Computational Framework for Uncertainty Quantification, Sensitivity Analysis and Experimental Design of Network-based Computer Simulation Models

Sichao Wu

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

Madhav V. Marathe, Chair
Henning S. Mortveit, Co-chair
Yang Cao
Xinwei Deng
S. S. Ravi
Anil Kumar S. Vullikanti

July 26, 2017
Blacksburg, Virginia

Keywords: Network-based Simulation Models, Uncertainty Quantification, Sensitivity Analysis, Experimental Design, Graph Dynamical Systems

Computational Framework for Uncertainty Quantification, Sensitivity Analysis and Experimental Design of Network-based Computer Simulation Models

Sichao Wu

(ABSTRACT)

When capturing a real-world, networked system using a simulation model, features are usually omitted or represented by probability distributions. *Verification and validation* (V & V) of such models is an inherent and fundamental challenge. Central to V & V, but also to model analysis and prediction, are *uncertainty quantification* (UQ), *sensitivity analysis* (SA) and *design of experiments* (DOE). In addition, *network-based computer simulation models*, as compared with models based on ordinary and partial differential equations (ODE and PDE), typically involve a significantly larger volume of more complex data. Efficient use of such models is challenging since it requires a broad set of skills ranging from domain expertise to in-depth knowledge including modeling, programming, algorithmics, high-performance computing, statistical analysis, and optimization. On top of this, the need to support reproducible experiments necessitates complete data tracking and management. Finally, the lack of standardization of simulation model configuration formats presents an extra challenge when developing technology intended to work across models. While there are tools and frameworks that address parts of the challenges above, to the best of our knowledge, none of them accomplishes all this in a model-independent and scientifically reproducible manner.

In this dissertation, we present a computational framework called GENEUS that addresses these challenges. Specifically, it incorporates (i) a standardized model configuration format, (ii) a data flow management system with digital library functions helping to ensure scientific reproducibility, and (iii) a model-independent, expandable plugin-type library for efficiently conducting UQ/SA/DOE for network-based simulation models. This framework has been applied to systems ranging from fundamental *graph dynamical systems* (GDSs) to large-scale socio-technical simulation models with a broad range of analyses such as UQ and parameter studies for various scenarios. Graph dynamical systems provide a theoretical framework for network-based simulation models and have been studied theoretically in this dissertation. This includes a broad range of stability and sensitivity analyses offering insights into how GDSs respond to perturbations of their key components. This stability-focused, structure-to-function theory was a motivator for the design and implementation of GENEUS.

GENEUS, rooted in the framework of GDS, provides modelers, experimentalists, and research groups access to a variety of UQ/SA/DOE methods with robust and tested implementations without requiring them to necessarily have the detailed expertise in statistics, data management and computing. Even for research teams having all the skills, GENEUS can significantly increase research productivity.

Computational Framework for Uncertainty Quantification, Sensitivity Analysis and Experimental Design of Network-based Computer Simulation Models

Sichao Wu

(GENERAL AUDIENCE ABSTRACT)

Uncertainties are ubiquitous in computer simulation models especially for network-based models where the underlying mechanisms are difficult to characterize explicitly by mathematical formalizations. Quantifying uncertainties is challenging because of either the lack of knowledge or their inherent indeterminate properties. Verification and validation of models with uncertainties cannot include every detail of real systems and therefore will remain a fundamental task in modeling. Many tools are developed for supporting uncertainty quantification, sensitivity analysis, and experimental design. However, few of them is domain-independent or supports the data management and complex simulation workflow of network-based simulation models.

In this dissertation, we present a computational framework called GENEUS, which incorporates a multitude of functions including uncertain parameter specification, experimental design, model execution management, data access and registrations, sensitivity analysis, surrogate modeling, and model calibration. This framework has been applied to systems ranging from fundamental *graph dynamical systems* (GDSs) to large-scale socio-technical simulation models with a broad range of analyses for various scenarios. GENEUS provides researchers access to uncertainty quantification, sensitivity analysis and experimental design methods with robust and tested implementations without requiring detailed expertise in modeling, statistics, or computing. Even for groups having all the skills, GENEUS can help save time, guard against mistakes and improve productivity.

Dedication

To the loving memory of my Grandmother Qiuhua Wu

Acknowledgments

Behind the successