic modeling and simulation. To simulating person-to-person epidemic propagation processes two key progressions—the within-host-disease-progression and the between-host-disease-progression—must be modeled accurately.

There are several prevalent models of within-host-disease-progression. The SIR (Susceptible-Exposed-Recovered) model and SEIR (Susceptible-Exposed-Infectious-Recovered) model are widely used in computational epidemiology [5, 40]. In this dissertation, we use the SEIR model as an example of the within-host-disease-progression model. But research results are also applicable to other within-host-disease-progression models. In the SEIR model, each person is a modeling unit associated with a series of health states, susceptible (S state), exposed (E state), infectious (I state), and recovered (R state). The health state of a person evolves chronologically. As illustrated in Figure 2.1, its evolution starts from the initial susceptible state (we assume all the persons are healthy except a set of infectious persons who are the propagation seeds at the simulation start). The person transitions to an exposed state when infected by another person who is in the infectious state. Such a transition happens when the two individuals are physically proximal. The meaning of proximity depends on the disease under consideration. It turns from the exposed state to the infectious state after an incubation period. During the incubation period, the exposed

person does not transmit the disease to others and has no detectable symptoms. Once in the infectious state, this person potentially transmits his disease to contacts and expresses symptoms. Finally, after the infectious period, he enters the recovered state. This person no longer transmits the disease and becomes immune. The SIR model only has three health states in contrast to the SEIR model (the exposed state is not included in the SIR model). An assumption made in the SIR and SEIR models is that the recovered individuals never re-enter the susceptible state. Therefore, these models are used to model infections such as measles, mumps, and rubella, in which a single exposure typically confers life-long immunity.

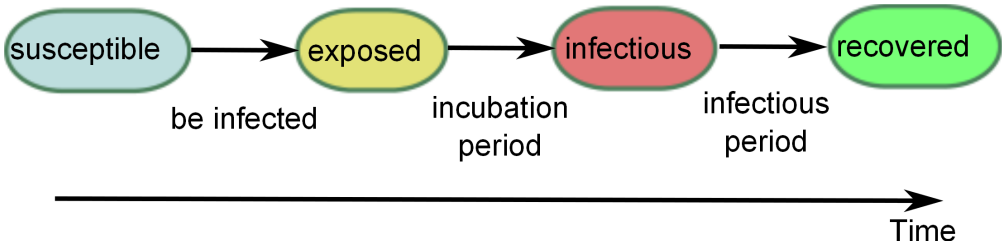

Figure 2.1: The within-host-disease-progression in the SEIR model.

Another process to be modeled in epidemic propagation is disease progression between hosts. There are several different models in the early work for between-host-disease-progression.

Differential Equation Models

The between-host-disease-progression model has been studied for decades. Early between-host-disease-progression models are based on mathematical differential equations [38, 5, 59, 36]. In these mathematic models, individuals in the population are assigned to different subgroups according to their health states and their demographic characteristics (such as age). The disease progression between these subgroups is formulated using differential equations. Our knowledge about the propagation pattern has been enriched from these early models [49, 41]. However, these models do not capture the role of demographic characteristics in propagation processes and the complexity of social contacts that serve as a major propagation mechanism for person-to-person infectious disease such as those that are airborne. Additionally, the number of different subgroups considered in these models is usually small and their parameters, such as mixing rate and reproductive number, are either unknown or hard to observe. By defining appropriate compartments, all this can be modeled. The problem is that the computational complexity keeps increasing.

Individual-based and Mix Group Models

The individual-based system is another methodology for between-host-disease-progression modeling. Demographic characteristics and social behaviors during the propagation pro-

cess are modeled as properties of the individuals. Limited by the computing capability of early computers, we were only able to simulate epidemic propagation using the individual-based model in small populations. With the development of modern computing technologies, epidemic propagation simulation using the individual-based model in large populations becomes possible [20, 45, 28]. Some of these systems assign individuals to different social mixing groups and between-host-disease-progression is modeled in social mixing groups. They assume that the social mixing groups are homogeneously mixed and individuals in the same group have an equal probability of contacting anyone else, thus, equal probability to transmit disease. Despite the fact that a homogeneous mixing assumption simplifies the simulation computation of between-host-disease-progression and their simulation results often agree with historic epidemiological patterns [61], the underlining social contact networks actually are not symmetric. For example, children in the same class have more social contacts and higher infection risk between them than children in the same school but different classes. Social network scientists and public health researchers are studying the role of the social contact network in epidemic propagation [39, 48]. Cauchemez, et al [18] reported several interesting observations about how the social network of an elementary school affects the H1N1 influenza propagation. They observed that boys were more likely to transmit influenza to other boys than to girls (and vice versa). According to these studies on the interaction between social contact network and the epidemic propagation, assuming people are equally mixed in social mixing groups and ignoring the crucial role of social contact patterns in epidemic propagation could lead to the loss of simulation fidelity. Thus, instead of dividing a population into social mixing groups, a population can be modeled as a graph (contact network). The individuals are vertices and their contacts are denoted by edges. The graphical model better represents the detailed social contact patterns than mixing group models. Simulating epidemic propagation over detailed social networks can better present the impact of the complexity of social contacts in person-to-person infectious disease propagation.

Individual-based and Interaction-based Models

Although there are various models for epidemic propagation simulation in academia, public health institutes and policy makers expect high resolution, high fidelity, and high performance computational models to support their planning and response to epidemics. A simulation environment is proposed recently [10, 8] in order to better understand the co-evolution of epidemic dynamics, human behavior, and epidemic control strategies. This proposed simulation environment is a complex simulation system, and its overall approach consists of four steps: 1) a model for creating a set of synthetic individuals, 2) a model for generating (time varying) interaction networks, and 3) a model for simulating the epidemic propagation process, and (iv) a model for representing and evaluating public disease control strategies and the co-evolution of individual behaviors. Control strategies and individual adaptive behaviors play a very important role in disease propagation. From human experiences with pandemics and recent epidemiology studies, we now have more knowledge of epidemic monitoring and

preparedness, effective medical treatments, and response strategies. Public health institutes are able to respond and control future epidemic outbreaks better than before through public disease control strategies.

The first step in building the proposed epidemic simulation environment is modeling synthetic individuals and their daily interactions. The individuals and their interactions are model as a directed, edge-labeled network. Nodes correspond to individuals in a population; edges represent contact between two end nodes. Each edge has a weight label that is the duration of contact and a type label that is the contact type (home, work, school, shopping, or others). The disease may be transmitted from an infectious individual to his contact neighbors in the susceptible state with a probability. In this dissertation, we assume the transition probability is defined as follow: If node u is in the infectious state, and his contact neighbor v is in a susceptible state on that day, then the probability of transmitting a disease from u to v is

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

where r_u^o is the probability of node u infecting any other node per unit of time of contact, r_v^i is the probability of node v getting infected by any other node per unit of time of contact, and $w(u, v)$ is their contact duration every day. A crucial assumption made in almost all epidemic models is that of *independence*: we assume that the propagation of infection from a node u to node v is completely independent of the infection from a node u' to node v. Similarly, an infected node u propagates infection to each neighbor v, independent of the other neighbors of u. This is a central assumption in almost all epidemic models and analytical results based on percolation.

The proposed simulation environment uses the above disease propagation model based on contact network to model the between-host-disease-progression and uses the SEIR model (Figure 2.1) to model within-host-disease-progression. Figure 2.2 illustrates an example of disease propagation over a contact network. The propagation starts from day 0 and only node A (a propagation seed) is infectious; others are all susceptible. The network evolves to day 10 when a transition occurs between infectious node A and its susceptible neighbor B, and node B becomes exposed on day 10. Without any interventions to prevent this propagation, the disease is transmitted to nodes C and D on day 20 and day 30 respectively, as illustrated in the top branch of Figure 2.2. If vaccination intervention is applied between day 10 and day 20 on susceptible nodes C and D, then the vaccines prevent them from being infected. Vaccination intervention changes the propagation trajectory. The individual-based epidemic simulation problem can be classified into two subproblems: the disease propagation simulation problem over a contact network, and the intervention decision making problem that decides when to apply interventions, who are intervened with and how to intervene.

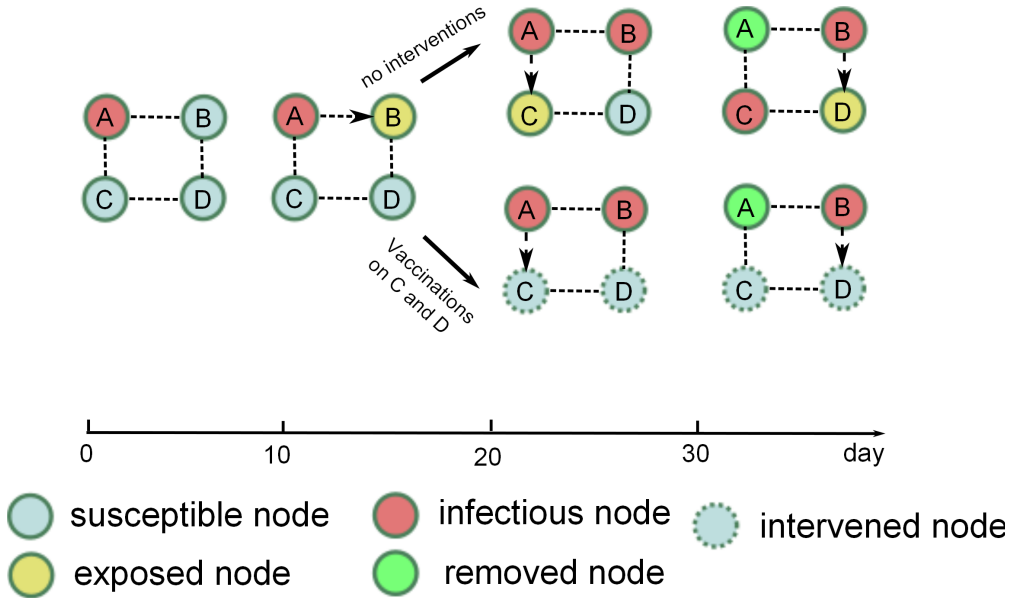

Figure 2.2: An illustration of the interaction-based disease propagation model with interventions. The nodes represent individuals and the edges represent their social contacts. The dark green nodes, yellow nodes, red nodes, and light green nodes are in the susceptible state, exposed state, infectious state, and removed state respectively. The arrows between nodes denote disease transition. The nodes with dashed circles denote vaccination.

2.2 Epidemic Intervention Modeling and Simulation

In the previous section, we briefly presented the value of simulating public health intervention and individual adaptive behavior along with epidemic propagation simulation. This section introduces the recent work on epidemic intervention modeling and simulation, and discusses some important but not well-resolved problems of intervention simulation in existing simulation systems.

The term *intervention* is widely used in epidemiology and the epidemic simulation area to represent all the possible exogenous events influencing the epidemic propagation process. Public health policies (such as vaccination campaigns) are the frequently examined and suggested control strategies. The individual adaptive behaviors for self-protection purposes (such as voluntary social activity reducing) are effective control measures, assuming that adequate amounts of a vaccine are unavailable [64, 44]. Therefore, this dissertation focuses on a subset of interventions: public policies and individual adaptive behaviors. We list some examples of these two intervention types in Table 2.1. The intervention model and its corresponding simulation framework introduced in this work for public health policy and individual behavior simulation also may contribute to other types of intervention simulation such as the co-evolution between epidemic propagation systems and fear propagation systems. But it is not the emphasis of this dissertation and we will reserve it for future work.

Table 2.1: Examples of suggested interventions to prevent epidemic propagation.

Measure	**Type**	**Application situation**
Vaccination campaign	Public health policy	Early stage of epidemic
Antiviral distribution	Public health policy	Adjunct to vaccination
Quarantine of household contacts	Public health policy	Epidemic is under way
School-closure	Public health policy	Children are the most vulnerable
Wearing face masks	Suggested behavior	The whole epidemic period

Epidemic interventions are applied to control the propagation of infectious disease by providing the public with pharmaceutical treatments (vaccine and antiviral) to promote their immunities or reducing transmission risk by social contacts (ascertained case isolation and school-closure). Modeling and simulating epidemic intervention within epidemic propagation is a challenging problem. It is non-trivial and worthy of research effort due to following reasons.

The World Health Organization(WHO) and early studies on epidemic control strategies have introduced several effective public health interventions and suggested individual protective measures to respond to emergent epidemic outbreaks [64, 33, 60, 26, 25]. Some of these interventions are enumerated in Table 2.1. These suggested interventions describe a set of response strategies and the infection situations in which to apply these strategies. For example, if children are the most vulnerable age group to an infectious disease or play a major role in the disease transmission, school-closure is strongly suggested. From the perspective of epidemic intervention simulation, we need to model interventions within a scenario. The intervention scenario expresses when and how interventions are applied during the epidemic propagation process. Take the school-closure intervention as an example. The timing of executing school-closures is an open question, and we can design different infection indicators to trigger this intervention, such as ascertained cases exceeding 1 % of the population or 1% of the school-age population. The pattern of closing schools is also open for design. The government may close all schools in the whole nation simultaneously or only close schools in the seriously infected areas. Public health policy makers could design countless and diverse scenarios by intervention combination, different intervention execution timing, and different execution strategies.

As a result, it is almost impossible to predict and enumerate all the possible intervention scenarios during the simulation system design period. Some existing epidemic propagation simulation systems [20] enumerate common interventions within the regular scenario. For example, if a school-closure intervention is studied, the most common scenario used is that all schools will be closed simultaneously. Modeling and simulating regular intervention scenarios can evaluate the effectiveness in these scenarios, but public health policy makers may investigate the intervention effectiveness in irregular scenarios or compare the effectiveness of different scenarios. We have to model and implement new intervention scenarios on demand. In previous intervention studies, the studied scenarios are often beyond the capability

Table 2.2: A comparison between current epidemic simulation systems and the user-expected simulation systems.

Model	**Experimental Performance**	**Scalability**	**Capability**	**Dev. Effort**
Differential equation [59, 36]	Fast using commodity computer	Up to the global population	Built-in interventions	Medium
Individual-based mixed-group [20]	Hours using cluster for national populations	Up to national populations	Predefined interventions	High
Individual-based interaction-group [7]	Hours using cluster for national populations	Up to national populations	Predefined interventions	High
User-Expected	Hours using cluster	Up to the global population	User-defined interventions	Low

of existing simulation systems. We believe this issue will continue to trouble us if it is not resolved.

Second, for a high fidelity study the public health policy makers may take into consideration factors that were not included in propagation models. For example, person-to-person infectious diseases have shown spatial propagation patterns in historic pandemics; therefore, public health policy makers may design a regional vaccination campaign based on the infection situation in the regions [44]. Geographic information about residents is not closely relevant to individual-based epidemic propagation models and it is typically not available in existing simulation systems. In epidemic simulation system design, integrating unknown extra information into intervention scenario simulation is an open and challenging issue.

Third, many existing epidemic simulation systems model their own intervention scenarios on top of their epidemic propagation models [20, 7, 14]. The same intervention scenario is simulated differently in these systems. The extra development effort of repeatedly implementing the same intervention can be avoided if intervention models are reusable. The epidemic intervention scenario needs a generic model on the basis of existing epidemic propagation models.

Since there are various suggested interventions and their disease controlling effectiveness is yet not completely known [64], it is necessary to evaluate their effectiveness before enforcing them in practice. One possible way of evaluating public health interventions is through high-fidelity computer simulations. We have made remarkable progress in building a simulation environment for public health researchers and policy makers to examine disease propagation and its intervention strategies in the face of emergent epidemic outbreaks. However, there still exists a gap between current epidemic simulation systems and the user-expected systems. Table 2.2 presents a comparison.

2.3 Database Supported Simulation

Epidemic propagation and its intervention simulations in individual-based systems are data-intensive. Epidemic simulation systems build synthetic population to simulate disease propagation in countries or cities. Such populations may include millions of synthetic individuals and each individual may have tens of demographic attributes [23]. Other than reducing the computation complexity of the simulation algorithms, efficiently organizing and storing such large volumes of data in modern high performance computer systems is also important. Current computer science is entering an age of database-centric computing, as evidenced by huge volumes of scientific data produced by laboratories, scientific simulations and surveys. Scientific research has altered its working pattern. Compared to the pattern of achieving conclusions by mathematical modeling and theory-based deduction, today more scientific discoveries come from data analysis, such as DNA sequences for life science and sky survey data for astronomy [30]. Available data for scientific research currently are huge and are still being produced worldwide [55]. The database-centric computing is emerging, but computer scientists are not well prepared for this challenge. Systematic and general approaches with an architecture that can scale up with data-intensive computing is under exploration [55, 30, 32, 31].

A high performance and scalable database-centric system for scientific data organization and storage is greatly needed. Traditional database systems designed for commercial transactions are not suitable for data-intensive computing. BeoWulf [52] and GrayWulf [50] are two projects building database-centric infrastructures on commodity computer clusters. In these projects, issues such as data allocation, computation locality, and I/O reduction have been studied, and data management services for front-end users also are emphasized to improve their productivity and capability. OGSA-DAI [4] studied database-centric computing from the perspective of grid computing. OGSA-DAI is a heterogeneous database architecture, and it integrates relational database, file-based database, and XML database technologies. It hides the data structure transformation and query translation details in OGSA-DAI, thereby improving query and analysis of the heterogeneous data.

Database-centric computing paradigms contribute to scientific research. Environmental and earth science request information systems that serve as data repositories and store geology data, atmosphere data, ecosystem data, and human activity data. Such information systems for environment evolution research are also expected for access to the datasets. A framework of a petabyte database system for High Energy Physics experiments was proposed and new issues emergent in building a petabyte database system addressed [12]. Aside from expecting success in building dedicated scientific databases, existing database systems have also been employed in scientific research. Finite Element Analysis (FEA) in physics has been attempted to manage large scale FEA datasets in databases [35]. FEA calculations are translated into relational calculus and computed by relational databases.

Most database systems in current applications are still using database prototypes proposed

two decades ago. Large scale datasets two decades ago were usually on the scale of gigabyte, and it was technically feasible to manage them in a single computer. The primary issue was avoiding latency between main memory and low speed physical disks. The databases two decades ago were also primarily for commercial transactions. Data today have dramatically evolved both in volume and structure. Those in data warehouses and produced in scientific laboratories are measured in terabytes and petabytes [13]. Because of the extreme volume, management using a single computer is no longer technically feasible. Computer cluster and grid computing are the direction for today's large scale data management. The data management services demanded today, such as data stream processing and data mining, are more complicated than before, and a revolution in database system design for today's data-intensive computing is now underway. A column-oriented database prototype was proposed, which performs more efficiently than traditional row-oriented prototypes in data warehouse analysis, decision support, and business intelligence applications [2, 3, 34, 1].

Chapter 3

System Design

Existing epidemic simulation systems are computationally efficient in propagation simulation, but they have several limitations in modeling and simulating complex public health interventions. Their propagation simulation and intervention simulation are coupled in an integrated system. The design of such an integrated system is complicated since their computations are different, and as a result the simulation subsystems have issues with usability and extensibility. In addition, complex interventions are either not able to be modeled or require considerable development effort. In this chapter, we revisit these challenges in epidemic simulation system design. We propose decoupling the disease propagation model and the intervention model, and modularizing this complex simulation system. A decoupled system can both simplify the complex system design and improve its usability and extensibility. We also explore simulating interventions using relational database systems to improve intervention modeling capability and simplify the development of intervention simulations. We study the feasibility and benefits of simulating interventions using relational databases. Finally, we present our formalization of a loosely coupled and database supported epidemic propagation and intervention simulation system.

As we presented in Section 1.4, decoupling the epidemic propagation simulation and its intervention simulation, and utilizing relational database systems for intervention simulation are our research hypotheses in this dissertation. To verify these two hypotheses, we redesign the epidemic modeling environment to apply these ideas in the simulation system. In this chapter, we will elaborate on our system design. We start with the motivations of applying these two ideas.

3.1 Motivations of System Decoupling

- **Modular design could simplify system design.**

  Epidemic propagation does not evolve as a closed system. The propagation process of-

ten is influenced by exogenous events such as epidemic interventions from public health institutes, or it co-evolves with other dynamical systems such as individual decision and behavior. Thus, the epidemic propagation process is typically simulated and studied with intervention. An epidemic simulation is a complex system, which has several simulation subsystems such as the disease propagation simulation subsystem, the human behavior simulation subsystem, and the public health intervention simulation subsystem. We could attempt to design a complete and integrated system that encompasses all subsystems. However, the system designer would need to acquire knowledge about all related subjects and then model these complicated subsystems in an integrated manner. In addition, an integrated system could cause problems in practice. If we plan to simulate disease propagation in large populations such as national populations, we have to deploy the integrated simulation system on a computer system with adequate computing capability. Usually such a powerful computing system is scarce. Moreover, a minor modification on a subsystem could cause a ripple effect of modifications in other subsystems. The subsystems, designed for particular simulation problems, are also hardly reusable. If we could instead modularize a complex simulation system to be loosely coupled, with functionality-explicit, and implementation-independent modules, then we not only simplify the system design, but also improve system extensibility and reusability.

- **Propagation simulation and intervention simulation are distinct in their computational characteristics.**

  Disease propagation, public health policy, and individual adaptive behaviors are key modeling issues in epidemic simulation. In this dissertation, we use the term *intervention* to represent public health policies intended to control epidemics and individual adaptive behaviors for self-protection. We examine public health institutes' suggested policies and recently studied individual adaptive behaviors. We find that propagation simulation and intervention simulation are distinct in their computational characteristics. They are compared in Table 3.1. First, we compare their models. Disease propagation simulation typically uses generic disease progression models with different parameters to represent various disease outbreaks such as ordinary seasonal flu and the 2009 H1N1 event. We use the SEIR model introduced in Section 2.1 to model within-host-disease-progression and the interaction-based model introduced in Section 2.1 to model between-host-disease-progression. The SEIR model and the interaction-based model with different parameters can model person-to-person disease propagation. However, it is challenging to explicitly model all recently studied interventions in a single model or function. For example, we cannot model the school-closure policy and the quarantine of household contacts policy (Table 2.1) using a single function. Second, the computational characteristics are different in these two simulations. The propagation simulation has both mathematical computation (such as probabilistic disease propagation between individuals introduced in Section 2.1) and logical computation (such as the state transition function in the SEIR model). In contrast, most computation

in intervention simulation is logical (here we assume the policies and adaptive behaviors are deterministic). For example, the school-closure policy simulation searches for targeted school-age children and changes their schedules. Finally, the data used in the propagation simulation are usually fixed. The propagation simulation data include the individual demographic information, the individual health information, and their social contact information. The intervention simulation may require a large scope of information such as household location [44] and individual vocation [43]. Since propagation simulation and intervention simulation are different, it is more reasonable to model and simulate them separately.

Table 3.1: Computational characteristic comparisons between individual-based epidemic propagation simulation and its intervention simulation

Characteristic	**Propagation**	**Intervention**
Model in different studies	Same propagation model with different parameters	Different functions for different interventions
Computation	Mathematical & logical computation	Logical computation dominant
Data	Individual health states and contact information	Individual health states and any relevant information

3.2 Motivations of Using Database

- Current High-Performance-Computing-based (HPC) propagation simulation systems enumerate interventions and implement each of them as an ad hoc HPC function. Although this methodology can almost simulate most intervention scenarios, its shortcomings include the following: 1) extending the existing intervention simulation system for a new intervention scenario is often complicated and labor-intensive, and sometimes such an extension is too complicated to implement; and 2) without exhaustive testing the new intervention simulation programs are error-prone. The development and testing period is the bottleneck in supporting real-time epidemic planning. Therefore, we explore simple methods to model and implement intervention scenarios instead of writing HPC functions. The new methods are expected to be capable of modeling interventions and comparatively effortless in their development and testing. Recent research [62, 63] on behavior simulation using data management technology suggests that using databases in intervention simulation could be a possible solution.

- The intervention scenario usually makes logical statements about the intervention decision based on the propagation situation. It could include some mathematical calculations, but the calculations included are not complicated. We may explore the logic

computation models and tools to simulate an intervention scenario. It is worth exploring the existing logic theories, such as relational calculus, to model intervention scenarios.

- The intervention scenario may consider supplemental information or co-evolving events. Thus, the intervention simulation module needs a data repository to manage supplemental information and share dynamic simulation results with the simulation systems of co-evolving events. One solution is using global memory as a data repository, but we have to consider its synchronization, data consistency, and so on. We could utilize a database as the data repository and utilize the automated services provided by database management systems.

- The simulation of intervention scenario could be complicated, and the dataset used in its simulation could be large and multi-dimensional. For example, we have studied an intervention strategy based on geographical information. The intervention scenario includes several phases, from spatio-temporal situation assessments, individual-level intervention trigger testings, and intervention decision making and action selection (see Section 7.1 for details). This complex intervention simulation requires optimization to support real-time intervention studies. If we can model and simulate epidemic interventions in database systems, then full-fledged databases will provide state of the art query optimization techniques. Such query optimizations are not easy to learn, and implementing them in our intervention simulation is challenging and time-consuming.

- The goal of building this modeling and simulation environment is to support epidemic intervention strategy making. Current decision support systems usually include a database as data repository and knowledge-base system, and its management system usually manages information and assesses situations. Databases have demonstrated their importance in modern decision support systems.

Driven by these motivations toward system decoupling and utilizing relational database system in intervention simulation, we explore redesigning the epidemic modeling and simulation environment. As presented in Section 1.3, we have to resolve existing challenges of building such a modeling and simulation environment. In the following sections, we present our proposed methods to address these challenges.

3.3 Intervention Modeling

As discussed in Section 1.3, modeling complex epidemic interventions in a realistic way with an epidemic propagation model is a challenging problem. In this section, we emphasize the intervention modeling and simulation problem. We will introduce our model of epidemic interventions along with a propagation process.

First of all, what is the epidemic intervention we have been talking about so far? For the purposes of this dissertation, **epidemic intervention** includes: 1) individual adaptive behaviors to protect oneself, one's family, one's community, and so on from infection; and 2) public health policies intended to control disease propagation. For the remainder of this dissertation, we use *intervention* for short. An intervention is comprised of epidemic propagation situation assessments and an intervention scenario.

Epidemic propagation situation assessment

Epidemic propagation situation assessment examines and evaluates disease propagation in the studied population. The assessment may evaluate the whole propagation process (primarily to examine the disease impact on public health and society) and the disease controlling effectiveness of applied interventions. From a computing perspective, we call this type of situation assessment off-line assessment, since we can perform them after the propagation process. Another type of assessment involves examining the propagation situation during the propagation process. We call this on-line assessment. The on-line situation assessment examines the propagation process so far, and intervention strategies may be adapted after the on-line situation assessments. For example, if there is an outbreak of an epidemic in Los Angeles, the entire state of California will be notified of the outbreak, and public health institutes may suggest that residents reduce unnecessary social activities. The on-line situation assessment is a critical step in the policy making process. It evaluates the effectiveness of applied interventions and adjusts intervention strategies at run-time. If not specified, the epidemic propagation situation assessment discussed later in this dissertation is an on-line situation assessment.

Intervention scenario

An intervention scenario usually describes who the intervened individuals are, as well as when and how they are intervened with. From our modeling perspective, an intervention scenario is comprised of intervention actions, intervention subpopulations, and intervention triggers.

Intervention action

An intervention action is what the intervened individual does. Examples of intervention actions include seeking a vaccine shot and reducing unnecessary social behavior (shopping, visiting parks, etc.).

Intervened subpopulation

An intervened subpopulation is comprised of the individuals who take intervention actions in response to public policies or their own decisions. Examples of an intervened subpopulations include school-age children stay at home when a school-closure is enforced, as well as people who voluntarily wear face masks after the public health institute warns of an ongoing epidemic.

Intervention trigger

An intervention trigger specifies when individuals are intervened. The intervention time could be a specific time (the second week after an epidemic outbreak), individual-level criteria (when the individual is sick), or aggregate-level criteria (when a percentage of the population becomes infected). An intervention trigger is associated with intervention actions and intervened subpopulations. When an intervention trigger is fired in our model associated subpopulations should take the associated intervention actions.

Table 3.2 shows our modeling of several interventions studied in the literature.

In our intervention model, the intervention scenario and situation assessment can refer to the state of the propagation process, but the mechanism of the propagation process and intervention scenario are not relevant. We speculate that the intervention model and propagation model can be loosely coupled in our model. In the following section, we will introduce a loosely coupled model separating the intervention process and the propagation process.

3.4 A Loosely Coupled Model

After presenting our intervention model, we can present our model loosely coupling the disease propagation process and the intervention process. In this section, we first introduce an epidemic propagation model, which integrates the individual-based and interaction-based epidemic propagation process and its intervention process. After introducing such an integrated system, we will discuss system decoupling in the integrated system.

The integrated, individual-based, interaction-based, and discrete time system for epidemic propagation and its intervention simulation is as follows:

- **Model the individuals, their states, and their social contacts as a graph**

  We model epidemic propagation processes using an individual-based and interaction-based model. The individuals are represented as vertices, and individual health state and demographic features are represented as properties of the vertex. The social contacts between individuals are represented by edges between vertices. The contact features, such as contact duration and contact type (classmates or work colleagues), are represented by edge labels. The within-host-disease-progression and between-host-disease-progression are modeled by two local state transition functions. We will present a formal graphical dynamical system in Section 3.5 which has been extended to model the person-to-person disease propagation process.

- **Model the sequence of propagation and intervention**

  Although propagation processes and intervention processes are continuous over time and could occur simultaneously in the real world, we approximate these two processes as follows. We model propagation as a discrete time system, and sequentially simulate the propagation process within a given time interval by calculating the local transition

Table 3.2: Our modeling of interventions recently discussed in the literature.

Intervention	**Sub-population**	**Triggering condition**	**Action**
Antiviral treatment of diagnosed cases [33]	Diagnosed people	Over 1% of population are infected	Reduce their infectivity
Home isolation of diagnosed cases [33]	Diagnosed people	Over 1% of population are infected	Remove their contacts with non-household members
School closure when disease prevalence is high [33]	All school-age children	Over 1% of population are infected	Remove in-school contacts between them
Deference of travel to unaffected areas [64]	People who live in affected area and plan to travel to unaffected areas	Over 1% of population are infected	Remove their contacts with people outside of affected area
Avoidance of contact with high-risk environments [64]	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 [44]	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 [25]	Individuals in the same household, school or workplace with diagnosed cases	Cases are diagnosed in this household, school, or workplace	Increase their immunity

functions for within-host-disease-progression and between-host-disease-progression of each vertex. We model interventions as discrete events at each endpoint of the propagation time interval. These two systems process alternatively. The propagation process simulates its propagation within the time interval and pauses at the end of the interval. Then the intervention process begins its simulation and modifies individuals' states or behaviors immediately at the time point (or with a programmed delay). After the intervention application, the propagation process resumes. Such an approximative model is reasonable and significant, as long as the timing of the intervention application has a negligible effect on the propagation process in the time interval. For example, if we choose the time interval of one day, and the hour during a simulated day at which a vaccine is taken (morning or afternoon) has a negligible effect in simulating epidemic propagation in one flu season, then our approximation will be reasonable without losing significant modeling fidelity.

- **Model intervention scenario**

  An intervention scenario in our model, introduced in Section 3.3, includes intervention triggers, intervention actions, and intervened subpopulations. The intervention trigger is modeled as a boolean function. The intervention action is modeled by a function mapping current individual states and current social contact pattern to new individual states and a new contact pattern. The intervened subpopulation is a set of individuals. In each intervention scenario, an intervention trigger associates with a set of intervention-action-intervened-subpopulation pairs. If the intervention trigger function is satisfied, the intervened subpopulation in each pair will take the associated intervention action. The intervention scenario represents the logic regarding which intervention action should be taken based on the state of the population in response to exogenous events. It typically quantifies a set of individuals to take a particular action. Take an approximative version of a school-closure as an example. The intervention scenario is to target all school-age persons, and the intervention action is to change their school schedule to a home schedule when a disease outbreak is monitored.

The whole simulation procedure of the disease propagation process and the intervention process is illustrated in Figure 3.1. As illustrated, the simulation procedure for each time step includes these primary computation tasks: propagation simulation, situation assessment, intervention scenario simulation, and intervention application. If we attempt to modularize the propagation simulation and intervention simulation, the question then is how to decouple the integrated simulation system. We propose making a cut around the black box for situation assessment and intervention scenario simulation, and decouple them from the integrated system. A loosely coupled system is illustrated in Figure 3.2. The left subsystem simulates the disease propagation and the effect of intervention actions on the graph. The right subsystem evaluates the situation and simulates an intervention scenario. Its input consists of the latest states of individuals and their social contacts (this module may store the previous states of the graph). Its output is a set of intervention-action-intervened-subpopulation pairs, which

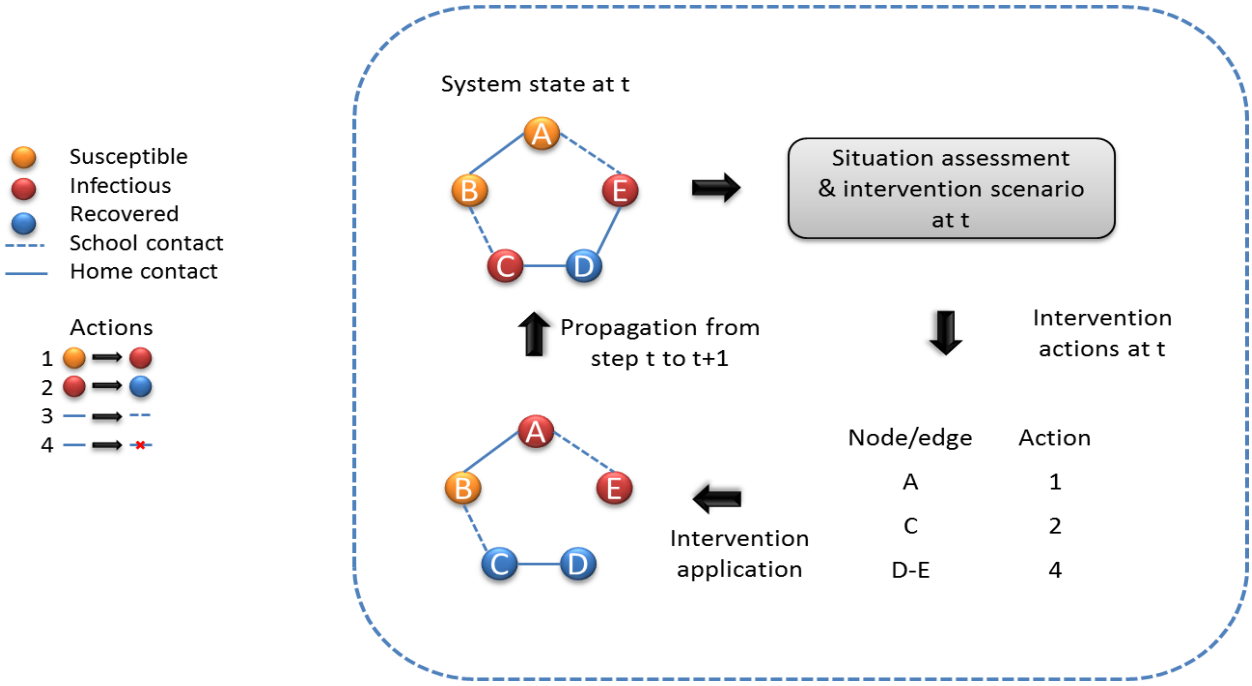

Figure 3.1: A diagram of an integrated simulation system for disease propagation simulation and intervention simulation. The pentagon represents a social contact graph with five individuals, A-E. The vertex colors represent the health states of individuals and the edges represent their contacts. In each discrete time step, the graph evolves from the previous state (the graph below) to the current state by simulating disease propagation. The black box represents the situation assessment and intervention scenario at the current step. The gray box decides the intervention actions (modifications on the vertex color or edge) and the intervened individuals. The intervention actions are applied on the graph to simulate the intervention effect. Finally, the disease propagation enters the next step.

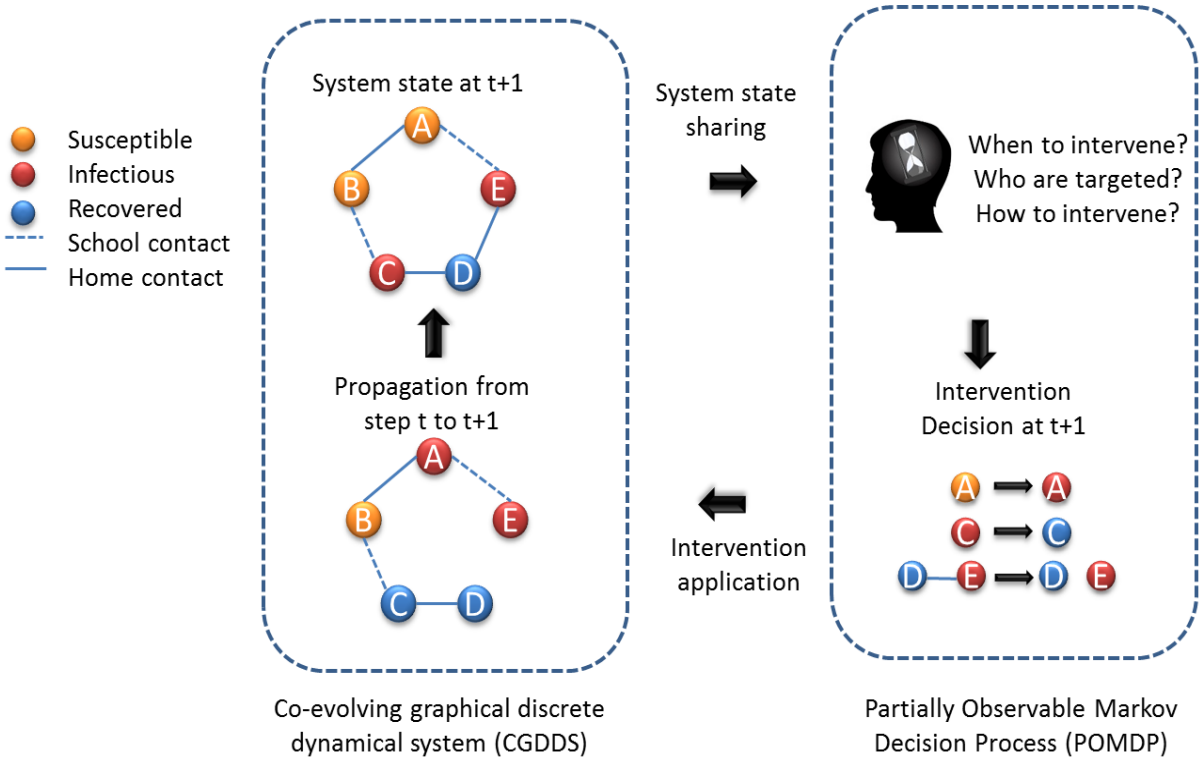

Figure 3.2: A diagram of a loosely coupled simulation system for disease propagation simulation and intervention simulation. The graph and the evolution process are the same as shown in Figure 3.1. The difference in Figure 3.1 lies in the system architecture. The propagation process and the intervention process are separated into two interactive subsystems. The left subsystem simulates the propagation process in the graph and applies interventions from the right subsystem. The right subsystem assesses the infection situation, simulates the intervention scenario, and decides upon the interventions.

specifies the intervened individuals and their associated intervention actions.

We assign the application of the intervention actions in the propagation simulation system because of the following reasons:

- Application of the intervention action in the propagation simulation system can minimize shared information between the subsystems. As a result, it reduces the communication cost. For example, if the intervention action is a function of "$property_a = 2 * property_a$" (changing the value of individual property a to 2 times the original value), the propagation simulation system could avoid moving the value of $property_a$ to the intervention scenario simulation system and compute this function locally. We follow Gray's advice "move the computation to the data rather than the data to the computation" to reduce the communication cost.

- The action application may have complicated mathematical computations, and we expect all of these are within the propagation system. As we analyzed in Section 3.1, we attempt to keep the intervention scenario simulation logical computation dominant. We attempt to simulate intervention scenarios in relational databases which are capable of logical computation, not mathematical calculation. Thus, we assign the intervention action application in the propagation simulation system and only leave the logical computation to the intervention scenario simulation system.

- Encapsulating the states of the propagation simulation, which are not necessary for the intervention scenario simulation, gives us better system modularity and security. If design of an encapsulated propagation simulation states is modified, for instance, by updating the data structure, the modifications will not affect the design of intervention scenario simulation. In addition, since we only allow the intervention scenario simulation system to change the graph states through the intervention action functions defined in the propagation simulation system, the security of the propagation simulation system is enhanced.

3.5 System Formalization

We have introduced two ideas, system decoupling and relational database utilization for epidemic propagation and its intervention simulation system design. We present a formal system here, in which the two ideas can be applied.

The formal mathematical model consists of two parts: 1) a CGDDS that captures the co-evolution of disease propagation, social network and individual behavior, and 2) a POMDP that captures various controls and optimization problems formulated on the phase space of this dynamical system. CGDDS is described in [6]. We will first extend this mathematical model so that: 1) it covers normal pharmaceutical and non-pharmaceutical interventions;

2) the system progresses in a synchronized manner; and 3) the vertex state modification function is separated into a propagation function, which changes vertex health state based on information of its neighborhood, and an intervention function, which changes other states of a vertex based on global information. Then 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 we overlay the extended POMDP over the extended CGDDS to form an interactive CGDDS, which formally describes our Indemics framework.

Extended CGDDS

An extended **Co-evolving Graphical Discrete Dynamical System** (CGDDS) $\mathcal{S}$ over a given domain $\mathbb{D}$ of state values and a given domain $\mathbb{L}$ of label values is a triple $(G, \mathcal{F}, W)$, 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 $s_i = (s_i^1, s_i^2, \ldots s_i^k) \in \mathbb{D} = (\mathbb{D}_1 \times \mathbb{D}_2 \times \cdots \times \mathbb{D}_k)$, 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 type (home, school, work, shopping, or others).

2. Functions $\mathcal{F} = (f, g^V, g^E)$, 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: 1) the current states of the agent, 2) the states of all its neighboring agents, and 3) 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 the states of vertices based on the current state of the whole graph. We assume V is constant.

Let $g^E = \{g_1^E, g_2^E, \dots g_{k_E}^E\}$ 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 the 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 $W = w_1^1 w_1^2 \dots w_1^{j_1} \dots w_t^1 w_t^2 \dots w_t^{j_t} \dots w_T^1 w_T^2 \dots w_T^{j_T}$ 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, $w_t^1 w_t^2 \dots w_t^{j_t}$, 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, the f functions correspond to between-host disease propagation, the g^V functions correspond to within-host disease progression and pharmaceutical interventions (**PI**'s, e.g., antiviral, vaccination) and other updates that directly change people's states (e.g., increase of fear level as the epidemic takes off), and the g^E functions correspond to non-pharmaceutical interventions (**NPI**'s, e.g., school closure, quarantine, social distancing) that change the graph structure.

Note that although theoretically an extended CGDDS model can be implemented in a simulation tool, e.g., EpiFast is an implementation of an extended CGDDS of epidemic dynamics, 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, so 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 3.3 we elaborate on an extended CGDDS using our EpiFast implementation [14] as a concrete example.

Algorithm 1 shows the propagation loop with static interventions (intervention actions and targeted individuals are determined before the propagation process) in the extended CGDDS. Note that each step is computed in a synchronized way among vertices to make sure that the execution order does not matter. That is, the network and state updates are *realized* only when all vertices have finished executing the relevant functions, e.g., in local state transition (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 far from 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 the real world system 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

Table 3.3: Extended CGDDS in EpiFast implementation: a concrete example.

CGDDS component	**In EpiFast**
$\mathbb{D}$	SEIR state, infectivity, vulnerability, diagnosis, symptomatic state
$\mathbb{L}$	contact duration, type
f	stochastic transmission from infectious vertex to susceptible vertex with probability determined by infectivity of infectious vertex and vulnerability of susceptible vertex and duration of contact between them
g^V	g_1^V: seeding (e.g., for 5 pre-selected vertices set their state 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: diagnose (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 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

with the propagation process. To this end, we introduce POMDP to capture the online decision making process for interventions.

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 making process, which is the intervention scenario we mentioned previously. We specify our extended POMDP using the terminology in [47]. 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 t, an observation function $o : S \mapsto O$, and a reward function r which gives reward for taking action $a \in A$ while system state is $s \in S$. Our extended POMDP is formally specified as follows.

Algorithm 1: Pseudo-code of computations in the extended CGDDS.

for $t = 0$ **to** T **do**
  let $w_t^1 w_t^2 \dots w_t^{j_t}$ be the t^{th} substring in W;
  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;

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 $n + \binom{n}{2}$.
2. Actions A are interventions that modify vertex states and edge labels. A responds to the intervention action, such as taking an antiviral and staying home.
3. State transition t computes propagation with interventions.
4. Reward function r can be number of infected cases, or it can be a combination of the economic and social cost of applying interventions and economic and social cost of disease outbreak (mortality and morbidity), or it can consist of vertex level local functions.

The intervention policy to determine actions based on observations can be history dependent $\pi_h : O^* \mapsto A$ [47], which make the process not markovian. Non-markovian interventions actually are common. With the extended POMDP overlayed on the extended CGDDS, CGDDS drives the system to a new state in each time step, POMDP derives intervention actions based on the 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 intervention process described in Algorithm 2 is adopted by several HPC-based epidemic simulation systems [14, 16]. However, the mapping from system states to intervention actions in Algorithm 2 is hard-coded in the simulation system. This characteristic causes at least the following critical drawbacks: 1) unless all possible intervention scenarios can be predicted precisely and enumerated exhaustively in the system, re-engineering on the intervention mapping for new policy scenarios is inevitable; 2) the data that can be included in vertex state for computing interventions is limited to what is already implemented within the simulation tool; and 3) there is no control and optimization based on the reward function. As we have mentioned in Chapter 1, we need freedom in these three aspects to assist real-time epidemic planning. A Database system and its query language are the key to tackling these issues. Relational query language based on relational algebra provides us a powerful programming language to express the intervention decision making process (intervention scenario), and it is closer to natural language than the programming language (such as C++) used to simulate intervention in previous systems.

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 S;
- apply each w_t^j function to modify vertex states or edge labels;
- apply all f_i functions to change the states of all vertices simultaneously;

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 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 $w_t = w_t^1 w_t^2 \dots w_t^{j_t}$;
- 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 3.5 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 antivirals are 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.

3.6 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 intervention changes the social contact network by adding or deleting edges, or modifying the edge labels (usually contact durations). The second type of intervention changes 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 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 it doubles exponentially according to 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 interventions using a relational query language. For the purposes of this dissertation, 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 $\mathcal{S} = \{(S_1, A_1), (S_2, A_2), \ldots\}$ 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).

3.7 Chapter Summary

This chapter presented our proposed approach to resolving the research questions on which this dissertation focuses. We enumerated the motivations for modularizing the epidemic simulation system and decoupling the intervention simulation from the epidemic propagation simulation. We presented our loosely coupled system design which separates the intervention simulation from the integrated simulation system. After system decoupling, we discussed the motivations for simulating intervention scenarios using relational databases and presented our model which uses that method. Following the introduction of these two ideas—decoupling the system and using relational databases—we presented a formal, loosely coupled system. This formal system will instruct our system implementation in the following chapter to evaluate our research hypotheses.

Chapter 4

System Implementation

In Chapter 3, we presented a loosely coupled and database supported system design for epidemic propagation and intervention simulation. We will now follow this system design and build a simulation system in order to verify our research hypotheses. This chapter introduces our implementation of the proposed system design. We first present its architecture and modules, then introduce its database design. Finally, we introduce our design of generic Application Programming Interfaces (APIs). With these APIs, we can build various special purpose applications extended from this system.

4.1 Architecture of Indemics

We have presented an interactive system design for individual-based epidemic propagation and its intervention simulation in Chapter 3. In the abstract interactive system, one subsystem simulates the epidemic propagation process, another the intervention scenario, with a computational hub orchestrating these two subsystem computations and exchanging their data. Based on the abstract interactive simulation system, we developed an Interactive Epidemic Simulation framework called *Indemics* for epidemic propagation and its intervention simulation.

4.1.1 System Architecture

As shown in Figure 4.1, 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).

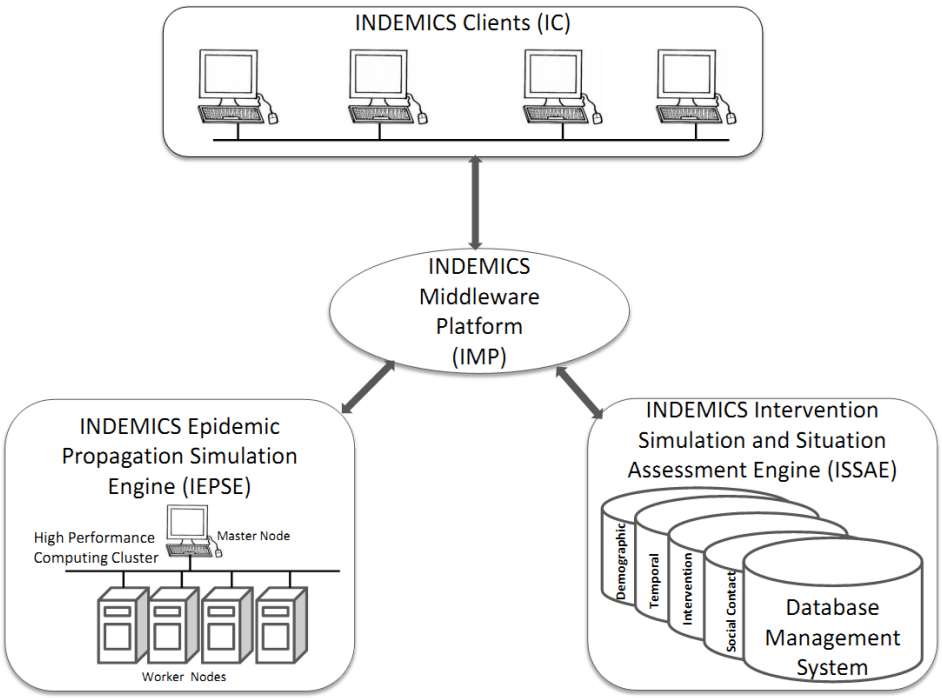

Figure 4.1: The high level architecture of Indemics. The Indemics Middleware Platform coordinates and synchronizes the communication between the Epidemic Propagation Simulation Engine, the Clients, and the Intervention Simulation and Situation Assessment Engine.

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 communication dependencies between these loosely coupled modules distributed across a network, these modules are connected in a star-shaped architecture, where the 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 4.1.1 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.

Indemics Middleware Platform

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

To account for differences in data formats across modules, the Indemics Middleware Platform is responsible for appropriate data transformations to facilitate communication. Also, to make the Indemics framework independent of specific implementations of its components and hide the implementation details of the message communication layer, Indemics abstracts the interactions between the Indemics Middleware Platform and other components and wraps them with a set of APIs which are part of the Indemics Middleware Platform. Section 4.1.1 will introduce the synchronization mechanism which the Indemics Middleware Platform uses to coordinate simulation between the distributed modules.

The implementation of the Indemics Middleware Platform 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 of the Indemics Middleware Platform has two sub-components: the Indemics adapter and the Indemics server.

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 done using any language that simplifies the interactions with the simulation engine as well as Indemics clients. In the current Indemics system, the adapter that interfaces with EpiFast is implemented using C++ and the interface with Indemics client is Java-based.

The Indemics server is capable of supporting multiple concurrent simulation sessions. As shown in Figure 4.2, each simulation session is registered with the simulation pool of the Indemics server and is managed by a session manager. The session manager manages appropriate data being passed from the database and Indemics client to the appropriate simulation session based on a session identifier. A message and data deliverer module of the Indemics server handles data transformations for data being passed to and from the database system.

Indemics Epidemic Propagation Simulation Engine

The epidemic propagation simulation engine provides the core computations that simulate disease propagation over a social contact network. The disease propagation simulation computes the evolution of disease 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 also can be enhanced with desirable features such as rollback and checking point, to allow users to reprocess disease propagation over a

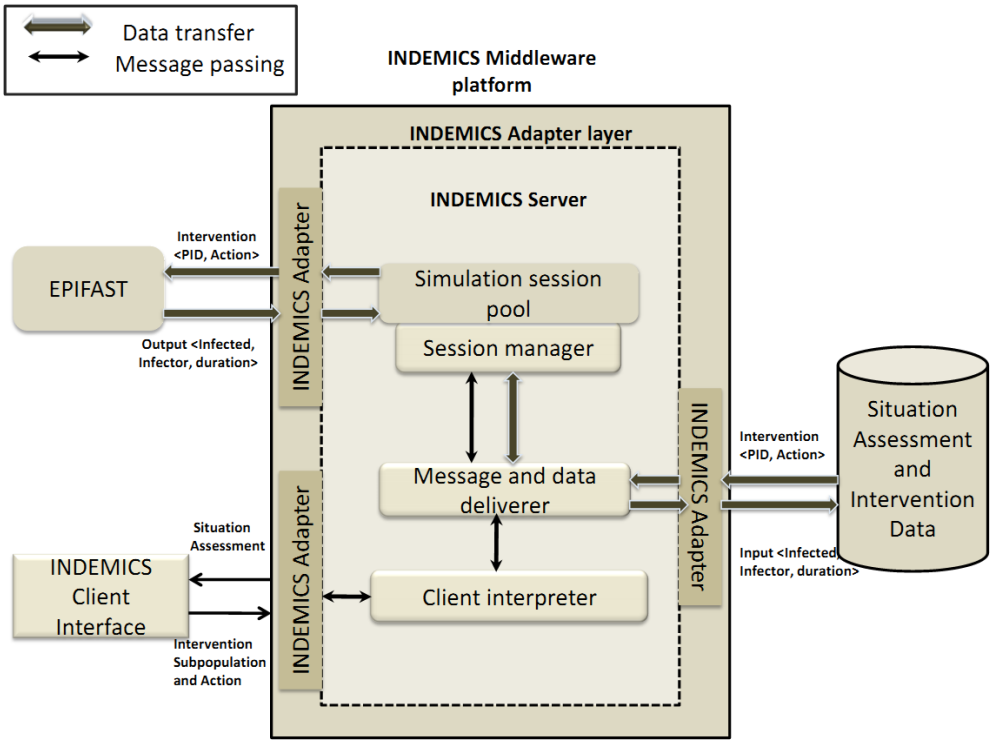

Figure 4.2: Details of Indemics architecture. This figure shows the architecture of Indemics, its modules and communication between them in detail. The Indemics server plays the role of message/data delivery center. The Indemics adapter hides the details of data transformations and communication by providing a set of APIs to EpiFast, Indemics clients, and the database.

given time period, perhaps with a different set of interventions.

In our current Indemics implementation, we have chosen the extended version of EpiFast [14] as the disease propagation simulation engine. However, as described before, any other CGDDS based simulation tool can fit into 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 3.5. 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 some random probability that is defined in the EpiFast algorithm.

To support external interventions, EpiFast has been modified to include an extended CGDDS system as described in Section 3.5, 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 f and g respectively in Section 3.5 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 sub-population 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 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 back 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, which ensures scalability and efficiency. However, since EpiFast holds the current state of all agents of the simulation in memory during a given time step, the number of concurrent users that it can support is constrained by the size of the main memory available. This limits the total number of concurrent simulation sessions supported by the Indemics system.

Indemics Intervention Simulation and Situation Assessment Engine

As discussed earlier, we use relational databases to support ISSAE. 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: social contact network data representing the set of proximity relationships for the given region, demographic data about individuals in the given region, temporal data about disease transmission at every time interval, and intervention data about disease control strategies applied.

The Indemics Client

The Indemics Client (also known as the user interface) provides an interface to the Indemics system. The 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, which is part of the Indemics middleware infrastructure.

Our current implementation of the IC module provides an interactive console interface to allow users to input run-time instructions to query the system state, intervene with the system dynamics, and control the simulation (e.g., rollback, pause, and resume). Indemics also supports a batch client (see examples in Chapter 7) which uses a script file consisting of a set of interaction rules to automatically feed the instructions to the Indemics server. This batch script client interface has embedded SQL (Structured Query Language) 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 selection of the type of interventions to be applied on a 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 databases for retrieving data to apply interventions.

Our Indemics implementation also has 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 analyzing simulation results graphically [21]. Using the web interface, users (epidemiologists and public health policy makers) can easily set up experiments with complex interventions, without having to know anything about the HPC environment or the Indemics deployment. Figure 4.3 shows a screenshot of the ISIS interface for configuring a simulation experiment with complex interventions using the Indemics model.

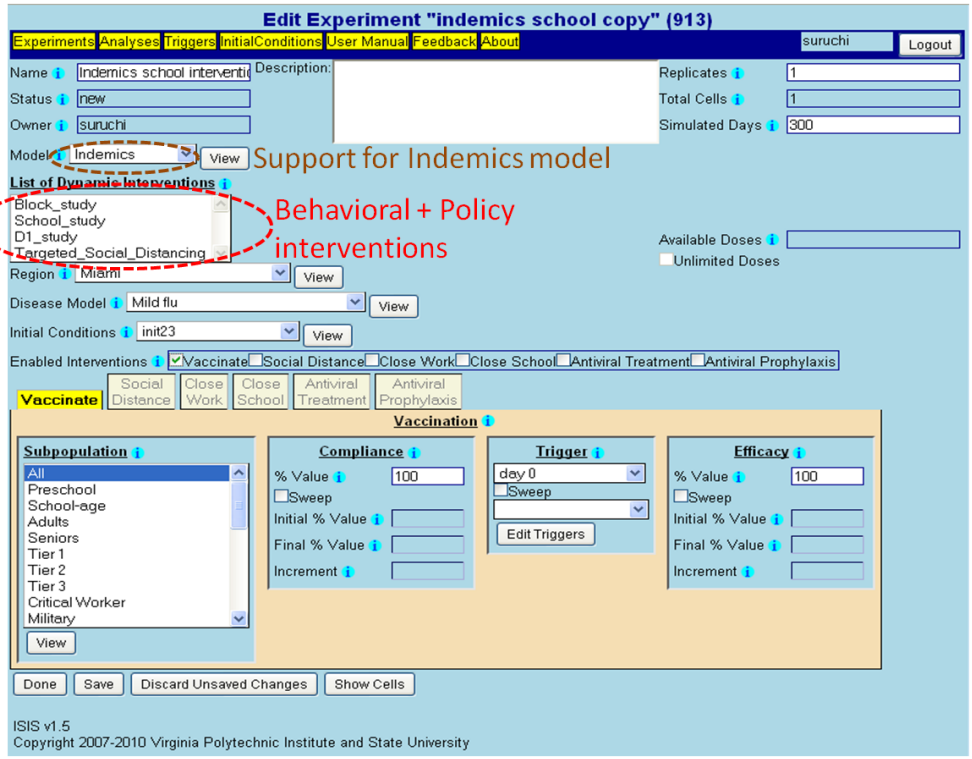

Figure 4.3: Screen shot of ISIS web-based interface for running simulations using Indemics. Using ISIS, users can set up and manage experiments for studying the effects of complex interventions to epidemic propagation simulation. Implementation details of the ISIS interface for Indemics have been explained in [21].

Synchronization Between Modules

Indemics is a loosely coupled and distributed simulation framework. Thus, its simulation process needs coordination and synchronization between its modules. In this section, the data flow and synchronization between modules within Indemics will be introduced. As explained in Section 3.5, the epidemic propagation and the intervention simulation within Indemics form a discrete time procedure. We introduce the message-sharing synchronization mechanism in Indemics and data flow between the four key distributed modules (IC, IMP, ISSAE, and IEPSE) in one discrete time step as an example to explain the entire discrete time procedure.

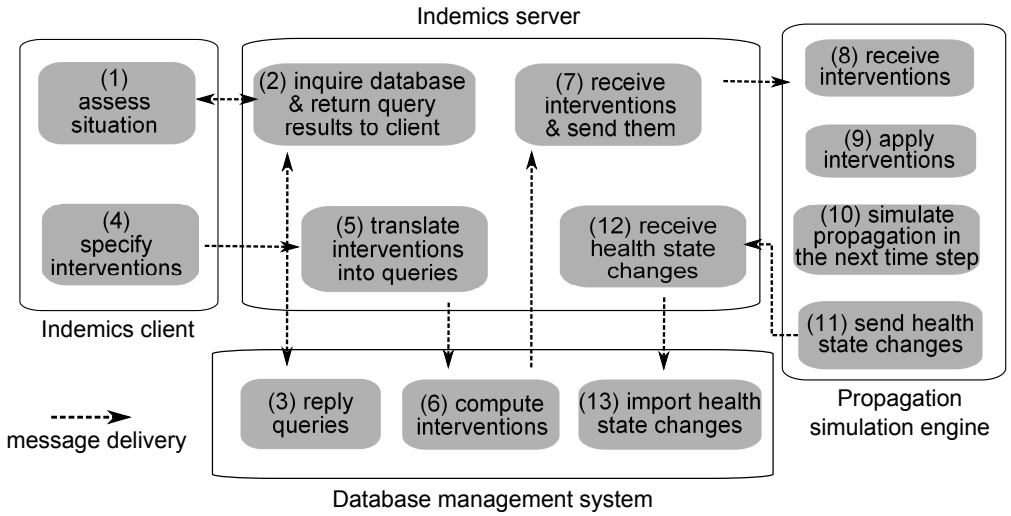

Figure 4.4: The message-sharing synchronization mechanism for Indemics modules. The Indemics server synchronizes the computations in the simulation engine and the client, and queries the database. The server also relays the data between the database and the simulation engine.

Figure 4.4 illustrates the synchronization mechanism and Figure 4.5 illustrates the computation flow in Indemics in one discrete time step. We assume the starting point of each time step is when IEPSE finishes the epidemic propagation simulation and exports the new propagation situation to the ISSAE (which means new infection data are sent to the ISSAE and stored in the database). At the starting point, a client first assesses epidemic propagation so far and decides which intervention scenarios will be applied at the current time step. When the client assesses the propagation situation (step 1), the assessment commands are sent from the IC to the IMP, where the assessment commands are translated into database queries in SQL, then the IMP queries the database using the translated database queries (step 2). The query results from the database are about the situation assessment (e.g., the infected subpopulation size) and the results are returned from the IMP to the client (step 3). During the situation assessment period, multiple clients (as a regular user or a

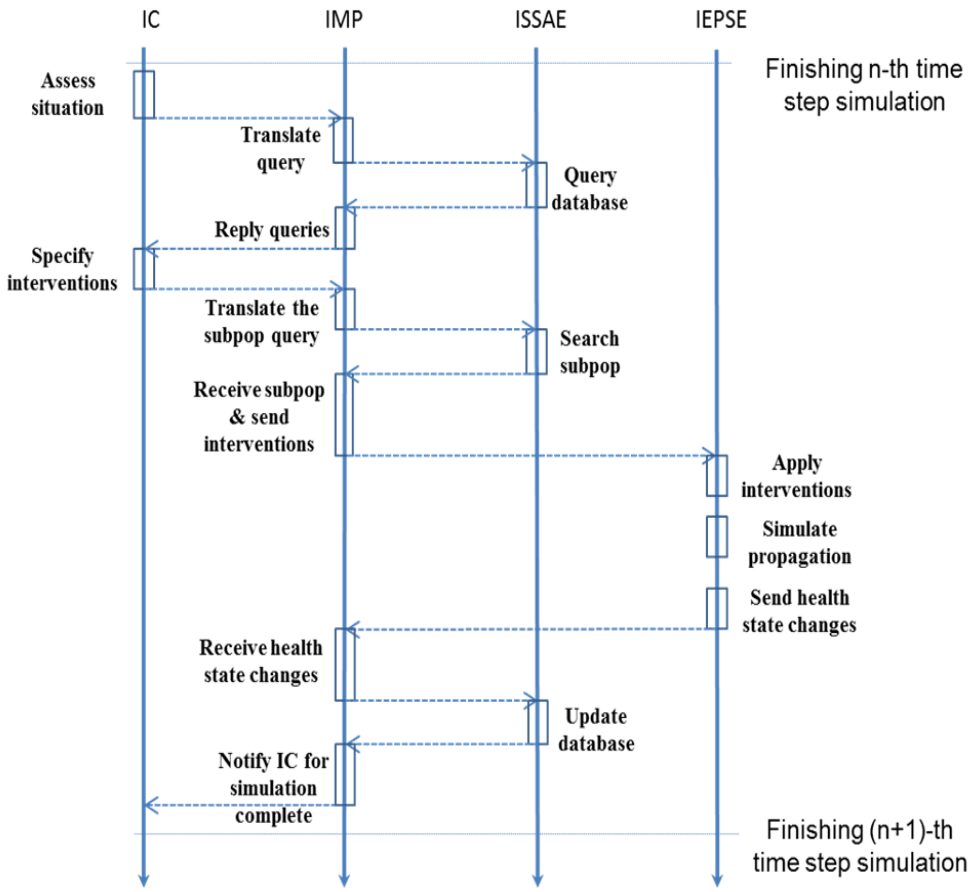

Figure 4.5: The computation flow in Indemics. It shows the computation flow in one time step between the four modules in Indemics.

super user) can assess the same simulation session concurrently. The situation assessment period ends when the client registered as a super user (who decides interventions) decides to specify his/her intervention scenarios to the IMP (step 4). When the situation assessment period ends, situation assessment requests after the situation assessment period are rejected or blocked by the IMP, but the unfinished situation assessment requests submitted earlier will be executed. The client registered as a super user also will be blocked for any situation assessments, until he/she finishes the intervention scenario specification for the current time step. After the super user describes the intervention scenarios and submits them to the IMP, the IMP translates the intervention scenarios into relational database queries in SQL (step 5). The intervention triggers are tested first, and then the intervened subpopulations are targeted and the IDs of the intervened individuals are returned from the database (step 6). The individuals' IDs are associated with the corresponding intervention actions specified by the intervention scenario. The IMP sends the pairs of individual ID and intervention action to the IEPSE, and waits for the IEPSE to apply the interventions at the current time step

and simulate the epidemic propagation at the next time step (step 7), until the IEPSE starts to share the new infection simulation data. The IEPSE remains in a waiting state until the IMP notifies the IEPSE of the interventions to be applied at the current time step. The IEPSE wakes up and receives the pairs of individual ID and intervention action from the IMP (step 8). When interventions are received, the IEPSE applies intervention actions on the targeted individuals (step 9) and continues the epidemic propagation at the next time step (step 10). After the propagation simulation, the IEPSE notifies the IMP to receive the new infection simulation data and sends the simulation data (the individual health state changes) to the IMP and waits for the IMP to deliver any future interventions (step 11). When the IMP finishes receiving the simulation data from the IEPSE (step 12), it imports the health state changes in the database (step 13). The clients associated with this simulation session will be unblocked. At this point, the simulation of one time step in Indemics is complete.

4.1.2 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 a minimum amount of data transfer and optimal frequency of data exchange between the modules, in order to satisfy performance requirements.

IEPSE

The IEPSE module in Indmiecs could have different implementations following the CGDDS model. In this dissertation, we use EpiFast as a concrete implementation of IEPSE to introduce data abstractions in IEPSE. 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 set of vertices representing individuals of the population and E represents contact between them. Each vertex $v \in V$ has an associated vector $\mathbf{V} = (pid, h, t_1, t_2, l_1)$ where:

- pid is the person identifier for a given vertex,
- h is the health state based on 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 to the vertex.

Each edge E is represented as a vector $\mathbf{E} = (V_1, V_2, p)$ 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 edge attributes and can simulate edge deletion by using an appropriate edge label.

The other input to EpiFast is the list of interventions selected by users from the ISSAE. For implementation, interventions are represented as $I = (pid, \mathbf{A})$, 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 intervention to be applied such as vaccination, social distancing, anti-viral,
  - *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 to the targeted population.

After the intervention I is obtained from Indemics as input at every time step, EpiFast computes disease propagation in the form of 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,incDur*) where:

- *infected* represents the set of newly infected vertices in the current time step,
- *infector* represents the corresponding vertex identifiers that infected them,
- *infDur* is the duration for which the vertex would remain in the Infectious disease state,
- *diagnosed* are the vertex identifiers that are not yet symptomatic and
- *incDur* is the period for which the diagnosed vertices remain asymptomatic.

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

ISSAE

As described before, the ISSAE stores and processes four kinds of datasets: social contact network data N, demographic data R, infection dendogram data D, and intervention data I_n. 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 a population.

The social contact network data N is stored as 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, stored in the database so that interventions based on social contact network structure can be formulated.

Temporal data relevant 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 T = (*infected,infector, infDur*) where

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

ISSAE is used to support situation assessment and intervention simulation. The output obtained from ISSAE is given by $I(S', A)$ where the set S' contains person identifiers on whom to apply interventions and A represents intervention actions. IMP sends this to EpiFast, which, when it resumes computation, uses the new vertex and edge labels to evaluate epidemic propagation for the next time step.

The output from ISSAE is given by $I(S', A)$ where set S' contains person identifiers on whom to apply interventions and A represents intervention actions. This is supplied as input to the EpiFast propagation engine for computation of 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 infectious 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.

4.1.3 Discussions of the Architecture

The loosely coupled and database supported simulation framework could be designed in different patterns. The reason we designed the architecture of Indemics in a star shape (Figure 4.1) is because that architecture has the following merits compared to other types.

- Modules in a star shaped architecture are loosely dependent. ISSEA and IEPSE are separated by IMP and both of them share the dynamic simulation results through IMP by computer network. The modules in this system design are more loosely dependent than a system design that shares dynamic simulation results directly between ISSEA and IEPSE. Another benefit of the loosely dependent system is that updates to the modules may require less reengineering work.

- A framework based on star shaped architecture can be deployed in a distributed computing environment. The four primary modules in this star shaped architecture are connected only through a computer network. It is a very useful feature. If all the modules have to be deployed in a single computer system, that system requires powerful computational capability when we study epidemic propagations in large populations. Therefore, it is more practical to deploy Indemics in a distributed computing environment.

- One weakness of this star shaped architecture is that it incurs extra communication costs for sharing dynamic simulation results between ISSEA and IEPSE. Dynamic simulation results have to be delivered between them and relayed by IMP. We will study the performance overhead of Indemics in Chapter 6.

Table 4.1: A capability comparison between EpiFast only and Indemics

Capability	**EpiFast**	**EpiFast + Indemics**
Scope of supported intervention scenarios	Predefined intervention scenarios only	User can define intervention scenarios intervention scenarios
Information used in intervention scenarios	Individual health states and social behaviors only	Any relevant information about the population
Prerequisite for creating new interventions	Knowledge of HPC, EpiFast code	Knowledge of SQL
Modifying/creating intervention scenarios	C++, typically months of development effort	Scripting, typically hours or days of development effort
Multiple co-evolving systems	Single simulation	Support multiple co-evolving systems
User online interaction	No interactions	Online analysis and simulation navigation

The current version of Indemics framework extends an existing HPC-based epidemic simulation system, EpiFast [14]. We enumerate several essential features of an epidemic simulation system in Table 4.1 and compare these features in EpiFast and Indemics. From the comparisons, we believe Indemics improves both capability and human productivity. By integrating EpiFast and Indemics, we forge a new simulation system supporting a wider scope

of intervention simulations, and enabling run-time user interaction with the simulation process. In addition, users can express their intervention strategies using an easy-to-learn and easy-to-understand script language, which remarkably improves the human productivity of performing case studies.

4.2 Database Schema Design

In this section, we will introduce our database design for Indemics in detail. Our database design is based on the following data management problems: data access control in concurrent sessions, computational efficiency of assessing infection situations and simulating interventions, and efficiency of importing dynamic results from the propagation simulation system into the database.

Intervention simulations with complicated scenarios may need supplemental data about the studied population such as its detailed demographic data and geographic data. The supplemental data may be integrated with dynamic simulation results to test intervention triggers and select intervention subpopulations. Dynamic simulation results such as the individual health states are retrieved from the propagation simulation system and imported into the database. Data stored there can be classified into two sets: static and dynamic. Static data are read-only, and are protected from being modified and overwritten at run-time. These include demographic data, household data, etc. Dynamic data can be read and overwritten at run-time. Run-time simulation results about individual health states and their behaviors are dynamic data. In addition, the studied intervention scenario may accumulate dynamic results or integrate dynamic results with static supplemental data, and store the interim query results in the database. For example, in the simulation of a school closure policy (see Section 5.2 for more details), the children' health states in every school need to be monitored for the school-closure decision making. In order to monitor the health states of schools, the individual health states and school-child relationship are intergraded. The health states of schools are interim query results stored in the database.

Static data are read-only, and are shared by different simulation sessions. For example, the geographic data of a population can be reused in different concurrent sessions simulating the propagations in this population with different disease models or intervention strategies. Dynamic data are session-associated, and should be exclusive in concurrent simulation sessions. Dynamic data in each simulation session are about the propagation process. Our solution is to share tables storing static data between concurrent sessions to reduce the database storage, while building dedicated database tables for the dynamic data of each session.

From a performance optimization viewpoint, the organization of static data should be read-optimized, and the organization of dynamic data should be both read-optimized and write-optimized. Most of the static database tables in Indemics are indexed, since indexing read-only static data in advance can remarkably speed queries. The dynamic database tables

Table 4.2: Schema of database tables storing dynamic data in Indemics.

Table Name	**Schema**	**Description**
Diagnosed_cases	pid number day number	Which day the individual 'pid' is diagnosed
Infection_cases	pid number infected_day number recovered_day number transmitter number infection_location number	The individual 'pid' is infected on day 'infected_day' and recovers on day 'recovered_day'. 'pid' is infected by 'transmitter' at location 'infection_location'
Intervention_history	action number day number pop_size number	Intervention action 'action' has been applied on day 'day' on a subpopulation whose size is 'pop_size'

Table 4.3: Database design strategies for static and dynamic data.

Feature	**Static data**	**Dynamic data**
Content	Demographic data Geographic data Social behavioral data	Health states Client activities Interim analysis results
Optimization	Query optimization (indexing, partitioning)	Query and update optimization
Access control	Shared by concurrent sessions	Private and exclusive in concurrent sessions
Schema	No fixed schema	Fixed schema
Normalization	Unnecessary Query efficiency is the first priority	Unnecessary. Inserting, not updating Query efficiency is the first priority

are updated frequently. Updates on an indexed table cause index rebuilding. From our experiments, we found that performance loss for index rebuilding outweighs the performance gain from the queries on indexed tables in most of our experiments. Thus, the database tables storing dynamic data are typically not indexed in Indemics.

In relational databases we may consider table normalization. For example, demographic data and household data can be stored in a single table (table schema as [person_id, age, gender, household_id, household_location, income, ...]), or in two tables (schema as [person_id, age, gender] and [household_id, household_location, income, ...]). Normalization or not depends on a concrete intervention scenario. If the script modeling the studied intervention queries both demographic data and household data, the schema with one table is more efficient, since it avoids a joint operation. If the script only queries one of them, the second schema with two tables is more efficient, since the tables are smaller and the disk I/O cost is reduced. Therefore, the schema of static data is not fixed in Indemics. We adjust schemas according to the studied intervention scenario during the simulation preparation period for better query efficiency. However, the schema of dynamic data is fixed, because Indemics has to know the exact schema in order to import the dynamic simulation results into the database at runtime. Since database update is a costly and slow database operation, we insert variations of the individual health state in the database instead of keeping the entire history, in order to speed the import of the dynamic simulation results. Table 4.2 describes the schemas of the dynamic database tables.

We summarize our database design based on the three data management problems proposed earlier (data access control in concurrent sessions, efficiency of assessing situations and simulating interventions, and efficiency of data import) in Table 4.3.

4.3 API

One of the expected features of Indemics is better extensibility than previous epidemic simulation systems. Indemics should support building many epidemic simulation-related applications. In order to make Indemics extensible, APIs are used. In this section, we will introduce our work on the Indemics API design.

4.3.1 Motivations

Indemics already provides a simulation platform for public health exports to study epidemic propagations with interventions. What are the motivations driving us to design the Indemics APIs? Here we present motivations that have inspired us to create Indemics APIs.

- Many previous epidemic propagation simulation systems have extensibility limitations. It is challenging to use these simulation systems as core simulation engines and build

epidemic propagation simulation-related applications from them. Figure 4.6 presents a system architecture without APIs. Front-end users can only access services provided directly by Indemics. Figure 4.7 presents a system architecture with well-designed APIs. Special purpose applications can implement their functions by requesting the general services provided by Indemics. With well-designed APIs, Indemics can serve as an extensible and flexible epidemic simulation core engine, from which many epidemic propagation simulation-related applications could be extended without a significant development effort.

- Second, it is almost impossible for simulation system designers to know and accurately predict all the services required by special purpose applications. For example, an educational game based on epidemic propagation simulations could be an extension from Indemics. However, interactions between game players and the core simulation engine may not be well known during the Indemics design period. With well-designed APIs, game developers can reuse Indemics as a core engine and call Indemics APIs to build applications.

- Third, a set of well-designed APIs can make technical integration between the back-end system and the application layer easier. The system designers of Indemics usually do not know what type of applications will be extended from it. The application designers understand how to make good use of the simulation data and the services provided by Indemics, but it is not necessary for them to understand the mechanisms of Indemics. With Indemics APIs, the simulation system designers and application designers can work separately and the APIs ease system integration.

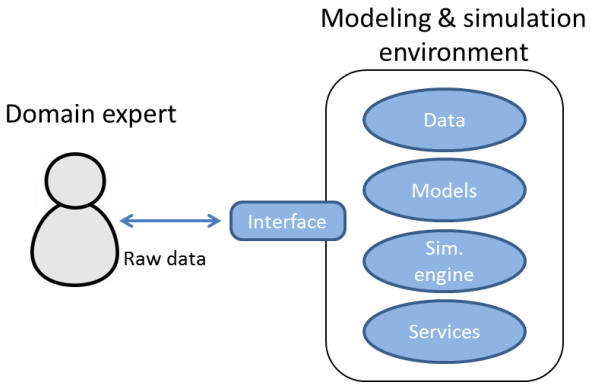

Figure 4.6: An architecture of Indemics without APIs. In this architecture, front-end users and applications can only access the services designed by Indemics designers and the raw data from Indemics. It is not trivial to customize the services provided by Indemics in this architecture.

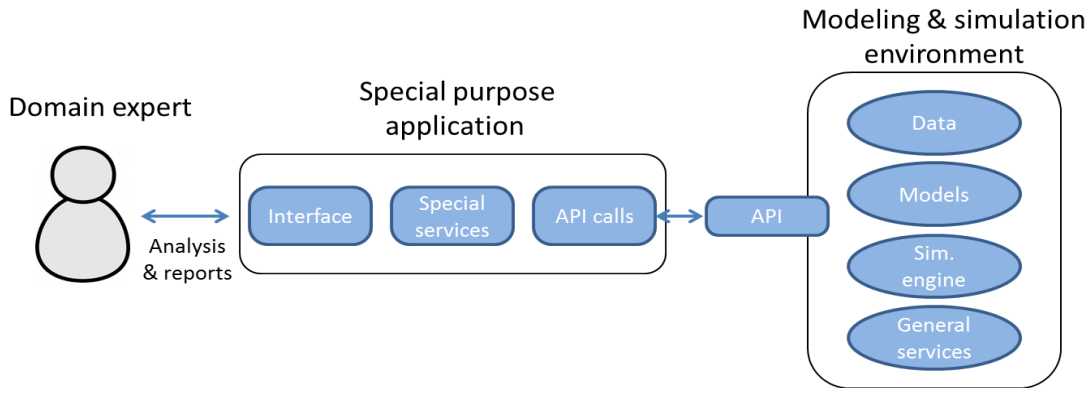

Figure 4.7: An architecture of Indemics with APIs. In this architecture, Indemics serves as a core simulation engine providing general services and managing the simulation related data. A special purpose application provides special services to users by calling Indemics APIs for general services and simulation data.

4.3.2 Objective

Before a discussion on the object of Indemics API design, we need to understand what generic functionality we plan to provide through these interfaces. Indemics has the following key assets that are valuable for special purpose applications:

- Demographic data about synthetic populations and synthetic social contact networks could be useful for special purpose applications.
- Disease propagation models provided by Indemics could be interesting for epidemiologists and public health experts. These experts may first learn the disease propagation models and adjust these models before designing and performing their studies.
- The available intervention models provided by Indemics could be reused in different special purpose applications.
- Obviously, epidemic propagation and its intervention simulation is the core service of Indemics.
- The real-time situation assessment service could be useful in many applications, such as epidemiology-related educational games and applications for the public to learn epidemic propagation patterns and its control strategies.
- The service of setting up simulation experiments and performing experiments is also necessary, since the application developers usually are not familiar with them.

The objective of our Indemics API design is to expose the aforementioned functionality to special purpose application developers in a generic, easy-to-use, and easy-to-understand way.

Meanwhile, the API design also needs to consider computational efficiency and security in its implementation. We emphasize generality in the API design for two reasons. The first is that predicting desired special purpose functions by Indemics designers is difficult. The second is that APIs with many special details could be misleading and misused for special applications.

4.3.3 Design

The API design introduced in this section is the minimum amount of functionality that Indemics should provide. Its design is targeted to a special application which provides our epidemic propagation simulation results, our situation assessment service, and our strategy analysis service to the public through online social media. Additional functions will be added afterward when more applications are extended from Indemics.

As mentioned above, we expect to expose the available information in Indemics to special purpose applications through its APIs. Although any information in relational format can be added to Indemics when necessary, here we delimit the scope of information that applications can access through Indemics APIs. With such a scope of the available information, we can explicitly define and explain the functionality of each API call. Table 4.4 shows the information accessible by Indemics APIs, which we believe to be sufficient for a large set of applications.

Table 4.4: Available data accessible from Indemics APIs.

Data	**Attributes**
Demographics	person's ID, age, gender, health state
Social contact network	first person's ID, activity, second person's ID, activity, the contact duration
Geography	person's ID, home coordination, zip code, block, tract, county

It is common in API design for applications to exchange data through special data types. To support current Indemics APIs, we defined a set of these special data types, which are introduced in Table 4.5.

Most data types in Table 4.5 are self-explanatory except the subpopulation filter. The subpopulation filter is a specially designed tree capable of expressing a wide scope of subpopulation criteria. Figure 4.8 illustrates an example. A leaf of the filter tree could be an attribute filter, a contact network filter, or a geographic filter. The leaf has an attribute name (such as age attribute), a filter value (such as school-age), and an operator ($>, <, ==, != $, ...). The internal nodes are logic operators, conjunctions or disjunctions. The logic operator combines the criterions expressed by its subtrees. The primary motivation of designing the subpopulation filter as such a tree structure is to provide application developers with a means of describing subpopulations criteria. Subpopulation specification is an important

operation in situation assessment and intervention strategy evaluation. It is expected to be flexible and capable in describing subpopulations. Loosely speaking, the tree structure for the subpopulation specification introduced here has a similar expression capability to propositional calculus.

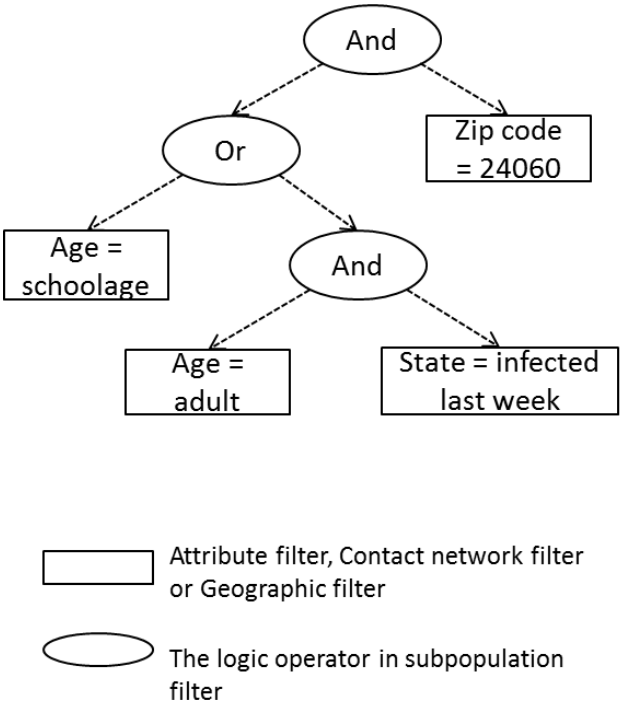

Figure 4.8: An example of the subpopulation filter tree. The leaves of this filter tree are Attribute filter, Contact network filter, or Geographic filter. The internal nodes are logic operators, conjunctions or disjunctions. This subpopulation filter describes a subpopulation of school-age children and infected adults, last week, living in zip code 24060.

With the introduction to accessible data in Indemics and specially designed data types, we now present and discuss the API calls exposed by Indemics. Table 4.6 shows the prototypes of the API calls.

As we discussed before, the object of the API design is to expose the functionality of Indemics in an easy-to-use and easy-to-understand way. We will introduce the functions of this set of API calls and discuss how we approach the object.

- getIntervention() returns available intervention actions such as vaccination campaigns and closing schools. These intervention actions are predefined and understandable by Indemics.
- getDiseaseModels() returns available disease models such as moderate flu and catastrophic flu. These disease models are also predefined.

Table 4.5: Data types used by Indemics APIs and their descriptions.

Name	Description
Plot	A sequence of lines with labels. Epicurve is an instance of this plot.
Map	A collection of coordination points with numbers. The numbers of the coordination points could be marked on a map.
Subpopulation	a list of the person IDs in a subpopulation.
Subpopulation metadata	Statistics about a subpopulation: subpopulation size, cases in different health states, etc.
Intervention action	An intervention action with the following parameters: action type, description, action duration, action delay.
Session	An instance of running simulation. A session has a session ID, region ID, and a description.
Application	An object representing the application calling Indemics APIs. All the resources consumed by this application will be registered in this object.
Attribute filter	A filter on one individual attribute. It filters out the persons not matching the value on the individual attribute. A filter needs to specify its attribute name, operation, and filter value.
Contact network filter	A filter on one contact network attribute. It filters out the persons not matching the filter. Its format is the same as the Attribute filter.
Geographic filter	A filter on one geographic attribute. It filters out the persons not matching the filter. Its format is the same as Attribute filter.
Subpopulation filter	A tree with the Attribute filters, the Contact network filters, the Geographic filters, and logical operators (and/or). It filters out the persons not matching the criterion defined by this subpopulation filter.

- getAttributes() returns the attributes of accessible data and their suggested filter values. Examples of demographic attributes are age, gender, and health state. Their suggested filter values could be school-age, male, and infected in last week, respectively.

These three calls are designed to expose the accessible data in Indemics. Applications can call the following situation assessment services after becoming familiar with the accessible information.

- getPlot() returns an epi-curve showing the temporal propagation trend in the given region from the startTime to the endTime.

Table 4.6: Functions provided by Indemics APIs.

Name	Parameters	Return Data Type
getInterventionActions	-	intervention actions
getDiseaseModels	-	disease models
getAttributes	-	individual attribute names
getPlot	region, startTime, endTime, subpopulation filter, sid, intervention	plot
getMap	region, subpopulation filter, sid, intervention	map
getSubpopulation	region, subpopulation filter	subpopulation
getIndividual Contact	region, target subpopulation filter, contact subpopulation filter	subpopulation
getAttributesByFilter	region, subpopulation filter, sid	<person ID, attribute values>
getAttributesByList	region, subpopulation filter, sid	<person ID, attribute values>
createSimulationSession	region, disease model	sid
intervene	sid, intervention action, Subpopulation filter, startTime, length	boolean

- getMap() returns a map showing the spatial distribution of infected individuals in the given region.
- getSubpopulation() returns a list of person IDs of a subpopulation matching the filter criterion.
- getIndividualContact() returns the social contacts of the individuals specified by the targetSubpopFilter; the social contacts will be filtered by the contactSubpopFilter.
- getAttributesByFilter() returns all attribute values of the subpopulation specified by the subpopulation filter. For example, this might include the demographic attributes of infected individuals.
- getAttributesByFilter() returns all attribute values of a group of people. The results of calling the first method are equivalent to calling getSubpopulation() followed by getAttributesByList().

This set of calls exposes the Indemics functionality of situation assessment and subpopulation investigation. The getPlot() and getMap() provide two measures of assessing the infection

situation from a temporal viewpoint and a spatial viewpoint, respectively. For example, the epicurve returned by getPlot() shows the temporal disease propagation pattern. The rest of the calls are generic queries about the population, their social contact network, and the infection situation.

In addition to the infection situation assessment services, simulating epidemic propagations with interventions is also a core functionality that Indemics provides. The following are a set of callings that allow applications to create simulation sessions for their special purposes and run simulations. Interactive applications, such as educational games for school students to learn epidemic propagations and control strategies can also be designed through these calls.

- createSimulationSession() creates an epidemic propagation simulation session using the specified disease model and population in the region.
- getSimulation Session() returns a set of available simulation sessions in Indemics. Applications can investigate the infection situation and intervene with these simulation sessions.
- intervene() assigns an intervention to the targeted simulation session (sid).
- SimulationContinue() instructs the targeted simulation session to continue its simulation.

4.3.4 Discussions of the API Design

We learned how to improve the Indemics API design from our hands-on experience. Modifying each previous design and adding new functionality were inevitable steps when attempting to facilitate the creation of extended applications from Indemics. The design outlined in this section is our initial effort, and we indeed learned valuable lessons from the initial work. Here are some of our lessons:

- Although a set of data types and their formats are proposed in our API design, the applications built on the top of Indemics may express the information in its own format, not using our data types. For example, the application we have integrated with Indemics might store the information expected by Indemics using the XML format and JSON format, two network data exchange formats. Although the data in JSON or XML contains the exact information expected by Indemics, they are in a different format from what Indemics expects. A translator between Indemics APIs and the extended application has been added. We think such a translator will be necessary, because the data formats in the future applications could be different from the format Indemics provides.

- Another lesson learned is balancing API computational efficiency with its service quality. Our first implementation version of the getMap() calculated the spatial distribution of infected individuals in a very low resolution. As a result, a very large set of spatial units with statistics on infected individuals were returned. The extended application and the Indemics computer system were deployed remotely, and this large data set caused serious communication latency. Although spatial distribution in a low resolution means better service quality, it can cause efficiency issues. This challenge of balancing between different expected features/efficiency could repeat in future applications.

Although we have proposed our initial API design here, there still exist many challenges in its implementation. We at minimum must resolve these issues: synchronization in multiple concurrent API calls, traffic balancing in Indemics while serving multiple concurrent API calls, data security, and prevention from a malicious attack. All of these concerns are challenging and not yet completely resolved.

4.4 Chapter Summary

In this chapter, we presented an epidemic simulation framework—Indemics—and introduced their primary modules in this framework. We introduced and discussed several important features of Indemics, such as the computational synchronization and the multiple concurrent simulation sessions. We also presented the database design and API design.

Indemics will be used to evaluate the proposed approaches of system decoupling and simulating interventions using relational databases. In the following chapters, we will present a series of experiments using Indemics to verify our research hypotheses.

Chapter 5

Efficiency of Performing Intervention Studies

High efficiency in performing simulation experiments is an expected feature of Indemics as a tool to support the studies of epidemic intervention strategies. The efficiency of supporting real-time epidemic intervention studies depends on the human productivity of preparing studies and the computational efficiency of performing simulation experiments. In this chapter we investigate the human productivity of using Indemics in preparing intervention studies.

5.1 Efficiency of Intervention Implementation

We have two ways to simulate epidemic interventions: within the high-performance epidemic simulation engine or using external database management systems to compute interventions and apply them to the epidemic simulation engine. In what follows, we use EpiFast [14] to represent the former and the Indemics framework [15] to represent the latter. Also, we assume that the simulation engine used in the Indemics framework for diffusion is EpiFast. The main factor for users to choose one execution method over the other should be the ability to complete end to end implementation and obtain simulation results within the time constraint of a case study. This puts a priority on short development time supported by efficient implementation, as well as reduced running time of intervention simulations. In this section, we first introduce epidemic interventions and their implementation methodologies, and then discuss their efficiency and corresponding performance.

Epidemic interventions. A typical epidemic simulation consists of two parts: co-evolving disease diffusion and human intervention. Correspondingly, an epidemic simulation system usually includes diffusion code and intervention code. E.g.,

```
EpiFast = diffusion code (C++) + intervention code (C++)
```

```
Indemics = EpiFast diffusion code (C++) + framework code (Java)
 + DBMS + intervention script (CNL)
```

where CNL means a controlled natural language.

Computation of most interventions requires both dynamic data generated within the diffusion computation (e.g., current diagnosed cases) and data not directly relevant to disease diffusion (e.g., household income). Some examples of interventions include:

- **Triggered intervention**: In this intervention type, a threshold is evaluated every day. When it is exceeded, then predefined actions are taken, e.g., when the percentage of diagnosed school-age children exceeds 20%, close all schools.
- **Targeted intervention**: In this type, intervention action is applied to people in a certain health state, e.g., administer an antiviral to diagnosed school-age children.
- **Local triggered intervention**. In this intervention type, many subpopulations are predefined and when any subpopulation incurs a local outbreak the intervention action is applied to the whole subpopulation, e.g., close any school where more than 5% of students show symptoms (called "School intervention"); vaccinate all people in any census block group if more than 1% of that block group are diagnosed (called "Block intervention".

Table 5.1: Development cost of implementing interventions in EpiFast vs. Indemics scripts

intervention	EpiFast implementation		Indemics implementation	
	lines of code	effort	lines of script	effort
school/block	500	8 weeks	40	8 hours
triggered	150	1 week	20	1 hour
targeted	600	12 weeks	20	1 hour

Real world case studies for ongoing epidemics often come with a tight deadline on the delivery of study results. From our experiences with various federal agencies, we were often requested to complete a study within a few weeks, which included running a whole set of simulations for several rounds, each round having a time budget of only a few hours.

In Table 5.1 we present our development experiences in implementing several interventions in EpiFast vs. in Indemics. Clearly Indemics reduces development time by at least one order of magnitude. This can shorten the overall study life cycle significantly, and also largely improves the productivity of the simulation tool users by allowing them to explore larger classes of intervention strategies.

5.2 Performance Modeling and Prediction

5.2.1 Performance Modeling Methodology

The experiment time of Indemics for real time epidemic planning is critical and should be estimated during experiment design period. The experiment execution period could be a bottleneck for real time study if the intervention computation is extremely complicated. In this section, a methodology of performance modeling and prediction for Indemics will be introduced. Performance prediction in experiment design could avoid performance issues and assure experiment completion on deadline using Indemics for real time epidemic strategy studies.

During an intervention case study, public health experts first design intervention strategies and give them to the experiment designers, the intervention scenarios are first translated into database queries using Indemics script language. The following is a script example using Indemics script language for the School intervention study. SQL statements that implement this intervention are presented.

```
Initialization;

Define School_Trigger as SCHOOL_DIAGNOSED_TOTALLY.persons
    > 0.05* SCHOOL_INFO.school_children
Reset table SCHOOL_INTERVENED intervened_day = -1 :
Update SCHOOL_INTERVENED set intervened_day = -1;
For simulation day Day from 1 to 300
    1. Count today new diagnosed children in each school,
    save to SCHOOL_DIAGNOSED_TODAY :
    Manipulate database:insert into SCHOOL_DIAGNOSED_TODAY
    select school, count(pid), Day from Person_School, Diagnosed_person
    where pid = diagnosed_pid and diagnosed_time = Day;

    2. Count diagnosed children in each school in the infectious
    state, save to SCHOOL_DIAGNOSED_TOTALLY :
    Manipulate database:insert into SCHOOL_DIAGNOSE_TOTALLY select
    school, sum(persons), Day from SCHOOL_DIAGNOSED_TODAY where
    diagnosed_time between Day and Day - 5 group by school;

    3. Set SCHOOL_INTERVENED intervened_day = Day
    if schools in SCHOOL_INTERVENED
    been intervened and School_Trigger is fired:
    Manipulate database: update SCHOOL_INTERVENED set  intervened_day =
    Day where SCHOOL_INFO.school = SCHOOL_DIAGNOSED_TOTALLY.school
```

Table 5.2: Atomic statement examples in Indemics

Atom	**Symbol**	**Algebra**	**SQL Example**
Projection	π	$\pi_\alpha R$	select pid from persons
Predicate	θ	$\alpha = \beta$	age = 18
Selection	σ	$\sigma_\theta R$	select from persons where age = 10
Nature Join	$\bowtie$	$P \bowtie I$	select from persons P, infections I where P.pid = I.transmitee
Insert	ϕ	$\phi_S R$	Insert into S from R
Update	μ	$\mu_{\alpha=x} R$	update R set $\alpha = x$
Group By	Σ	$\Sigma_\alpha R$	group by α
Index	Ψ	$\Psi_\alpha R$	create index on $R(\alpha)$

Table 5.3: SQL statement examples in Indemics

Statement	**Algebra**
Insert infection cases R into table $INFECTION$	$\phi_{INFECTION} R$
Update status to x for schools in R	$\mu_{status=x} R$
Search household with infections	$\pi_{hid}(HOUSEHOLD \bowtie_{pid} INFECTION)$
Count new infection cases in each block	$\pi_{block,count(pid)} \Sigma_{block} BLOCK \bowtie_{pid} INFECTION$

```
and diagnosed_time = Day and SCHOOL_DIAGNOSED_TOTALLY.persons
> 0.05* SCHOOL_INFO.school_children;

4. Apply vaccination treatments to the schools in SCHOOL_INTERVENE
with intervened_day = Day:
Apply interventions:Anti-viral, 100% efficacy, population:
select pid from Person_School, SCHOOL_INTERVENED where
Person_School.school = SCHOOL_INFO.school and intervened_day = Day;
```

As illustrated in the given script example, the query statement specifies how to manipulate dynamic data and select intervened subpopulations from the database. SQL statements in Indemics script language consist of SQL atomic statements. A legal SQL statement is either solely an atomic statement or a composition of atomic statements. Table 5.2 lists SQL statement atoms supported in Indemics script language with their symbols. To model the performance of a given intervention scenario, the scenario is first translated into database queries and then each statement is decomposed into query atoms. The performance modeling is based on these query atoms. Typical query statements and their atoms are given in Table 5.3.

The intervened subpopulation selection during intervention simulation is data-intensive, and running time for manipulating and selecting data from the database is a considerable part of the entire intervention study. It is essential for Indemics to model its performance in

intervention simulation and use this performance model to predict the running time in the study before actually performing it. It is important, because case study designers usually request results within a given period. By using the performance model, designers can estimate running time directly from a given intervention scenario.

We model Indemics performance based on query statement atoms. Given the script in Indemics language for a case study, we decompose query statements in the script into atoms, and estimate configurations for these atoms. The atom configuration includes size of queried table, size of query results, and other parameters. In order to predict query time, we profile the performance of query atoms in the database employed by Indemics in advance. This database profile forms a performance lookup library which collects the performance of each statement atom under different configurations in the database system. The performance of query atoms can be referred to from this performance lookup library.

Definition 1 *We denote the predicted running time of a query atom* α $AP(\alpha)$*, the value of* $AP(\alpha)$ *under diverse configurations could refer to the performance lookup library.*

Since each query statement consists of query atoms, the predicted running time of a given query could be calculated by summing up the running time of its atomic queries if the database executes its atomic queries sequentially.

Definition 2 *The predicted running time* $QP(Q)$ *of a query statement* Q *is given by:*

$QP(Q) = AP(a_1) + AP(a_2) + ... + AP(a_n)$

where Q *can be decomposed into* $AP(a_1), AP(a_2), ..., AP(a_n)$.

A script in Indemics language specifies database queries on each simulation day, therefore, the total running time of a script is predicted by summing up the predicted running time of all query statements on each day.

Definition 3 *The predicted running time* $SP(S)$ *of a script* S *is calculated by:*

$PP(S) = \Sigma_{day=0}^{lastday} QF(Q_1^{day}) + QF(Q_2^{day}) + ... + QF(Q_i^{day})$.

where day is simulation day, and queries $Q_1^{day}, Q_2^{day}, ..., Q_i^{day}$ *model the intervention scenario.*

In order to predict intervention simulation time, we refer to the performance lookup library and then calculate SP.

5.2.2 Predicting Simulation Performance

In this section we introduce our performance prediction method for Indemics. This prediction method is based on the performance model outlined in Section 5.2. Intervention simulation time is predicted by analyzing the query atoms in an intervention script. School intervention is used as an example to demonstrate our prediction method. Experimental verification of this prediction method is presented at the end of this section.

Performance Profiling

We examined our database system, collected the performance profiles of each query atom under typical query configurations, and organized the performance profiles as a performance lookup library. The query atoms in this library are the relational query atoms listed in Table 5.2. This library maintains the performance profiles of all query atoms, but, limited by space, in Table 5.4 we only list here a representative segment of this library to demonstrate its performance profiles.

Table 5.4: A representative segment of the performance lookup library.

Statement	**Algebra**	**Configuration**	**Time (seconds)**
Insert	$\phi_S R$	S: 1K	0.005
Insert	$\phi_S R$	S: 100K	0.1
Update	$\mu_{S.\alpha=x} R$	R: 1K	0.002
Update	$\mu_{S.\alpha=x} R$	R: 5K	0.01
Select	$\sigma_\theta R$	R: 10K	0.001
Select	$\sigma_\theta \Psi_\alpha R$	$\sigma_\theta \Psi_\alpha R$: 2M	0.02
Nature join	$P \bowtie I$	P: 0.5M, I: 1K	0.06
Nature join	$P \bowtie I$	P: 0.5M, I: 100K	0.5
Group by	$\Sigma_\alpha R$	R: 10K	0.3

To predict intervention simulation time, it is estimated by analyzing the Indemics script s modeling an intervention and computing $SP(s)$ introduced by Definition 3. In this section, we demonstrate how to analyze Indemics script and calculate SP using a concrete intervention. Along with this example, the predicted running time of simulating several interventions is presented, and it was compared with the actual experiment time to validate our prediction methodology.

Scenarios of Case Studies

In this section, School intervention is used as an example to demonstrate our performance prediction. The scenario of School intervention is that any school where the percentage of

Table 5.5: Table statistics for performance prediction experiments

Table	Region	Size
Child_School	Chicago	2.2M
Child_School	Boston	0.9M
Child_School	Seattle	0.7M
Child_School	Miami	0.5M
School_Info	Chicago	8K
School_Info	Boston	6K
School_Info	Seattle	3K
School_Info	Miami	2K
Person_Block	Chicago	9M
Person_Block	Miami	2M

children diagnosed with flu crosses a certain threshold (such as 5% of enrolled children), then those children in the school will be given a vaccination treatment. This School intervention will be enforced from the first day when the flu propagation begins to the 300th day, when the epidemic vanishes. We first modeled this School intervention by relational algebra. The formulae in relational algebra given by Algorithm 5 model School intervention. Then the formulae are translated into an Indemics script—illustrated as the script example in Section 7.1—to simulate School intervention under the Indemics framework

Algorithm 5: School intervention in relational algebra

Initialization;

for simulation day α from 1 to 300 **do**

1. Count diagnosed children today in each school, save the results in SCHOOL_DIAGNOSED_TODAY;

$\phi_{SCHOOL\_DIAGNOSED\_TODAY}\pi_{school,dignosed\_children=count(pid),today}$
$\Sigma_{school}\Psi_{pid}CHILD\_SCHOOL \bowtie_{pid} \sigma_{today}DIAGNOSED$;

2. Count diagnosed children in each school still sick, save the results in SCHOOL_DIAGNOSED_TOTALLY:

$\phi_{SCHOOL\_DIAGNOSED\_TOTALLY}\pi_{school,total=sum(dignosed\_children),day}$
$\Sigma_{school}\sigma_{\alpha-6<day<=\alpha}(SCHOOL\_DIAGNOSED\_TODAY)$;

3. Set SCHOOL_INTERVENED intervened_day = α if this school has not been intervened and the number of diagnosed children has crossed the threshold :

$\mu_{intervened\_day=\alpha}\sigma_{persons>0.05*school\_children}(SCHOOL\_INFO \bowtie_{school}$
$\sigma_{today}SCHOOL\_DIAGNOSED\_TODAY$;

4. Apply vaccination treatments to the target schools:

$\pi_{pid}(\sigma_{intervened\_day=\alpha}SCHOOL\_INTERVENED \bowtie_{school} CHILD\_SCHOOL)$;

end for

Analysis of Query Atoms

Now we present a performance analysis of the queries simulating School intervention in the Miami population. As illustrated in Algorithm 5, the Indemics script queries the database in the following four steps: counting new diagnosed children today in each school, counting the sick children in each school, finding schools that have crossed the intervention threshold, and finally applying medical treatments to those targeted schools. These steps are presented as the formulae in Algorithm 5. The first formula (counting new diagnosed children today in each school) consists of these query atoms: $\sigma_{today}DIAGNOSED$, $\Psi_{pid}CHILD\_SCHOOL \bowtie_{pid} \sigma_{today}DIAGNOSED$, Σ_{school}, π, and ϕ. The estimated average number of daily diagnosed cases is less than 1000 in our disease model, so Ψ_{pid} $CHILD\_SCHOOL$, and $\sigma_{today}DIAGNOSED$ only have thousands of rows (Table 5.5). We concluded by referring to the performance lookup library that the bottle neck in these query atoms is the join query $\Psi_{pid}CHILD\_SCHOOL \bowtie_{pid} \sigma_{today}DIAGNOSED$. By referring to the performance lookup library (Table 5.4), we evaluated the average running time of this join atom (counting new diagnosed children today in each school) would be 0.06 seconds in this School intervention simulation. The formula for the second step consists of these atoms: $\sigma_{\alpha-6<day<=\alpha}(SCHOOL\_DIAGNOSED\_TODAY)$, Σ, π and ϕ. Since the estimated average number of daily diagnosed cases is less than 1000, the size of table $\sigma_{\alpha-6<day<=\alpha}(SCHOOL\_DIAGNOSED\_TODAY)$ is less than 1000. By referring to the performance lookup library, we concluded that the running time of the second formula can be ignored compared with that of the first formula. We analyzed the running time of the third and last formulae using the same method, and we concluded the running time of the third formula can be ignored compared with that of the others.

Calculation of Predicted Performance

After the query analysis, we calculate SP by summing up the predicted running time of all the queries. The atoms with insignificant costs can be ignored in our performance prediction. As a result, we only count the performance dominant atoms in SP. Q_1, Q_2, Q_3, and Q_4 denote the four queries, and $QF(Q_2)$ and $QF(Q_3)$ can be ignored in calculating SP. Eventually, we calculate SP as $\Sigma_{day=0}^{300}(QF(Q_1) + QF(Q_4)) = 300 * 2 * 0.06 = 36$.

The performance predictions for the regions other than Miami also follow the same method. From analyzing the School intervention script, we know the performance bottle neck is the join query involving with $CHILD\_SCHOOL$ table. The size of $CHILD\_SCHOOL$ table for Seattle is very close to that for Miami, therefore the estimated running time of Seattle simulation is 36 seconds (the running time of the Miami simulation). The school-age subpopulation in Boston is approximately twice larger than that in Miami, therefore the estimated running time of the Boston simulation is 72 seconds.

To evaluate this performance prediction method, actual running time and predicted running

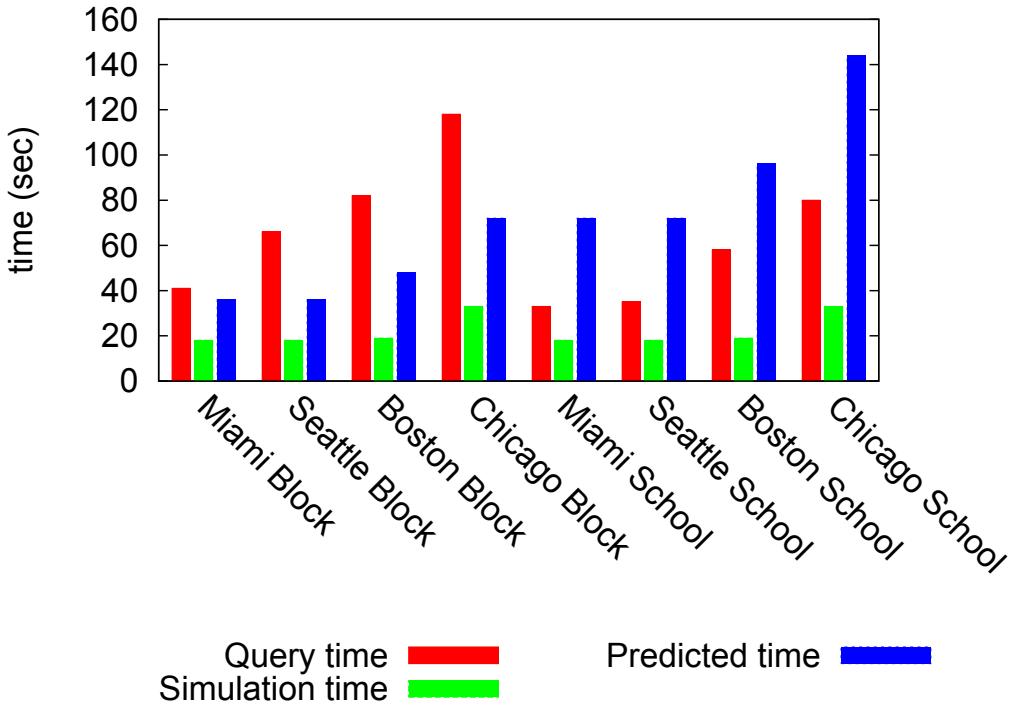

Figure 5.1: The predicated database query time and actual query time. Query time is actual query time using by the database of Indemics; simulation time is the actual propagation simulation time; the predicted query time is our predicted query time.

time is compared in Figure 5.1. Although the predicted time is not so accurate, our prediction still is useful for study planning. It is typical that each intervention strategy will be simulated under tens of different disease models for tens of iterations, and the total experiment period could be hours or weeks. By this performance prediction, experiment period can be estimated approximately in advance. Figure 5.2 shows the estimated development days, experiment days, and total study periods for intervention studies, which must be repeated for 50 iterations under 20 different disease models in four regions. This figure shows us that Indemics can reduce case study period.

5.3 Chapter Summary

One of the research problems we focus on in this dissertation is the efficiency of preparing studies for real-time intervention studies. Therefore, in this chapter, we studied and compared the intervention implementation efficiency of using Indemics versus previous simulation systems. Implementation efficiency has been improved using Indemics, primarily because interventions are modeled as a set of database queries. In addition, we also introduced our database performance modeling and database performance predication methodology.

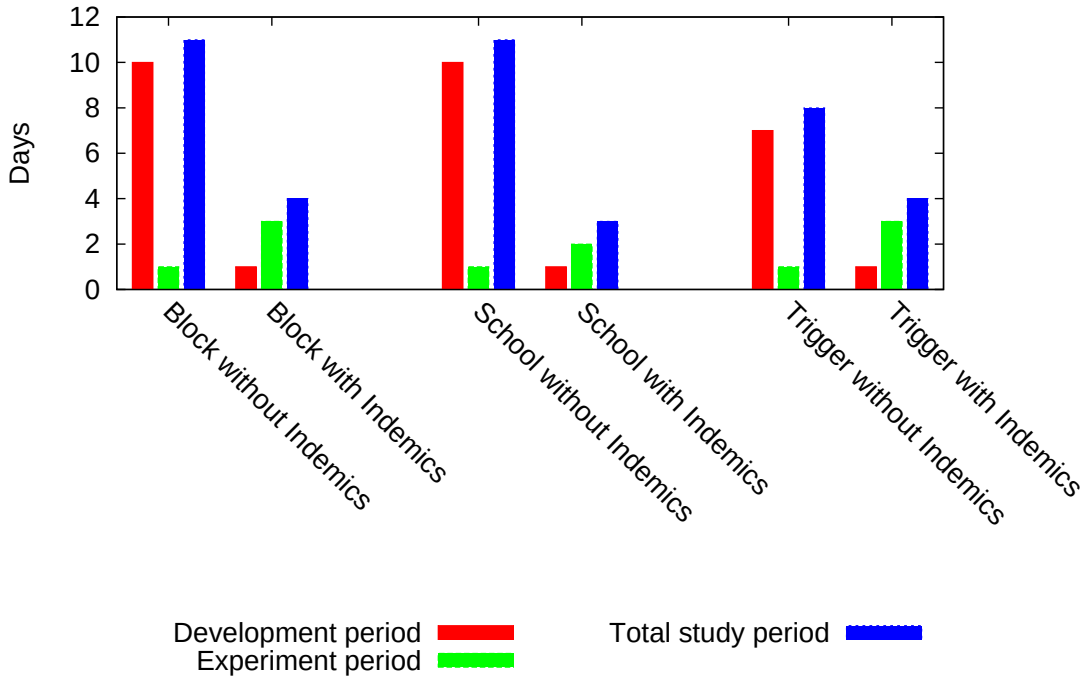

Figure 5.2: Comparisons of development days, experiment days and whole study periods between using EpiFast only and using Indemics. The experiment days of Block and School interventions without using Indemics are estimated as 1 day, since they are difficult to implement without Indemics.

Chapter 6

Computational Efficiency

The computational efficiency of this database supported epidemic modeling environment is one of our research problems. This chapter focuses on studying the computational efficiency of this modeling environment in simulating epidemic propagations and interventions. This environment is distinct from previous modeling environments in its loosely coupled and database supported architecture. Thus, we will show the performance overhead caused by this new architecture, the primary computational costs of this environment, and discuss system performance optimizations.

6.1 Performance Study of Indemics

In this section, we present a series of detailed performance studies on our proposed simulation framework, Indemics.

6.1.1 Experiment Set-up

We first introduce the experiment settings of the computing environment and the interventions selected for these performance studies before presenting the detailed performance results and our analysis.

Computer System

In our experiments, we use EpiFast [14] as the propagation simulation engine and run all simulation experiments on a 60-node Linux cluster. Each compute node has two Intel Xeon X5670 processors running at 2.93GHz and 48 GB memory (4 GB/core). The Indemics server runs on the head node of the cluster. The database systems chosen for the experiment are

Table 6.1: A computation resource comparison between Oracle 11g and Greenplum (GP)

Hardware	**Oracle**	**Master server of GP**	**Segment server of GP**
CPU	Intel Xeon 2.4 GHz	X5680 3.33 GHz	Intel 5670 2.93 GHz
Number of cores	4 /CPU * 4 CPUs/server * 1 server	6 /CPU * 2 CPUs/server * 2 servers	6 /CPU * 2 CPUs/server * 2 servers
Memory (GB)	64 total	48 GB/CPU, 192 total	48 GB/CPU, 192 total
DBMS	Oracle 11g enterprise	Linux	PostgreSQL 8.2.15
Data replication	Data Guard disabled, no replication	N/A	2 replications

Oracle 11g and Greenplum. The computation resources and the configurations of these two database systems are given in Table 6.1. The computer cluster used to run the simulations and the two database systems are located within a computing center.

We compared the performance of our Oracle 11g database system and Greenplum database system in the Indemics framework based on the following motivations. In the early stages of the Indemics project, the data management characteristics and database system requirements were not clear. We chose an Oracle database because it is a general purpose database system. After the Indemics framework was established, we employed Indemics for several real intervention case studies. In these case studies, many intervention scenarios are modeled as complex database operations. The scripts of the intervention scenarios may have complex database operations such as multiple-table-join, large numbers of updates, and queries over large tables. Such complex queries caused a performance bottleneck in previous studies. Thus, we explored the possibilities of using the Greenplum database, a massively parallel processing database system specialized for analysis on large scale datasets.

From an architecture perspective, our Oracle database is deployed on a shared-memory computer system, and our Greenplum system is deployed on a shared-nothing computer system. The scalability of Indemics with population size is a significant issue. We are studying the simulation of intervention strategies in national or regional populations. Databases based on shared-memory computer architecture usually has scalability limitations, but databases based on shared-nothing architecture typically can scale up to hundreds of processors [22]. Thus, we presume databases based on a shared-nothing architecture, such as Greenplum, can scale better in Indemics.

Table 6.2: The statistics of three U.S. metropolitan areas used in the Block intervention and Distance-1 intervention simulations.

Data	Region	Population size	Number of rows	Size of raw data file
Network	Miami	2.1M	105M	2.7GB
	Chicago	9M	538M	16.7GB
	Los Angeles	16.2M	919M	23.4GB
Block	Miami	2.1M	2.1M	33MB
	Chicago	9M	9M	154MB
	Los Angeles	16.2M	16.2M	271MB

Table 6.3: The statistics of the areas in the Household intervention simulations.

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

Simulation Problems

The populations we selected for the performance studies include several U.S. areas. The reason we selected these areas is their typical population sizes and contact network structures. The statistics of these populations are given in Table 6.2 and 6.3. Due to their different population sizes, we used different numbers of computer nodes to run EpiFast for their epidemic propagation simulations. We ran the Chicago simulations using 5 compute nodes and 8 cores per node, the Los Angeles simulations using 8 compute nodes and 8 cores per node, and others using 4 compute nodes and 4 cores per node. We did not increase computation resources in the database system during the intervention simulations.

The intervention scenarios we selected include Household intervention, Targeted intervention, Block intervention, Distance-1 (D1) intervention, and the Honest behavior in the sick-leave intervention (For the rest of this chapter, we will call it simply "Honest".). Their scenarios are given in Table 6.4. These interventions are selected from real studies in which the interventions were simulated by Indemics [44, 43].

The reason we selected these interventions is that our scripts which model these intervention scenarios contain different types of database operations. The scripts for Block intervention

Table 6.4: The selected intervention scenarios for the performance study.

Intervention	**Scenario**
Household	All members of households with one or more diagnosed household members take social distancing actions.
Targeted	Treat people with a certain health state immediately, e.g., administer antiviral drugs to sick school-age children when they are diagnosed.
Block	Provide vaccines to all the people residing in a census block group if an outbreak is observed in that block group.
Distance-1	Self-motivated individuals take protective action (seek vaccine) when a percentage of their direct contacts who are diagnosed exceeds a threshold, say 3% of their total contact neighbors.
Honest behavior	Honest workers leave their work and stay at home from their diagnosed day to their recovery day.

Table 6.5: Query features of the studied intervention scenarios. N denotes the population size. "Block" denotes Block intervention. "Honest" denotes the Sick-leave-honest intervention. "D1" denotes the Distance-1 intervention.

Feature	**Block**	**Honest**	**D-1**
Temporary table writing	Yes	No	Yes
Size of table	$2 * N$	$30\% * N$	$50 * N$
Query types	Mixing join, insert, group by, and update	3-table-join	Mixing join, insert, group by, and update

and D-1 interventions have join, insert, group by and update database operations. Both of these intervention simulations join two database tables—one dynamic table about individual health states and one static table about auxiliary information for the population. However, the tables queried in the D1 intervention simulations are 25 times larger than the Block intervention simulations. The Honest intervention simulation represents a different query pattern. The script of Honest intervention has no temporary table writing (no table update, insert, or delete), but the script has several 3-table-join operations. Two tables are dynamic health state tables and the other is a static table containing worker information. The 3-table-join operations make this intervention simulation complex and interesting for a performance study. These three tables joined in Honest intervention are smaller than the tables in Block intervention and D-1 intervention. A comparison between the database scripts of these three interventions is provided in Table 6.5.

6.2 Performance Overhead

In this section, we choose two interventions that are already implemented both in Indemics and EpiFast, and compare the performance of simulating them in the two systems. The performance of EpiFast simulation is viewed as a lower bound, and the purpose of this experiment is to find out the performance overhead of using Indemics. To make an impartial comparison, simulations using EpiFast and Indemics are the same, both for the propagation process and for the intervention process, and produce exactly the same outcome. The only difference is that the intervention is computed using C++ code in the former solution and JAVA plus Oracle in the latter.

Note that although we spent months implementing the two interventions in EpiFast, it only took a few hours to write Indemics scripts for them. In this experiment, however, we are mainly concerned about simulation execution time. For EpiFast simulations we collect the total running time. For Indemics simulations we collect: 1) the execution time of database queries related to intervention simulations, and 2) the execution time of disease propagation simulations in EpiFast.

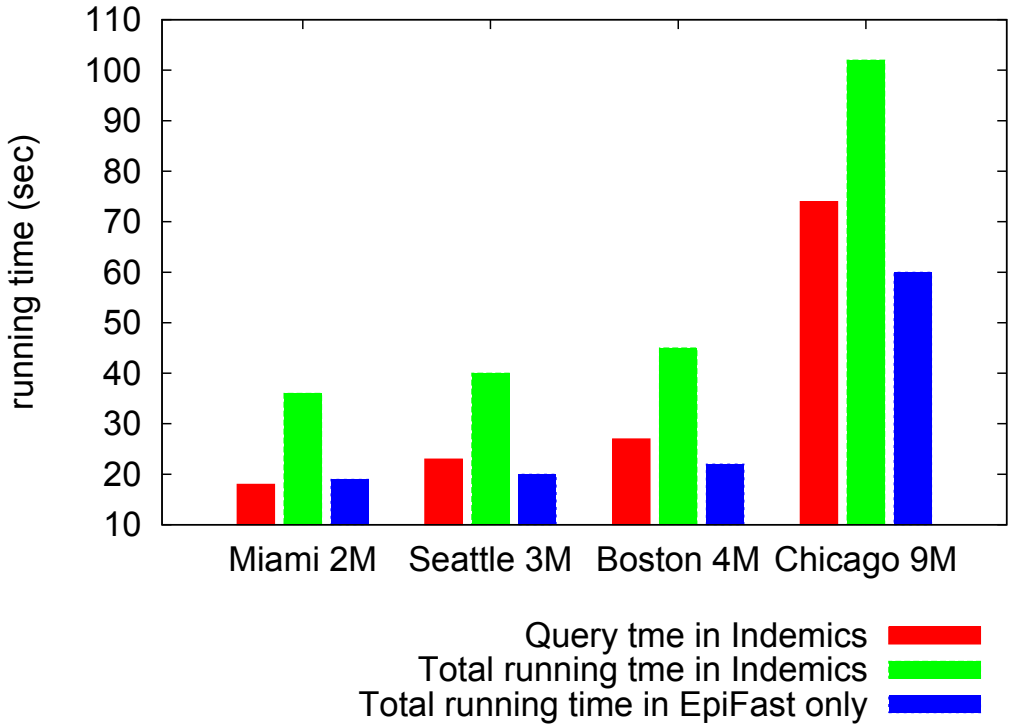

Figure 6.1: The performance of Trigger intervention in Indemics and EpiFast. Trigger intervention was simulated in three U.S. metropolitan areas with different population sizes.

Trigger intervention. In this experiment we are mainly concerned with the performance of threshold evaluations. We plot database query time (for counting diagnosed people)

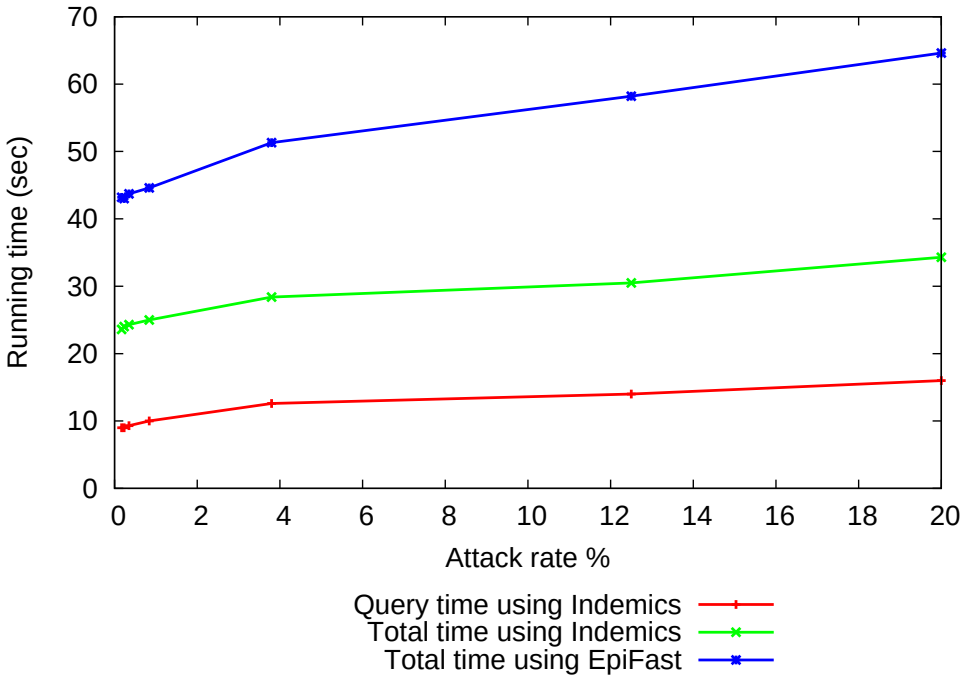

Figure 6.2: The performance of Target intervention in Indemics and EpiFast. Target intervention was simulated in the population of Miami under disease models with different attack rates.

and the total execution time of Indemics simulations along with the total running time of EpiFast simulations in Figure 6.1. We simulate Trigger intervention in four US urban regions: Miami, Seattle, Boston, and Chicago, with increasing population sizes. Figure 6.1 shows that, with Indemics, the intervention simulations incur a reasonable performance loss which easily outweighs the benefits derived from ease of query composition.

Target intervention. Target intervention is simulated in the Miami population under disease models with different infectivities. The moderate disease model has a lower attack rate, where attack rate is the percentage of population infected during an epidemic; the strong disease model has a higher attack rate. A higher attack rate leads to a larger set of people requiring intervention. From Figure 6.2 we observe that Indemics simulations perform better than EpiFast simulations for this target intervention.

The experiment suggests that Indemics may improve performance in some cases and incur a reasonable overhead in others. The performance overhead ratio, i.e. the ratio of total execution time using Indemics simulations to that using EpiFast simulations, is illustrated in Figure 6.3. Indemics incurs about 100% performance overhead for Trigger intervention and reduces the total execution time to half in Target intervention simulations. Even in the case of Trigger intervention, Indemics is still preferable, because the benefits of Indemics derived from the quick re-deployment of interventions, a decrease in the risk of code errors,

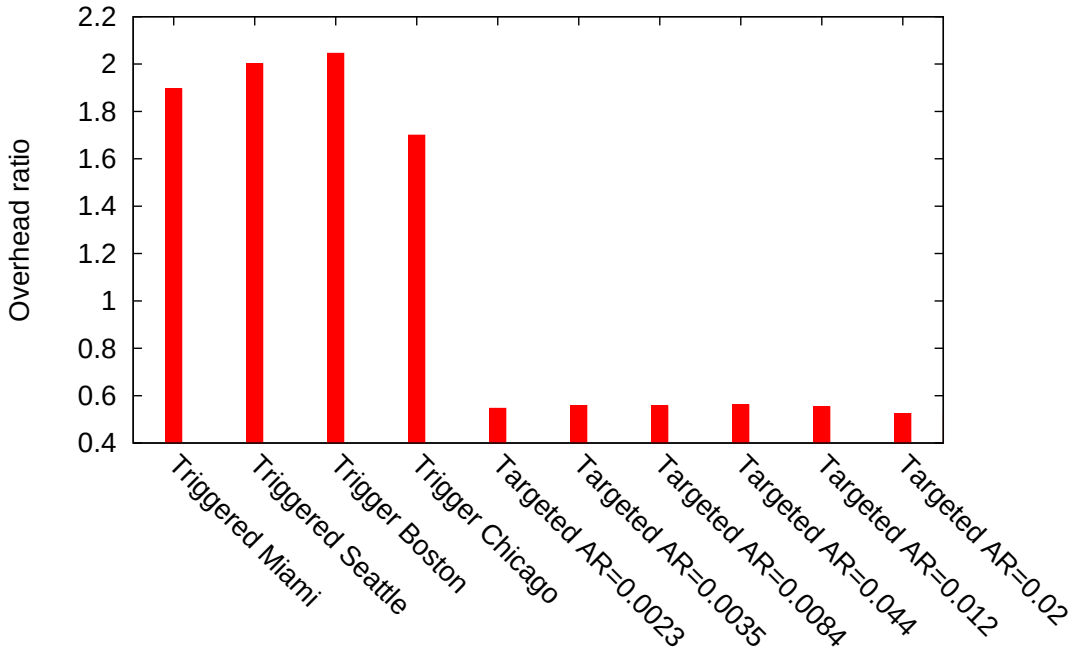

Figure 6.3: The performance overhead ratio in Trigger intervention and Target intervention. (AT: attack rate). The ratios of Trigger interventions are around 2, but the ratios of Target interventions are around 0.5.

and a reduction of human effort which can easily outweigh reasonable performance losses.

6.3 Performance Analysis

This database supported epidemics simulation framework, Indemics, presents a novel simulation architecture. Computational efficiency is one of the key questions about this architecture. In this section we will show the computational efficiency of Indemics in simulating epidemic propagations and interventions. The performance of Indemics in simulation experiments depends on many factors, such as social network structures, disease models, interventions, and compute systems. We will show the correlations between these factors and the computational efficiency of Indemics.

As introduced in Section 4.1, Indemics is a distributed and database supported simulation framework. In Indemics, the IEPSE subsystem (currently EpiFast) simulates epidemic propagations within social contact networks, the ISSAE subsystem (currently relational databases) simulates intervention scenarios, and these two subsystems interact with each other at run-time to simulate the co-evolution of epidemic propagation and intervention. Therefore, we consider the following primary computational costs of Indemics in this per-

formance study: the propagation simulation time by EpiFast, the intervention simulation time by relational databases, and the communication time between EpiFast and relational databases. EpiFast is an existing epidemics simulation system. Its performance analysis has already been studied [14]. Thus, we will not study it again in this section, but will focus on the other two costs in detail.

6.3.1 Communication Costs

We first study the communication time (between EpiFast and relational databases) in Indemics. We choose Household intervention introduced in Section 6.1.1. Household intervention is a continuous process by which a set of individuals will be targeted and will take intervention actions each day of the simulation. Thus, we can see the communication time trend during its simulation. Also, the number of intervened individuals during Household intervention simulation is roughly proportional to the number of infected individuals, which allows us to analyze the correlations between the communication time and the disease propagation process.

Since Indemics exchanges data about infected individuals and intervened individuals between EpiFast and relational databases, the communication time between them is affected by the infected subpopulation size and intervened subpopulation size. Both of these subpopulations are related to the epidemic propagation dynamics. Thus, Figure 6.4 illustrates the communication time in simulating Household intervention in the Miami social contact network with different disease propagation processes. The disease model we used in modeling disease propagation has several variables [14]. Among these variables, we choose the disease propagation transmissibility, because the propagation dynamics and episize (the final total number of infected individuals) vary with the changes of transmissibility.

From Figure 6.4, we observe that communication time during Household intervention simulations grows with episize. This is within expectation, since the intervened subpopulation size is roughly proportional to the infected subpopulation size in Household intervention. The larger these two supopulations are, the more communication time is spent in exchanging them in Indemics.

To clearly see the correlation between communication time and episize in Household intervention, we plot their ratios in Figure 6.5 (transmissibility changes cause episize changes, see Figure 6.6). We observe that their ratios are between 0.02 seconds/thousand and 0.03 seconds/thousand when episize exceeds 100,000. The communication time is proportional to episize after this threshold in these Household intervention simulations. There are a few outliers when episize is less than 100,000 because the communication overhead (such as the time it takes to send the header of a data package) cannot ignore when the subpopulations are not large enough.

We show the total communication time in Household intervention simulations with different

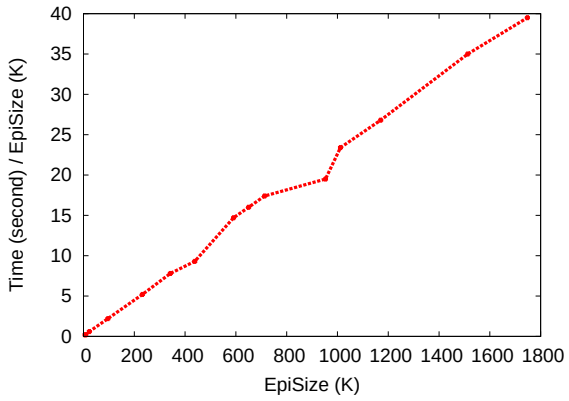

Figure 6.4: The communication time of Household intervention in the same population (Miami) with different transmissibility. It grows with transmissibility.

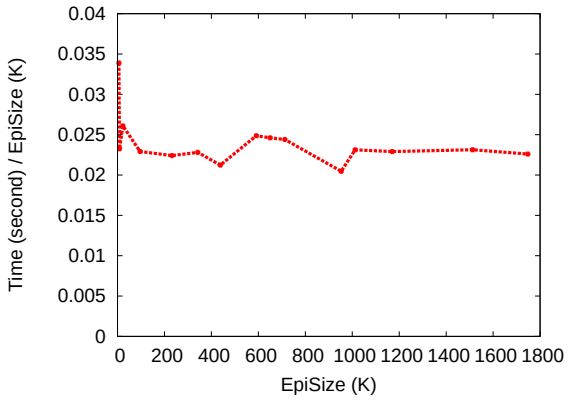

Figure 6.5: The ratios between the communication time of Household intervention simulation and episize in Miami with different transmissibility.

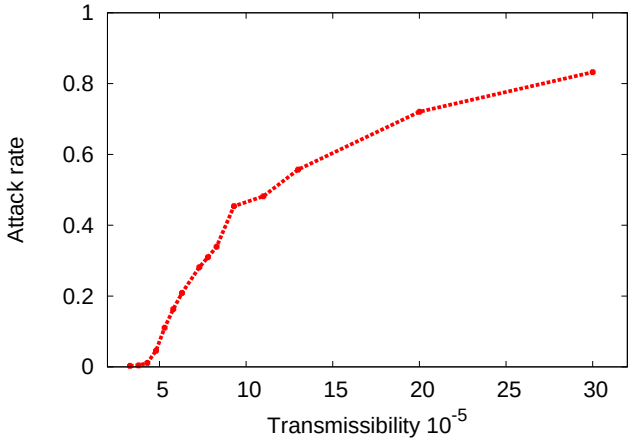

Figure 6.6: The variations of attack rate with disease propagation transmissibility without any intervention in a Miami contact network. For a fixed population, its attack rate is proportional to its episize.

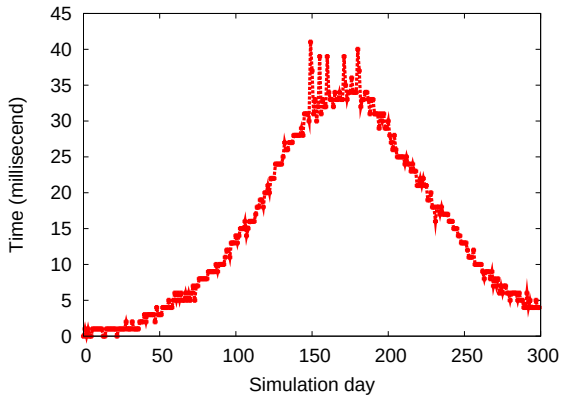

Figure 6.7: The communication time trend during the Miami Household intervention simulation.

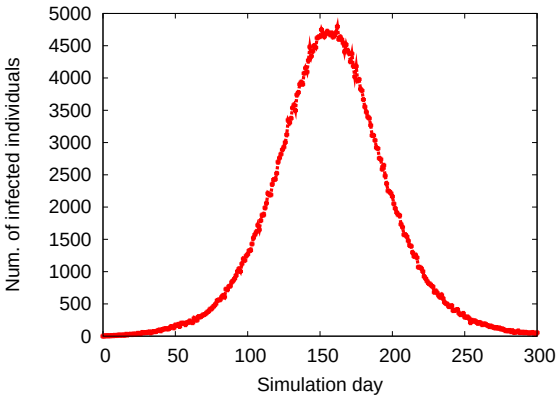

Figure 6.8: The daily number of new infected individuals during the Miami Household intervention simulation.

episizes in Figure 6.6. Figure 6.7 presents the communication time trend during a Household intervention simulation. We observe that the communication time trend of Household intervention simulation is similar to the trend of daily new infected individuals (the epicurve) shown in Figure 6.8.

Figure 6.9 shows the total communication time in Block and D1 interventions within the Miami, Chicago, and Los Angeles networks. First, the communication time shown by Figure 6.9 in the same intervention simulation with different populations grows with the population size. Second, the communication time in simulating different interventions within the same population is different. These interventions target different subpopulations, thus, they change the propagation dynamics differently.

Figure 6.10 illustrates the weight of the communication time in total simulation time (including the disease propagation simulation time and the intervention simulation time) in different intervention simulations. In these simulations, the communication time constitutes less than 30% of the total simulation time. The weight of the communication time in the D1 intervention simulations is much less than that of the others because D-1 intervention is more computationally complicated in the intervention simulation.

6.3.2 Intervention Simulation Time Using Databases

In the previous section, we presented and discussed the communication time of Indemics between the propagation simulation system and the intervention simulation system. This section will present the intervention simulation time using relational databases in detail and will discuss the factors that affect the intervention simulation time. The intervention simulation system used in the simulation experiments is an Oracle database system. The system configuration is given in Table 6.1.

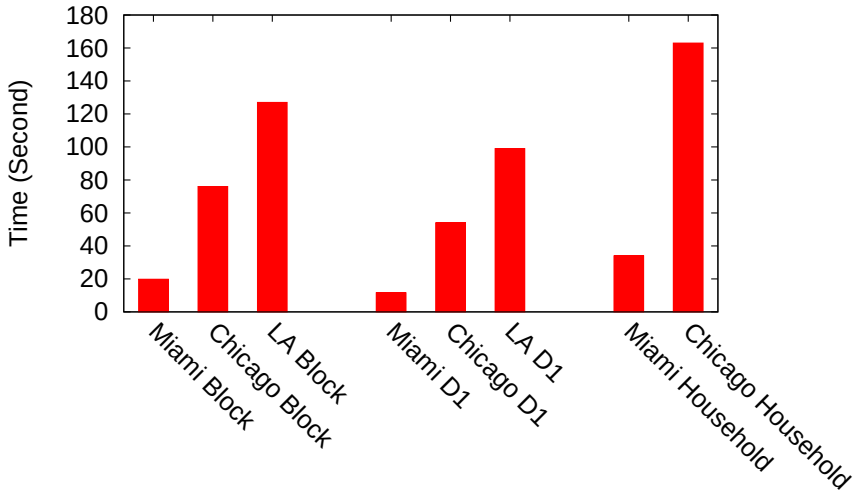

Figure 6.9: The total communication time in the Block and D1 intervention simulations.

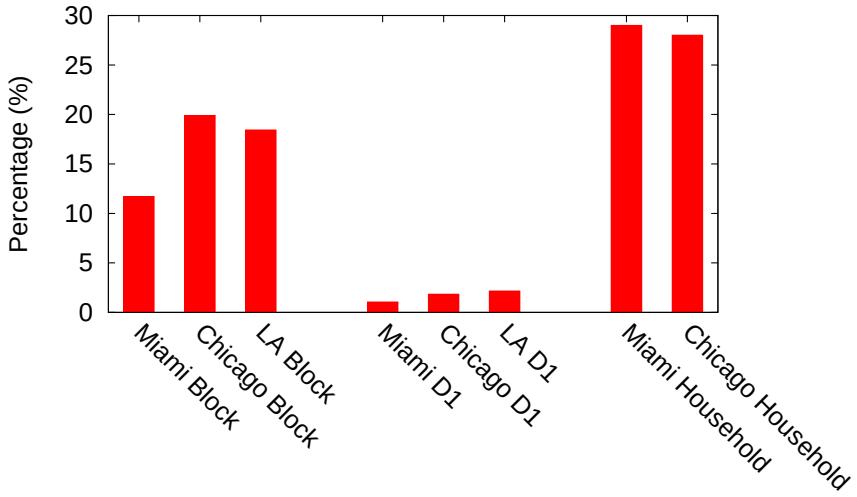

Figure 6.10: The weight of the communication cost in the Block and D1 intervention simulations.

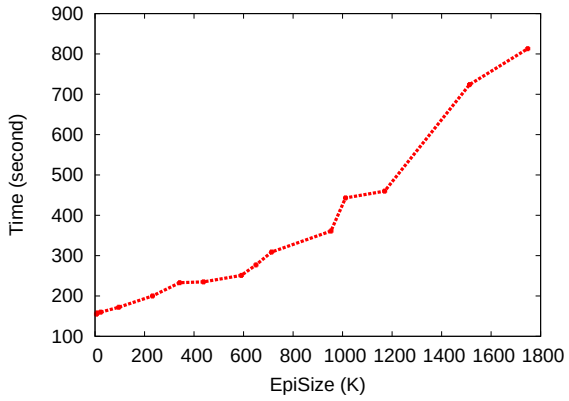

Figure 6.11: The database query time of Household intervention in the same population (Miami) with different transmissibility. The time grows monotonously with transmissibility.

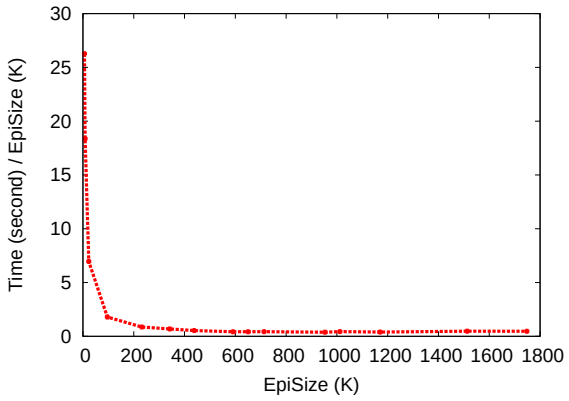

Figure 6.12: The ratios between the database query time of Household intervention simulation and episize in the same population (Miami) with different transmissibility.

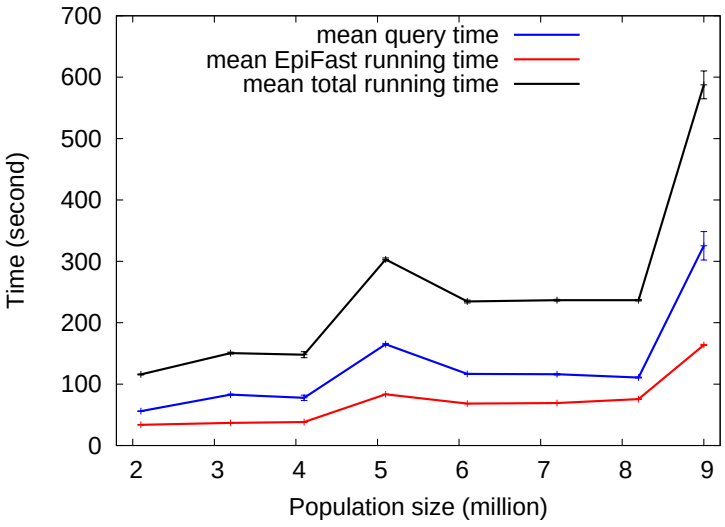

Figure 6.13: The query time of Household intervention with the same disease model in different populations. The populations include Miami, Seattle, Boston, Dallas, Indiana, Virginia, New Jersey, and Chicago. Their population sizes and average household sizes are given in Table 6.3.

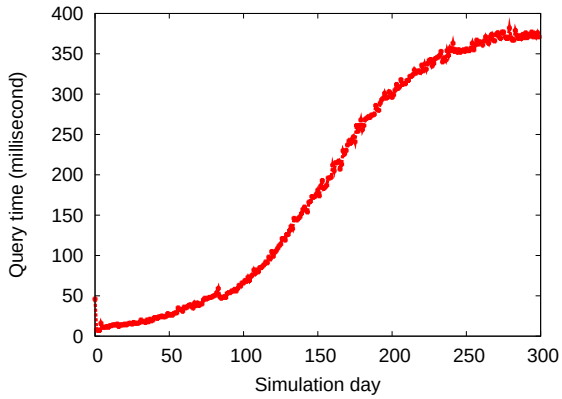

Figure 6.14: The database query time trend during the Miami Household intervention simulation.

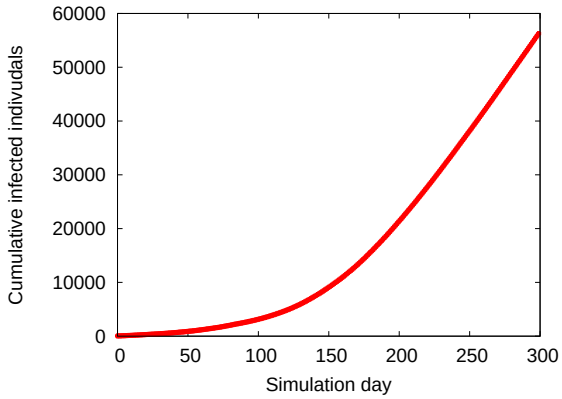

Figure 6.15: The cumulative infected individuals in the Miami Household intervention simulation.

Several factors can affect the intervention simulation time when using databases. The disease propagation dynamic is one such factor. In intervention simulations, the dynamic simulation data concerning the temporal individual health states is usually merged with other supplemental data, such as detailed demographic data about the simulation population (see the case studies introduced in Chapter 7 as examples) to test intervention triggers or to target intervened-upon individuals. Figure 6.11 shows the database query time required to simulate Household intervention with different disease transmissibility. The database query time as shown by this figure grows with episize. Figure 6.12 shows the ratio between the query time and episize. The most costly query in our Household intervention simulations is a join operation that merges the temporal individual health states and the household membership data to target the households to be intervened. The complexity of this join query depends on the size of the temporal individual health states (the daily new infected individuals are stored in the database). Thus, the ratios appear to be a horizontal line after episize exceeds a threshold. Database query overhead, which includes such things as initializing a query, leads to higher ratios than the horizontal line before the threshold.

The query time of intervention simulations also depends on the size of static data such as the total population size. Take Household intervention as an example. To target the households in which to apply interventions, we join the individual health states with the household membership data. The size of the household membership data is roughly proportional to the total population size. Figure 6.13 shows the query time of Household intervention with the same disease model in different populations. The query time does not increase monotonously with the population size because we fixed the disease model, but the disease propagation patterns are not similar in these populations due to the differences of their network structures.

Finally, the query time of each simulation day in Household intervention simulation is shown in Figure 6.14. Since the size of the database table storing daily new infected individuals increases during the simulation process, the daily query time also increases. Thus, the daily

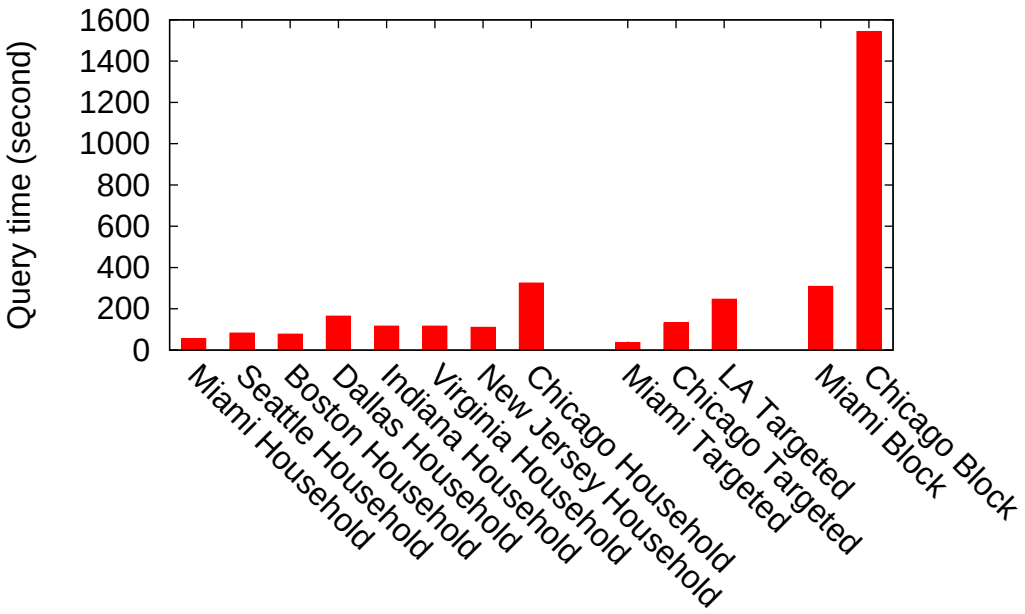

Figure 6.16: The intervention simulation time (query time) of different intervention simulations.

query time shown by Figure 6.14 demonstrates a similar trend with the cumulative infected individuals shown by Figure 6.15.

The query time of intervention simulations also depends on the applied interventions during the propagation simulations. First, the complexity of the database queries is decided by the interventions scenarios. In addition, the interventions change the disease propagation processes and, as a result, the daily new infected individuals, who are frequently queried during the intervention simulations, are different. Figure 6.16 shows the database query time in different intervention simulations with several populations. We can observe that the query time in Figure 6.16 varies in these intervention simulations with the same population. Figure 6.17 shows the weight of intervention simulation time in the total simulation time. The query time does not take more than 60% of the total simulation time in the Household and Targeted intervention simulations, but it takes nearly 90% of total simulation time in the Block intervention simulations. The Block intervention simulation is more computationally complicated than the other two. The database query time of computationally complicated interventions, such as Block intervention, drove us to improve the database performance in Indemics. We will discuss this topic in a later section.

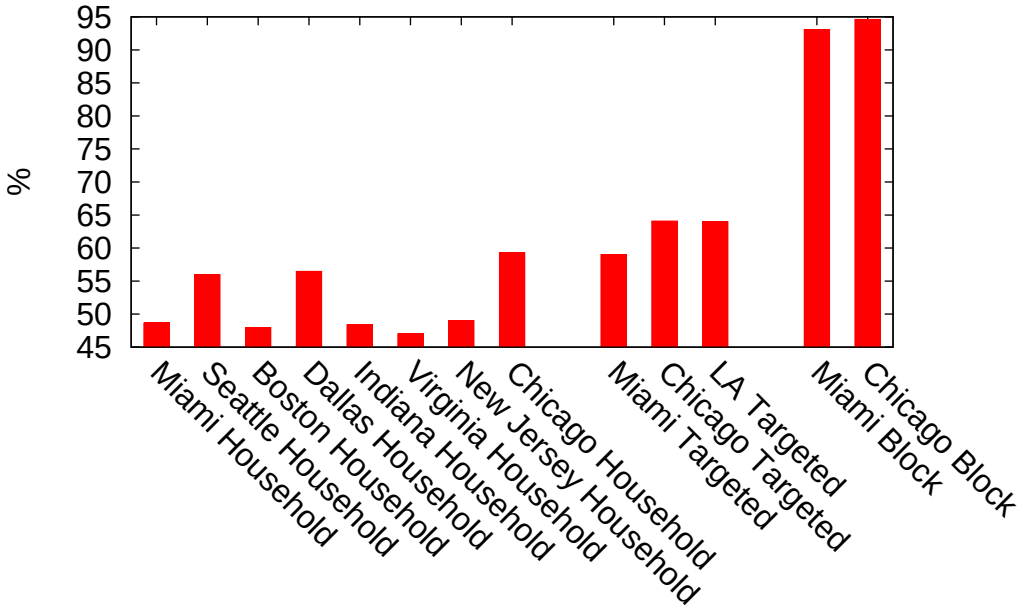

Figure 6.17: Ratio between intervention simulation time (query time) and total simulation time.

6.3.3 Performance Improvement By Using a Parallel Database

In this section, we will present the performance results of simulating interventions in Indemics using our Greenplum database system, a parallel relational database. The system configuration of our Greenplum database is given by Table 6.1. Our motivations for studying the performance of a Greenplum database system in Indemics are twofold. First, we believe data parallelism and massive parallel data processing can significantly improve database performance in our intervention simulations. Second, the Greenplum database represents a set of databases based on shared-nothing architecture, and its performance in Indemics could be useful for other ongoing research about database supported simulations.

We first present the overall simulation time of the selected interventions in three metropolitan areas. We measure the overall simulation time of one simulation iteration as the period from the simulation start to the simulation termination. Table 6.6 lists the average of the overall simulation time and the standard deviation of overall simulation time over the five iterations. D1C in Table 6.6 denotes the D1 intervention simulations using column-based database tables to store the contact networks. The other intervention simulations use row-based database tables to store their simulation data. Refer to recent work [42] for an explanation of column-based databases. From Table 6.6, we can make following observations. First, the overall simulation time of the same intervention simulation in different iterations is very close (see the standard deviations); this is because the overall propagation patterns and the

Table 6.6: The overall simulation time of all intervention simulations using Greenplum.

Population	Intervention	Average (sec)	Standard deviation	Average (min)
Miami	Block	169	1.1	2.8
	Honest	954	3.6	15.9
	D1	1109	5.9	18.5
	D1C	1067	4.8	17.8
Chicago	Block	382	4.1	6.4
	Honest	1033	3.4	17.2
	D1	2950	5.0	49.2
	D1C	2701	15	45
LA	Block	689	8.8	11.5
	Honest	1307	5.4	21.8
	D1	4586	26.5	76.4
	D1C	4117	55.1	68.6

intervened subpopulations are close in the iterations. Second, Block intervention took the shortest overall simulation time, and Honest intervention the second shortest. The overall simulation times of D-1 intervention and D-1C intervention are comparatively close in the three populations, and they are the longest simulations.

The average running time and their percentage of the overall simulation time are given in Table 6.7. We can make the following observations from this data:

First, the most time-consuming computation tasks in these simulation experiments are intervention simulation using a database and propagation simulation in EpiFast. These two tasks take approximately 60% of the overall running time of the Block intervention simulations and approximately 90% in the cases of Honest and D1 interventions. We can see from Table 6.7 that while the time of propagation simulation and the time of intervention simulation are very similar in Block interventions, propagation simulation time is dominant in Honest intervention, and intervention simulation time is dominant in D-1 intervention. The simulation time differences are rooted in the computation complexity of these interventions. The simulation of the D1 intervention scenario in the database needs to join the dynamic health state table with the large contact network table. Simulating the effect of Honest intervention in the propagation process is computationally complicated because there are different intervention actions decided by Honest intervention.

Second, the communication time and the data importing time in different intervention simulations do not vary remarkably in the same population. These two performance costs are primarily related to the size of the infected population and the size of the intervened population. The differences in communication times using the same population with different interventions are due to the effectiveness differences of these intervention strategies. The communication time and the data importing time take a small percentage of the overall sim-

Table 6.7: The running time of primary computation tasks in Indemics with our Greenplum database. The running time (sec) of primary computation tasks: the intervention simulation in Greenplum (Query), the epidemic propagation simulation in EpiFast (EF.), the communication between subsystems (Comm.), and the time (Imp.) of importing simulation results to the database.

Pop.	**Inte.**	**Total**	**Query**	**EF.**	**Comm.**	**Imp.**
Miami	Block	169	40.4 (29.3%)	52.4 (31.0%)	19.8 (11.7%)	4.7(2.8%)
	Honest	954	37.2 (3.9%)	858 (89.9%)	17.8 (1.9%)	9.4(0.99%)
	D1	1109	762 (68.7%)	294 (26.5%)	11.6 (1.0%)	6.0(0.54%)
	D1C	1067	723 (67.8%)	291 (27.3%)	12.4 (1.2%)	6.0(0.56%)
Chicago	Block	382	130 (34.0%)	83 (21.7%)	76 (19.9%)	13.7(3.6%)
	Honest	1033	102 (9.9%)	779 (75.4%)	78 (7.6%)	14.8(1.4%)
	D1	2950	2505(84.9%)	320 (10.8%)	54 (1.8%)	12.5(0.42%)
	D1C	2701	2231(82.6%)	327 (12.1%)	58.2 (2.2%)	13.5(0.5%)
LA	Block	689	259 (37.6%)	168 (24.3%)	127 (18.4%)	22.8(3.3%)
	Honest	1307	212 (16.2%)	858 (65.7%)	127 (9.7%)	24.1 (1.9%)
	D1	4586	3873 (84.5%)	460 (10.0%)	99 (2.2%)	23.9(0.52%)
	D1C	4117	3418 (83.0%)	457 (11.1%)	95 (2.3%)	22.8(0.55%)

ulation time, except in Block intervention. These two performance costs take a considerable percentage in Block intervention because the overall simulation time of Block intervention is much shorter than that of other interventions; thus they become distinct.

Third, the D1 intervention simulation is the most time-consuming. The major performance cost is the intervention simulation using a database due to a table join operation on the large contact network table. We organize the contact networks in column-based tables. By only reading and writing the queried columns in a column-based table instead of the whole large table, the disk I/O cost can be reduced [53, 2, 1].

We simulate D-1 intervention using row-based contact network tables (the default table structure) and column-based contact network tables in Miami, Chicago, and Los Angeles. The overall simulation time and the database query time for the intervention simulation are given in Table 6.8. Both times decrease when using column-based tables in these three areas, but the time reduction is not significant. The overall simulation time and the database query time are about 10% less in these three areas. Although the contact network table is very big, it is read only once during the entire simulation experiment. Column-based tables can reduce the disk I/O cost of table reading, but they do not reduce the computational complexity of the queries.

Finally, we compare the query time differences of using our Oracle and Greenplum databases. As shown by Figure 6.17, the query time of the Block intervention simulation using our Oracle database is the bottleneck. Figure 6.18 shows the database query time of the Block intervention simulations using our Oracle database and Greenplum database. The query

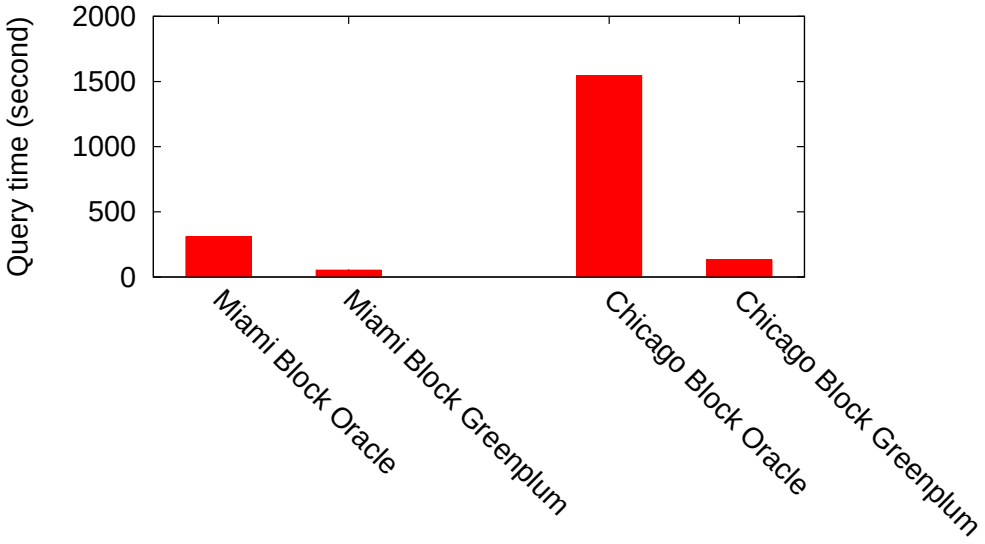

Figure 6.18: The database query time of Block intervention simulations using Oracle and Greenplum.

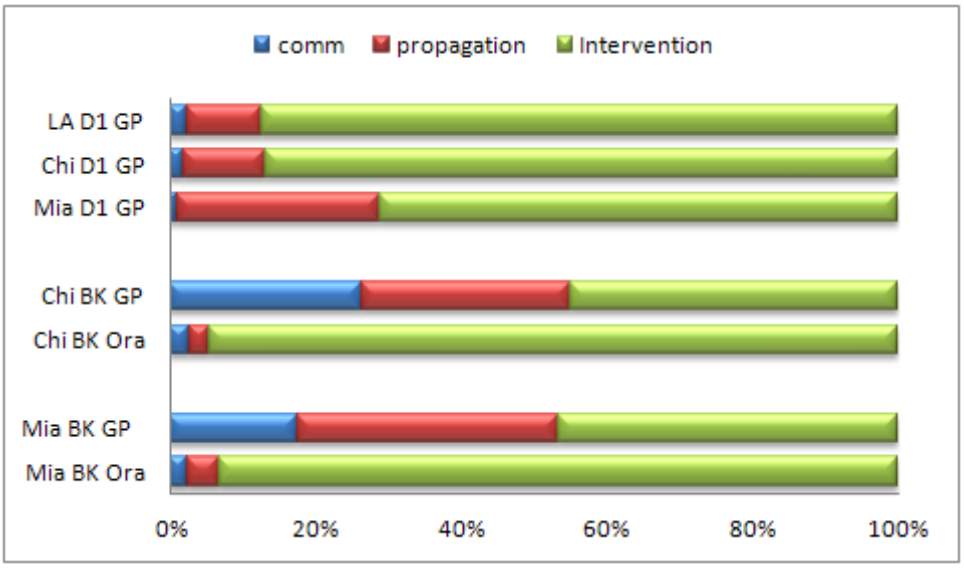

Figure 6.19: The proportion of the primary costs of Indemics using Oracle and Greenplum in different intervention simulations. Mia, Chi, LA denote Miami, Chicago, and Los Angeles respectively. BK and D1 denote Block intervention and D1 intervention respectively. Ora and GP denote Oracle and Greenplum. The differences between the communication time and the propagation simulation time are due to running the propagation simulations using two computing systems.

Table 6.8: The total running time (minutes) and query time of D1 intervention using row-based network tables and column-based network tables.

Population	Row-based table		Column-based table		Column-based/Row-based	
	Total	**Query**	**Total**	**Query**	**Total**	**Query**
Los Angeles	76.4	64.6	68.6	57.0	89.8%	88.3%
Chicago	49.2	41.8	45	37.2	91.5%	89.0%
Miami	18.5	12.7	17.8	12.1	96.2%	95.3%

time has been remarkably reduced by using our Greenplum database due to its parallel data processing. Figure 6.19 shows the proportion of the primary costs of Indemics using the Oracle database and Greenplum databases in different intervention simulations. The query time is the bottleneck when our Oracle database is used, but the query time of using our Greenplum database is not as dominant as with the Oracle database. In addition, we are not able to simulate D-1 intervention using our Oracle database due to performance issues, but D-1 intervention can be simulated using our Greenplum database without major performance issues.

6.4 Chapter Summary

We have proposed a database supported simulation framework, Indemics, in the early chapters. In this chapter, we continued the evaluation of this simulation framework and we focused on the computational efficiency of Indemics. We examined its performance overhead compared with one previous system, since the system decoupling and use of relational databases instead of high performance computing technology could cause performance losses.

In order to better understand the performance of Indemics, we examined its performance in various scenarios and under different conditions in detail. We analyzed the performance of the key computation tasks in Indemics from different perspectives. We also verified our hypothesis of computational efficiency using detailed performance results.

Chapter 7

Capability Improvement in Intervention Modeling

One of the primary objectives of building this environment for epidemic propagation and intervention simulation is to provide public health researchers and policy makers with a platform to efficiently study epidemic propagation processes while introducing diverse interventions. In this chapter, we introduce two real intervention case studies whose simulation experiments were performed using Indemics. We present the scenarios of the intervention strategies, our intervention modeling, the database design of these studies, and brief simulation results. We show how to prepare and perform intervention studies using Indemics with these concrete examples to help public health researchers and policy makers understand its capability in simulating complex interventions and its usage. The introduction of these studies could motivate them to employ Indemics in their future intervention studies.

7.1 Public Interventions and Individual Protection

7.1.1 Introduction

This research compares the performance of bottom-up, self-motivated behavioral interventions with top-down interventions targeted at controlling an "influenza-like-illness". Both types of interventions use a variant of the *ring strategy*. In the first case, when the percentage of a person's direct contacts who are diagnosed exceeds a threshold, that person decides to seek prophylaxis, e.g., vaccine or antivirals; in the second case, we consider two intervention protocols, Block intervention and School intervention. Results show that the bottom-up strategy outperforms the top-down strategies under most scenarios. Even in situations where the Block strategy reduces the overall attack rate well, it incurs a much higher cost. These findings show that if people use antivirals effectively, making them available quickly and on

demand to private individuals can be a very effective way to control an outbreak.

The goal is to understand how individualistic actions, based on personal knowledge and beliefs, and aimed at self protection, fare in comparison to similar actions imposed by public policy makers who depend on private individuals to comply with their orders. This research compares the performance of three variants of ring strategies, a self-motivated, self-imposed, "bottom-up" strategy and two "top-down" public health intervention strategies. The bottom-up strategy depends on each person's private awareness about the health state of his/her direct social contacts, while the top-down strategy is based on public information about disease prevalence in a school or census block group.

The bottom-up strategy (D1) works almost like an inverse ring strategy. Under this strategy, self-motivated private individuals take action when a percentage of their direct contacts who are diagnosed exceeds a threshold. The top-down strategies are Block intervention and School intervention. The outbreak is assumed to be observed when the current percentage of people diagnosed exceeds a threshold. In all three strategies considered here, the experiments are conducted with outbreaks defined by threshold fractions of 1% and 5%. The factorial experiment design of the intervention policies are given in Table 7.1.

Table 7.1: Experimental design for the study of the top-down and bottom-up strategies. The factorial design shows the factors and the parameters used in the experiment. The results are reported based on the average of 25 replicates for each case.

Strategies: Distance 1 neighbors (D1), *Block*, *School*
Threshold value for taking actions: 0.01 or 0.05 diagnosed
Individual vs. local threshold: D1 intervention depends upon the percentage of direct contacts diagnosed; block (school) intervention on the percentage of block group (school) subpopulation diagnosed
Compliance rates: 100% or 50%
Interventions: Antiviral (AV) and Vaccination (VAX)
Delay in implementing interventions: 1 day or 5 days delay for *Block and School*; B1, B5, S1 and S5 reflect 1 and 5 days delay for *Block and School* respectively. No delay in D1.
Delay in effectiveness: AV are immediately effective but VAX become effective after 2 weeks
Duration of effectiveness: Each AV course is effective for 10 days and VAX is effective forever.
Simulation days and replicates: 200 simulation days and 25 replicates.

7.1.2 Intervention Scenario Modeling

As introduced as above, the bottom-up intervention strategy is modeled as its intervention action and its intervened subpopulation. The protective actions taken by the self-motivated

individuals are common protective actions, seeking medical treatment. The intervened subpopulation of D1 is the individuals whose diagnosed direct contacts exceeds a percentage of the total contacts. We translate the D1 intervention scenario into the following SQL scripts. We choose 0.05 as the threshold of the protective action.

```
for each simulation day $i = 0 to 200
     insert into Num_Of_Today_Diagnosed_Neighbors(pid, neighbors, day)
     select n.tail, count(n.head), $i from contact_network n,
     diagnosed_cases d where n.head = d.pid and d.day = $i group by tail;

     insert into Num_Of_Sick_Neighbors (pid, neighbors, day)
     select pid, sum(neighbors), $i
     from Num_Of_Today_Diagnosed_Neighbors where day
     between $i - 3 and $i group by pid;

     update Person_Intervened set intervened = $i where intervened = 0
     and pid in (select pid from Num_Of_Sick_Neighbors sn,
     Person_Intervened int where sn.day = $i and
     sn.neighbors/int.neighbors > 0.05 and sn.pid = int.pid);

     select pid from Person_Intervened where intervened = $i;
```

We simulate D-1 intervention in an epidemic propagation process. D-1 intervention becomes effective from the first simulation day when the propagation starts to the 200th simulation day when the propagation dies out. The computation identifying the individuals who cross the intervention threshold is divided into four steps. The first step is integrating the daily new diagnosis cases with the contact network. As shown in above script, the contact network table and the diagnosis case table are joined to count the number of contact neighbors diagnosed today for every individual. The individuals and the quantities of their neighbors diagnosed today are stored in table '*Num_Of_Today_Diagnosed_Neighbors*'. The second step is counting the number of diagnosed contacts who are still sick and not recovered. In this experiment, we assume every individual will be sick for 3 days to simplify the simulation computation. After the calculation of the sick and diagnosed cases of each individual, we now are able to check the intervention threshold for each individual. As the third step in the script, we compare the number of direct contact neighbors and the number of diagnosed neighbors for each individual. The individuals who cross the threshold are marked (update the 'intervened' property in table 'Person_Intervened' to current simulation day). The final step is selecting the individuals crossing the threshold. The individual identifications and intervention action, such as vaccination or taking antivirals, are forwarded to the propagation simulation system to simulate the propagation process during the next simulation day after applying the interventions of the current day. A similar computation of D-1 intervention is

Table 7.2: Schema of database tables used in D-1 intervention.

Table Name	**Table Schema**	**Description**
Diagnosed_cases	pid number day number	Which day the individuals 'pid' are diagnosed
Contact_network	head number tail number	The social contacts between the heads (pid) and the tails (pid)
Num_Of_Today _Diagnosed_Neighbors	pid number neighbors number day number	The number of today diagnosed contact neighbors. 'pid' has 'neighbor' neighbors diagnosed on day 'day'
Num_Of_Sick_Neighbors	pid number day number neighbors number	The number of sick contact neighbors. individual stays in the sick status for 3 days after diagnosis.
Person_Intervened	pid number intervened number neighbors number	Individual 'pid' has 'neighbors' neighbors, 'pid' has cross the intervention threshold on day 'intervened'.

repeated in each simulation day. The schema of the tables used in the above script is given in Table 7.2.

The Block intervention scenario can be expressed as following SQL statements.

```
for each simulation day $i = 0 to 300
     insert into Num_Of_Today_Diagnosed (block_id, infections, day)
     select block_id, count(b.pid), $i from block_info b, diagnosed_case d
     where day = $i and  b.pid = d.pid group by block_id;

     insert into Num_Of_Sick_Residents (block_id, infections, day)
     select block_id, sum(infections), $i from Num_Of_Today_Diagnosed
     where day between $i - 3 and $i group by block_id;

     update Block_Intervened set intervened = $i where intervened = 0
     and block_id in (select block_id from Num_Of_Sick_Residents sr,
     Block_Intervened int where sr.day = $i and
     sr.infections/int.residents > 0.05 and sr.block_id = int.block_id);

     select pid from block_info where block_id in
     (select block_id from Block_Intervened where intervened = $i );
```

The School intervention scenario has been introduced in Chapter 5 as an intervention example to model and predict the simulation time. We present the SQL script modeling the School

Table 7.3: Database tables used in the Block intervention simulation

Table Name	**Table Schema**	**Description**
Diagnosed_cases	pid number day number	Which day the individuals 'pid' are diagnosed
Block_info	pid number block_id number	The residents 'pid' are in block 'block_id'
Num_Of_Today _Diagnosed	pid number infections number day number	The number of today diagnosed residents. Block 'block_id' has 'infections' diagnosed residents on day 'day'
Num_of_Sick_Residents	block_id number infections number day number	The number of sick residents in each block. Individual stays in the sick status for 3 days after diagnosis.
Block_Intervened	block_id number infections number intervened number	Block 'block_id' has 'infections' sick residents. Outbreak occurs in block 'block_id' on day 'intervened'.

intervention scenario again here. The computation of School intervention is very similar to Block intervention. Therefore, we do not explain its computation in detail again.

```
for each simulation day $i = 0 to 300
     insert into Num_Of_Today_Diagnosed (school_id, infections, day)
     select school_id, count(b.pid), $i from Person_School s,
     diagnosed_case d where day = $i and  s.pid = d.pid group by school_id;

     insert into Num_Of_Sick_Students (school_id, infections, day)
     select school_id, sum(infections), $i from Num_Of_Today_Diagnosed
     where day between $i - 5 and $i group by school_id;

     update School_Intervened set intervened = $i where intervened = 0 and
     school_id in (select school_id from Num_Of_Sick_Students ss,
     Block_Intervened int where ss.day = $i and ss.infections/int.students
     > 0.01 and ss.school_id = int.school_id);

     select pid from Person_School where school_id in
     (select school_id from School_Intervened where intervened = $i );
```

Table 7.4: Database tables used in the School intervention simulation

Table Name	**Table Schema**	**Description**
Diagnosed_cases	pid number day number	Which day the individuals 'pid' are diagnosed
School_info	pid number school_id number	The students 'pid' are studying in school 'school_id'
Num_Of_Today _Diagnosed	pid number day number infections number	The number of today diagnosed students. School 'school_id' has 'infections' diagnosed students on day 'day'
Num_of_Sick_Students	school_id number day number infections number	The number of sick students in each school. individual stays in the sick status for 3 days after diagnosis.
School_Intervened	school_id number intervened number infections number	School 'block_id' has 'infections' sick students, Outbreak occurs in school 'school_id' on day 'intervened'.

7.1.3 Case Study Efficiency

During the 2009 H1N1 outbreak, our group planned to perform this intervention strategy study to support the H1N1 control. We expected to present simulation results at a CDC conference two weeks later. We decided to use Indemics to run the study, because the case study period was too short to implement these complex intervention strategies in EpiFast. After the epidemiologists in our group designed the intervention scenarios, we translated their intervention scenarios into intervention scripts. Using the performance prediction methodology introduced in Section 5.2, we predicted that if we adopted the current version of the intervention scenarios, the simulation experiments would take more than two weeks. The epidemiologists modified the scenarios and we translated the new scenarios into the above intervention scripts. We estimated that the simulation experiments of the new scenarios could be done within one week. We spent one day preparing the experiments, including setting up the computing environment for the simulation experiments. Then we started our simulation experiments with thousands of different cases. The whole experiment took about one week as we predicated. The epidemiologists analyzed simulation results, prepared a report, and presented it at the conference two weeks later.

7.1.4 Simulation Results

Since the analysis of the effectiveness of these interventions in disease propagation control is not closely relevant to this dissertation topic, we do not present the simulation results with details in this dissertation. We present the epicurves generated from applying the bottom-up strategy and top-down strategies in Figure 7.1. We present these figures because we use these

interventions in the Indemics performance studies (Chapter 6) and the subpopulation sizes and disease propagation patterns of applying these interventions are useful for the analysis of system performance. The complete simulation results and the analysis of these disease propagations are presented in the literature [44].

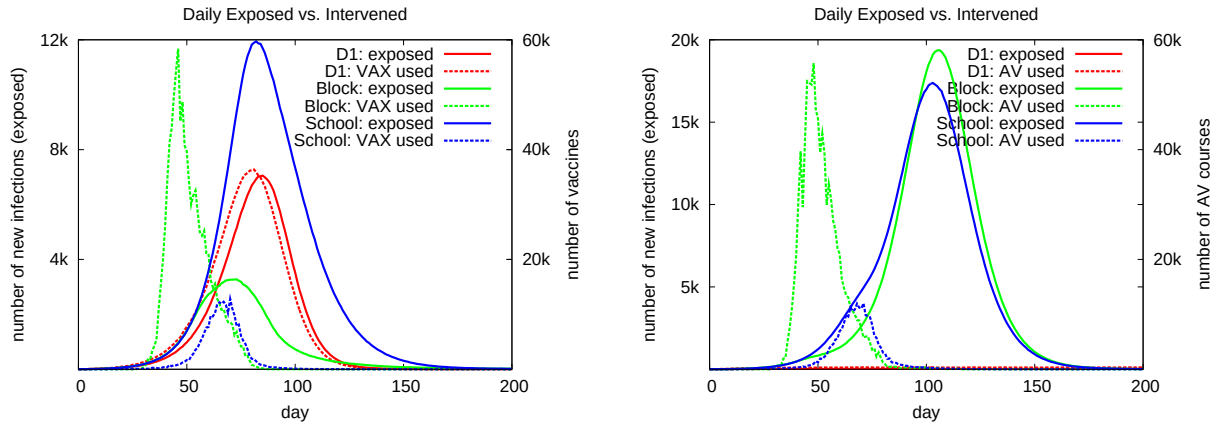

Figure 7.1: Number of people exposed versus the number of vaccines used on a daily basis under each of the three strategies considered.

7.2 Sick Leave Policy

7.2.1 Introduction

This study focuses on an economically driven intervention, i.e. a paid sick leave policy that allows the workers to stay home from work without loss of income. We believe that the sick leave policy as a tool to contain epidemics has not been studied in detail in the health care literature, but noteworthy exceptions are [29, 51].

Previous researchers have pointed out that presenteeism may be more damaging than absenteeism. From public health viewpoint, it is desirable to reduce the disease *attack rate*, defined as the percentage of the population becoming infected. A liberal sick leave policy will discourage sick employees from coming to work which will help contain the disease, but it will also affect the productivity of the society. To determine whether a sick leave policy is indeed an effective tool for controlling epidemics and whether it is economically efficient, our paper considers a variety of sick leave policies and worker behavior. The analysis takes a social welfare point of view to study the cost effectiveness by comparing the productivity loss of sick workers with the socio-economic gain caused by a lower attack rate in the population.

A detailed experimental design considers a variety of scenarios based on the number of paid

Table 7.5: Factorial design for the study of the Sick-leave policy

Factor	**Description**	**Values**
Dis	disease types: catastrophic and moderate flu.	Cat, Moderate
Comp	compliance: probability each workplace complies with the sick leave policy	50%, 100%
$D_{\max}$	sick leave policy: max number of sick days allowed to diagnosed workers	3, 4, 5, 6
Beh	workers' behavior towards the sick leave policy: take the exact sick days off (honest) or take the maximum possible days off (rational)	rational, honest
e	productivity level of those working while sick	20%, 50%, 80%

sick days allowed, disease type, the behavior of the workers and the compliance level of the employers with the sick leave policy. The worker behavior is assumed to be of two types: *rational* and *honest*. In case of rational behavior, the workers take the maximum number of sick days available regardless of how long they are sick, but those with honest behavior take off only the number of days they are sick for. Honest behavior can also be thought of as a proxy for full information sharing, or symmetric information between the employer and employees; and rational behavior can be interpreted as partial information sharing or asymmetric information between the employer and employees [51]. Our simulation results show that if employees behave honestly, a liberal sick leave policy would maximize the social benefit, but if they behave rationally the social benefit is maximized when the paid sick days are equal to the mean number of sick days. In the rational case, a liberal sick leave policy reduces the overall social welfare.

We considered the following 5 factors in our experimental design: disease type, compliance level, maximum number of sick days allowed, workers' behavior and the productivity level of workers who work while they are sick. The disease types are moderate flu or catastrophic flu. The compliance levels of the employers/workplace are set at 50% and 100%, which refer to the percentage of work locations that comply with the sick leave policy. In the 50% compliance case, only 50% of the work locations choose to comply and provide paid sick leave to its employees. In the 100% case, all work locations and hence all workers are given paid sick leave. However note that only diagnosed workers are allowed to take sick leave.

Regarding the number of maximum paid sick days, workers can take sick leave up to $D_{\max}$ =3, 4, 5, or 6 days without any income loss. Workers' behavior is considered to be of 2 types, rational and honest; if rational, workers take all available sick leave so the actual number of sick leave days taken is $D_{sl} = D_{\max}$. In the honest case, eligible workers take off only when they are sick: $D_{sl} = \min(D_{sick}, D_{\max})$, where D_{sick} is the actual number of sick days. All of the experimental factors are summarized in Table 7.5.

7.2.2 Intervention Scenario Modeling

This liberal sick leave intervention changes the regular daily activities of workers after they are diagnosed. Therefore, the worker information is necessary in this intervention simulation. The worker populations are stored as database tables. In this study, we chose Miami as the simulation population and the database schema of the synthesized Miami workers are given in Table 7.6.

Since the entire factorial design has tens of different intervention scenarios, we only present the translated script for the scenario of the sick-leave rational case with 5 maximum leaving days.

```
for each simulation day $i = 0 to 300
     apply intervention: stay home, 5 days, select pid from
     Worker_Info w and Diagnosed_cases d
     where w.pid = d.pid and d.day = $i;
```

The daily intervened workers who should take sick leave are targeted by a database query, which joins the daily diagnosed individual table with the worker table to select the daily new diagnosed workers. A stay-home intervention action for 5 days and the identifications of the intervened workers are sent to the disease propagation simulation system to modify their activities in the coming 5 days.

The translated script for the sick-leave honest case with the 5 maximum leaving days is illustrated as following:

```
for each simulation day $i = 0 to 300
     apply intervention: stay home, 3 days, select pid from Worker_Info w,
     Infection_Cases i and Diagnosed_cases d
     where w.pid = i.pid and i.pid = d.pid
     and d.day = $i and i.recovered_day - $i = 3;

     apply intervention: stay home, 4 days, select pid from Worker_Info w,
     Infection_Cases i and Diagnosed_cases d
     where w.pid = i.pid and i.pid = d.pid
     and d.day = $i and i.recovered_day - $i = 4;

     apply intervention: stay home, 5 days, select pid from Worker_Info w,
     Infection_Cases i and Diagnosed_cases d
     where w.pid = i.pid and i.pid = d.pid
     and d.day = $i and i.recovered_day - $i > 4;
```

Table 7.6: Schema of database tables used in the Sick-leave intervention

Table Name	**Table Schema**	**Description**
Diagnosed_cases	pid number day number day number	Which day the individuals 'pid' are diagnosed
Infection_cases	pid number infected_day number recovered_day number	the individuals 'pid' are infected on day 'infected_day' and recover on day 'recovered_day'
Worker_Info	pid number	The workers pids

In contrast to the rational behavior case, workers behave differently based on their recovered time. In the honest case, workers will voluntarily come back to their workplaces whenever they are recovered with the 5 sick leave days. Therefore, we classify the intervened workers into three groups. Group one is the workers who are diagnosed today and will recover 3 days later; group two is the workers who are diagnosed today and will recover 4 days later; group threes is the workers who are diagnosed today and will not recover in the next 5 days. We assume the diagnosed individuals will be sick at least for 2 days in our disease models. We search the workers in these three groups by database queries using the above script, and make the intervened workers to stay home for 3, 4, and 5 days respectively.

7.2.3 Case Study Efficiency

The experimental procedure of this sick leave study is different from our experience of the CDC study introduced in Section 7.1. We did not have a very tight case study period as we experienced in the CDC study. Although we did not have a deadline, we experienced another challenge: how to model this policy in a realistic and computationally efficient way. We tried several different models for this sick leave policy, implemented them in our simulation system and performed preliminary experiments to see the simulation results and computational efficiency. In each version of these models, the development work only took us several hours in translating intervention scenarios into Indemics intervention scripts. Without Indemics, we would not have been able to try so many intervention models within a short period.

7.2.4 Effect of the Sick Leave Policy

The epicurves of applying this policy are shown in Figure 7.2 for the catastrophic flu and Figure 7.3 for moderate flu. In both figures, we show the epicurve for the base case (i.e. no sick leave at all) and the curves for $D_{\max}$ values set at 3 days and 6 days only. The epicurves for $D_{\max} = 4$ and $D_{\max} = 5$ are very close to the 3-day and 6-day cases and are hence omitted to avoid clutter. Table 7.7 shows the total attack rate, peak day and peak size for all values of $D_{\max}$ and compliance levels.

It is important to note that the epidemics did not change across honest and rational cases, because in the SEIR model a recovered individual does not transmit the disease to others and cannot be infected by others either. The honest case and the rational case differ only when a sick worker recovers before $D_{\max}$ is reached: an honest worker will then go back to work while a rational worker will remain out of work. But it does not matter with respect to the disease spread because this worker is already recovered. Therefore the epidemic curves are the same under these two cases and we show the curves only once.

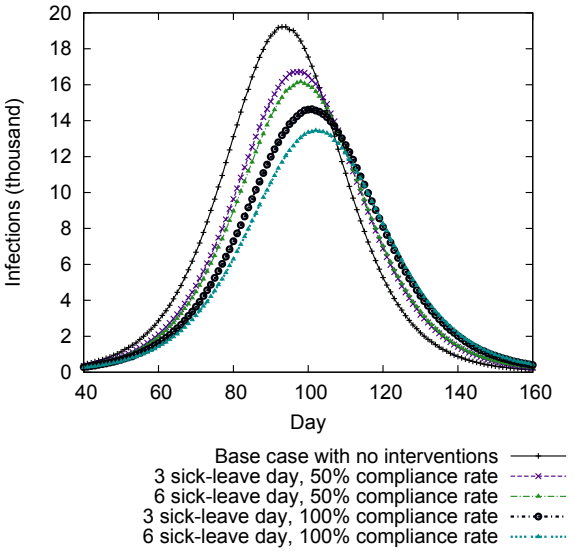

Figure 7.2: Epidemic curves under the catastrophic flu case

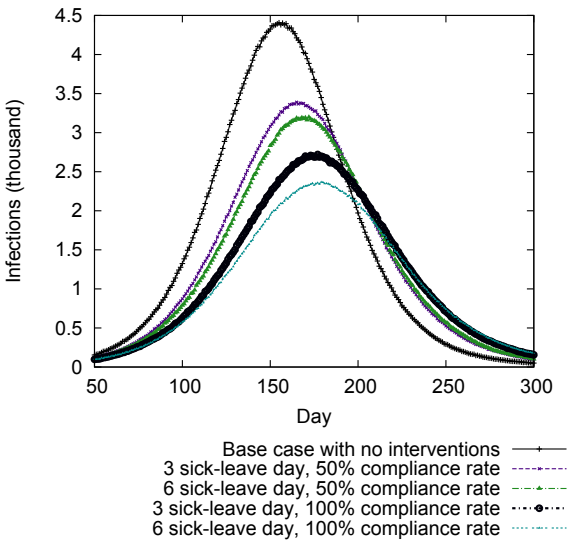

Figure 7.3: Epidemic curves under the moderate flu case

Figure 7.2 and Table 7.7 clearly show that the sick leave policy has a significant effect on the epidemic dynamics in catastrophic case. At 50% workplace compliance, the sick leave policy of 5 days can reduce the peak size from 19,000 to 16,000, i.e. by 15%. It can be reduced by another 15% to 13,600 if the compliance goes up to 100%. The effect is more prominent than it appears because the targeted intervened people only account for at most 6% (in case of 100% compliance rate) or 3% (in case of 50% compliance rate) of the total population[1]. The results show that the higher the compliance, the lower the overall attack rate, and changes to the maximum number of sick days by even a single day can cause statistically significant change to the attack rate and the medical costs. A longer sick leave results in lower attack rates which is not surprising, but the marginal effect due to an extra day of sick leave is nonlinear. The sick leave policy can help postpone the peak day by 5-8 days depending upon the compliance rate.

[1]This is because only one third of the population is workers. In addition, the attack rate is at most 40%, the symptomatic rate is 2/3, and diagnosed rate is 2/3 which makes the targeted people only a small fraction of the society.

Table 7.7: The simulation results in the Sick leave study. This table shows the percentage of people infected (attack rate), peak infection day (peak day), and the maximum number of infections on one day (peak size) in the catastrophic and moderate flu cases along with the base case (no intervention).

scenario	**catastrophic flu**			**moderate flu**		
	attack rate	**peak day**	**peak size**	**attack rate**	**peak day**	**peak size**
base case	40.0%	93	19,000	20.0%	155	4,400
Comp=50%, $D_{\max}$=3	37.3%	98	16,700	17.4%	165	3,400
Comp=50%, $D_{\max}$=4	36.5%	98	16,200	16.9%	169	3,200
Comp=50%, $D_{\max}$=5	36.2%	99	16,000	16.7%	168	3,200
Comp=50%, $D_{\max}$=6	36.1%	98	16,000	16.7%	170	3,200
Comp=100%, $D_{\max}$=3	34.5%	101	14,600	15.1%	177	2,700
Comp=100%, $D_{\max}$=4	33.1%	101	13,800	14.3%	177	2,500
Comp=100%, $D_{\max}$=5	32.5%	101	13,600	13.9%	180	2,400
Comp=100%, $D_{\max}$=6	32.4%	102	13,400	13.8%	179	2,400

A similar pattern is found in Figure 7.2 and Table 7.7 for the moderate case. The policy here has even more significant effects. It reduces the peak size by 27% compared to the base case (from 4400 to 3200) when compliance is at 50%. At 100% compliance the peak size is reduced by another 25%. The peak day can be delayed by 15 days at 50% compliance and another 10 days at 100% compliance.

For both catastrophic flu and moderate flu, the marginal effect of one additional day of sick leave is decreasing. When $D_{\max}$ changes from 3 to 4, the effect on attack rate reduction is fairly large but as we increase $D_{\max}$ to 5 and 6 days, the marginal effect of additional sick days on the attack rate starts to decrease. This can be explained by the fact that the mean infectious period is 4.1 days and hence most people stay infectious for 4 days; there are relatively fewer people who stay sick for 5 or 6 days. Therefore a greater number of sick leave days are not needed for a large proportion of the population and the marginal improvement in the attack rate from the additional days drops.

This case study also modeled and examined the economic benefit and cost of a paid sick leave policy. Since the economic analysis is not the focus of this dissertation, we do not elaborate on it here. Liao, et al [43] published the details about the models and analysis.

7.3 Chapter Summary

In this chapter, we introduced two realistic intervention studies and showed how we prepared and performed these studies using Indemics. We presented intervention strategy modeling, database design, and implementation of these interventions using Indemics. The interven-

tions studied in this chapter are challenging to implement in previous simulation systems, but they have been modeled in hours using Indemics. These concrete examples demonstrate the capability improvement and human productivity improvement gained by using Indemics.

Chapter 8

Closing

In this dissertation we focused on the system design of a modeling environment for pandemic planning and course of action analysis. Previous simulation systems designed for this modeling environment have limitations in the user expected features. In this dissertation, we revisited the expected features of this environment, analyzed the limitations of the previous systems, and proposed a new system design to improve the modeling environment.

8.1 Conclusions

With the objective of improving this modeling environment with respect to the expected features, we explored two concepts in the simulation system design: 1) abstracting the key modeling subtasks in this modeling environment, decoupling their simulations, and applying suitable computing technologies in the simulation subsystems, and 2) utilizing relational databases for the epidemic situation assessment and intervention simulation. The motivation of improving the modularity and extensibility of the modeling environment drives us to explore these concepts. Using relational databases can improve the capability of supporting complex intervention studies and increase human productivity while using this modeling environment. To verify the feasibility and effect of these concepts, we proposed several research hypotheses in section 1.4. The work introduced in this dissertation verifies these hypotheses.

Chapter 3 proposed an abstract and loosely coupled system to model the co-evolving process of the epidemic propagation, the situation assessment, and interventions. Chapter 4 introduced a simulation framework based on the abstract and loosely coupled system. From the work introduced in Chapters 3 and 4, we concluded that a loosely coupled simulation system is feasible if sharing run-time system states. The subsystems are implementation-independent in this loosely coupled system, as a result, we can apply suitable computing technologies in the subsystem implementation.

Chapter 3 introduced our intervention model and Chapter 4 presented how we simulate interventions using our database supported simulation system. Based on our analysis, we concluded that simulating a large scope of complex interventions using this database supported simulation system is possible. In addition, the database supported simulation system also enables run-time situation assessments and adaptive interventions.

We have verified the system decoupling hypothesis and the database utilization hypothesis in Chapters 3 and 4, but also needed to examine the computational efficiency of the proposed simulation system. Chapter 5 introduced our performance studies of Indemics. From the performance results and our analysis, we showed the loosely coupled and database supported simulation system is computationally efficient. Although Indemics may incur performance overhead compared with the previous system, we believe the potential performance loss is acceptable if its improvements on other expected features are considered such as its capability of supporting complex studies and its extensibility to special purpose applications.

To summarize, we concluded that the proposed system introduced in this dissertation balances the expected features of the epidemic modeling environment, and the simulation framework based on this system design shows better overall performance than the previous simulation systems.

8.2 Future Work

The research work introduced in this dissertation which explores a new system design for modeling environment is not complete. Many opportunities exist for future research.

- To simplify the usage of performing situation assessments and intervention studies in Indemics, a domain script language which is easy to understand and capable of expressing intervention scenarios is demanded. We presented our preliminary progress on the domain script language design in Chapter 7. More research effort is needed on its grammar and expression capability.

- Chapter 4.3 introduces the Indemics API design based on the requirements for special applications extended from Indemics. The API callings introduced here may not be sufficient for future application extensions. We expect to create further API extensions when we are familiar with more special purpose applications.

- Chapter 4.1 introduces a star-shaped system architecture. There exists a central middle platform which synchronizes and coordinates the simulation processes in the subsystems connected with it. The simulation system also supports multiple simulation sessions and multiple client sessions. Thus, the question of the most efficient method for balancing the workload in the subsystems to maximize system throughput is open. A workload scheduling module should be added in the central middle platform.

- As a modeling environment providing reliable services for epidemic planning and response strategy studies, a fault recovery module needs to be considered. Many possible software or hardware errors could happen in Indemics. It is challenging to enumerate all the errors that could seriously affect the system reliability, and then propose their recovery solutions. This dissertation does not cover this aspect.

- One of the interesting questions raised in epidemic planning and response strategy is the search for optimal interventions to control a given disease outbreak. Although this dissertation does not discuss this topic, we believe that an optimal intervention exploration system could be extended from Indemics. Indemics provides an intervention simulation and situation assessment service. An optimal intervention exploration system would analyze the situation assessments and suggest the intervention strategies to be applied in order to control disease propagation.

Bibliography

[1] D. J. Abadi, S. Madden, and M. Ferreira. Integrating compression and execution in column-oriented database systems. In *ACM SIGMOD International Conference on Management of Data*, pages 671–682, 2006.

[2] D. J. Abadi, S. Madden, and N. Hachem. Column-tores vs. row-stores: how different are they really? In *ACM SIGMOD International Conference on Management of Data*, pages 967–980, 2008.

[3] D. J. Abadi, D. S. Myers, D. J. DeWitt, and S. Madden. Materialization strategies in a column-oriented DBMS. In *IEEE International Conference on Data Engineering*, pages 466–475, 2007.

[4] M. Antonioletti, M. P. Atkinson, N. P. C. Hong, B. Dobrzelecki, A. C. Hume, M. Jackson, K. Karasavvas, A. Krause, J. M. Schopf, T. Sugden, and E. Theocharopoulos. Grid enabling your data resources with OGSA-DAI. In *International Conference on Applied Parallel Computing*, pages 799–808, 2006.

[5] N. Bailey. *The Mathematical Theory of Infectious Diseases and Its Applications*. Hafner Press, New York, 1975.

[6] C. L. Barrett, K. R. Bisset, J. Chen, S. G. Eubank, B. Lewis, V. S. A. Kumar, M. V. Marathe, and H. S. Mortveit. *Interactions among human behavior, social networks, and societal infrastructures: A Case Study in Computational Epidemiology*. Springer Netherlands, 2009.

[7] C. L. Barrett, K. R. Bisset, S. G. Eubank, X. Feng, and M. V. Marathe. Episimdemics: an efficient algorithm for simulating the spread of infectious disease over large realistic social networks. In *International Conference for High Performance Computing Networking, Storage, and Analysis*, pages 1–12, 2008.

[8] C. L. Barrett, K. R. Bisset, J. Leidig, A. Marathe, and M. V. Marathe. An integrated modeling environment to study the co-evolution of networks, individual behavior and epidemics. *AI Magazine*, 31(1):75–87, 2010.

[9] C. L. Barrett, K. R. Bisset, J. Leidig, A. Marathe, and M. V. Marathe. Economic and social impact of influenza mitigation strategies by demographic class. *Epidemics*, 3(1):19–31, 2011.

[10] C. L. Barrett, S. G. Eubank, and M. V. Marathe. An interaction-based approach to computational epidemiology. In *AAAI Conference on Artificial Intelligence*, pages 1590–1593, 2008.

[11] N. E. Basta, D. L. Chao, M. E. Halloran, L. Matrajt, and I. M. L. Jr. Strategies for pandemic and seasonal influenza vaccination of schoolchildren in the United States. *American Journal of Epidemilogy*, 170:679–686, 2011.

[12] J. Becla and D. L. Wang. Lessons learned from managing a petabyte. In *Biennial Conference on Innovative Data Systems Research*, pages 70–83, 2005.

[13] G. Bell, J. Gray, and A. Szalay. Petascale computational systems. *IEEE Computer*, 39:110–112, 2006.

[14] K. R. Bisset, J. Chen, X. Feng, A. Kumar, and M. V. Marathe. EpiFast: a fast algorithm for large scale realistic epidemic simulations on distributed memory systems. In *International Conference on Supercomputing*, pages 430–439, 2009.

[15] K. R. Bisset, J. Chen, X. Feng, Y. Ma, and M. V. Marathe. Indemics: an interactive data intensive framework for high performance epidemic simulation. In *International Conference on Supercomputing*, pages 233–242, 2010.

[16] K. R. Bisset, X. Feng, M. V. Marathe, and S. M. Yardi. Modeling interaction between individuals, social networks and public policy to support public health epidemiology. In *Winter Simulation Conference*, pages 2020–2031, 2009.

[17] K. M. Carley, D. B. Fridsma, E. Casman, A. Yahja, N. Altman, L.-C. Chen, B. Kaminsky, and D. Nave. Biowar: Scalable agent-based model of bioattacks. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 36(2):252–265, 2006.

[18] S. Cauchemez, A. Bhattarai, T. L. Marchbanks, R. P. Fagan, S. Ostroff, N. M. Ferguson, D. Swerdlow, and the Pennsylvania H1N1 working group. Role of social networks in shaping disease transmission during a community outbreak of 2009 H1N1 pandemic influenza. *Proceedings of the National Academy of Sciences*, 108(7):2825–2830, 2011.

[19] Centers for Disease Control and Prevention. CDC H1N1 flu report, 2009. `http://www.cdc.gov/h1n1flu/`.

[20] D. L. Chao, M. E. Halloran, V. Obenchain, and I. M. Longini Jr. FluTE, a publicly available stochastic influenza epidemic simulation model. *PLoS Computational Biology*, 6(1):e1000656, 2010.

[21] S. Deodhar, K. R. Bisset, J. Chen, Y. Ma, and M. V. Marathe. Enhancing user-productivity and capability through integration of distinct software in epidemiological systems. In *ACM SIGHIT International Health Informatics Symposium*, pages 171–180, 2012.

[22] D. DeWitt and J. Gray. Parallel database systems: the future of high performance database systems. *Communications of the ACM*, 35(6):85–98, 1992.

[23] S. G. Eubank. Scalable, efficient epidemiological simulation. In *ACM symposium on Applied computing*, pages 139–145, 2002.

[24] S. G. Eubank, H. Guclu, V. S. A. Kumar, M. V. Marathe, A. Srinivasan, Z. Toroczkai, and N. Wang. Modelling disease outbreaks in realistic urban social networks. *Nature*, 4:180–184, 2004.

[25] N. M. Ferguson, D. A. T. Cummings, S. Cauchemez, C. Fraser, S. Riley, A. Meeyai1, S. Iamsirithaworn, and D. S. Burke. Strategies for containing an emerging influenza pandemic in Southeast Asia. *Nature*, 437:209–214, 2005.

[26] N. M. Ferguson, D. A. T. Cummings, C. Fraser, J. C. Cajka, P. C. Cooley, and D. S. Burke. Strategies for mitigating an influenza pandemic. *Nature*, 442:448–452, 2006.

[27] H. V. Fineberg and M. E. Wilson. Epidemic science in real time. *Science*, 324:987, 2009.

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

[29] D. Gilleskie. A dynamic stochastic model of medical care use and work absence. *Econometrica*, pages 1–45, 1998.

[30] T. H. Gordon Bell and A. Szalay. Beyond the data deluge. *Science*, 323:1297–1298, 2009.

[31] J. Gray, D. T. Liu, M. Nieto-Santisteban, A. Szalay, D. J. DeWitt, and G. Heber. Scientific data management in the coming decade. *ACM SIGMOD Record*, 34:34–41, 2005.

[32] J. Gray and A. Szalay. Where the rubber meets the sky: bridging the gap between databases and science. *IEEE Data Engineering Bulletin*, 27:3–11, Jan. 2006.

[33] E. Halloran, N. M. Ferguson, S. Eubank, J. Ira M. Longini, D. A. T. Cummings, B. Lewis, S. Xu, C. Fraser, A. Vullikanti, T. C. Germann, D. Wagener, R. Beckman, K. Kadau, C. Barrett, C. A. Macken, D. S. Burke, and P. Cooley. Modeling targeted layered containment of an influenza pandemic in the United States. *Proceedings of the National Academy of Sciences*, 105:4639–4644, 2008.

[34] S. Harizopoulos, V. Liang, D. J. Abadi, and S. Madden. Performance tradeoffs in read-optimized databases. In *International Conference on Very Large Databses*, pages 487–498, 2006.

[35] G. Heber and J. Gray. Supporting finite element analysis with a relational database backend; part II: Database design and access. *Microsoft Technique Report*, MSR-TR-2006-21, 2005.

[36] H. W. Hethcote. The mathematics of infectious diseases. *Society for Industrial and Applied Mathematics Review*, 42(4):599–653, 2000.

[37] M. D. V. Kerkhovea and N. M. Ferguson. Epidemic and intervention modelling - a scientific rationale for policy decisions? Lessons from the 2009 influenza pandemic. *Bulletin World Health Organization*, 90:306–310, 2012.

[38] W. O. Kermack and A. G. McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London. Series A*, 115(772):700–721, 1927.

[39] M. Kretzschmar and R. T. Mikolajczyk. Contact profiles in eight European countries and implications for modelling the spread of airborne infectious diseases. *PLoS ONE*, 4:e5931, 2009.

[40] Y. Kuznetsov and C. Piccardi. Bifurcation analysis of periodic SEIR and SIR epidemic models. *Journal of Mathematical Biology*, 32:109–121, 1994.

[41] D. B. L. Hufnagel and T. Geisel. Forecast and control of epidemics in a globalized world. *Proceedings of the National Academy of Sciences*, 101:15124–15129, 2004.

[42] A. Lamb, M. Fuller, R. Varadarajan, N. Tran, B. Vandier, L. Doshi, and C. Bear. The vertica analytic database: C-Store 7 years later. *Proceedings of the VLDB Endowment*, 5(12):1790–1801, 2012.

[43] S. Liao, J. Chen, Y. Ma, and A. Marathe. Paid sick-leave: Is it a good way to control epidemics? In *International Conference on Complex Sciences: Theory and Applications*, 2012.

[44] A. Marathe, B. Lewis, C. Barrett, J. Chen, M. Marathe, S. Eubank, and Y. Ma. Comparing effectiveness of top-down and bottom-up strategies in containing influenza. *PLoS ONE*, 6:e25149, 2011.

[45] G. J. Milne, J. K. Kelso, H. A. Kelly, S. T. Huband, and J. McVernon. A small community model for the transmission of infectious diseases: Comparison of school closure as an intervention in individual-based models of an influenza pandemic. *PLoS ONE*, 3:e4005, 2008.

[46] S. S. Morse, R. L. Garwin, and P. J. Olsiewski. Next flu pandemic: What to do until the vaccine arrives? *Science*, 314(5801):929, 2006.

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

[48] J. M. Read, K. T. Eames, and W. J. Edmunds. Dynamic social networks and the implications for the spread of infectious disease. *Journal of Royal Society Interface*, 5(26):1001–1007, 2008.

[49] L. A. Rvachev and I. M. Longini. A mathematical model for the global spread of influenza. *Mathematical Biosciences*, 17:3–22, 1985.

[50] Y. Simmhan, R. Barga, C. van Ingen, M. A. Nieto-Santisteban, L. Dobos, N. Li, M. Shipway, A. S. Szalay, S. Werner, and J. Heasley. Graywulf: Scalable software architecture for data intensive computing. In *Hawaii International Conference on System Sciences*, pages 1–10, 2009.

[51] J. Skåtun. Take some days off, why don't you? Endogenous sick leave and pay. *Journal of Health Economics*, 22(3):379–402, 2003.

[52] T. Sterling, J. E. D. D. J. Beckerand D. Savarese, U.A.Ranawake, and C. V. Packer. Beowulf: A parallel workstation for scientific computation. In *International Conference on Parallel Processing*, pages 11–14, 1995.

[53] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. J. O'Neil, P. E. O'Neil, A. Rasin, N. Tran, and S. B. Zdonik. C-Store: A column-oriented DBMS. In *International Conference on Very Large Databses*, pages 553–564, 2005.

[54] P. Stroud, S. Del Valle, S. Sydoriak, J. Riese, and S. Mniszewski. Spatial dynamics of pandemic influenza in a massive artificial society. *Journal of Artificial Societies and Social Simulation*, 10(4):9, 2007.

[55] A. Szalay and J. Gray. Science in an exponential world. *Nature*, 7083:413–414, 2006.

[56] C. Taylor, A. Marathe, and R. Beckman. The economic impact of pandemic influenza in the United States: priorities for intervention. *Emerging Infectious Diseases*, 5(5), 1999.

[57] C. Taylor, A. Marathe, and R. Beckman. Same influenza vaccination strategies but different outcomes across US cities? *International Journal of Infectious Diseases*, 14(9):e792 – e795, 2010.

[58] C. Viboud, O. N. Bjornstad, D. L. Smith, L. Simonsen, M. A. Miller, and B. T. Grenfell. Synchrony, waves, and spatial hierarchies in the spread of influenza. *Science*, 312(5772):447–451, 2006.

[59] E. Vynnycky and R. G. White. *An Introduction to Infectious Disease Modelling*. Oxford University Press, 2010.

[60] J. Wallinga, M. Boven, and M. Lipsitch. Optimizing infectious disease interventions during an emerging epidemic. *Proceedings of the National Academy of Sciences*, 107(2):923–928, 2010.

[61] J. Wallinga, W. Edmunds, and M. Kretzschmar. Perspective: human contact patterns and the spread of airborne infectious diseases. *Trends in Microbiology*, 9:372–377, 1999.

[62] G. Wang, M. A. V. Salles, B. Sowell, X. Wang, T. Cao, A. J. Demers, J. Gehrke, and W. M. White. Behavioral simulations in MapReduce. *Proceedings of the VLDB Endowment*, 3(1):952–963, 2010.

[63] W. M. White, A. J. Demers, C. Koch, J. Gehrke, and R. Rajagopalan. Scaling games to epic proportion. In *ACM SIGMOD International Conference on Management of Data*, pages 31–42, 2007.

[64] World Health Organization. WHO consultation on priority public health interventions before and during an influenza pandemic, 2004. `http://www.afro.who.int/index.php?option=com_docman&task=doc_download&gid=5116`.

[65] J. T. Wu, S. Riley, C. Fraser, and G. M. Leung. Reducing the impact of the next influenza pandemic using household-based public health interventions. *PLoS Medicine*, 3(9):e361, 08 2006.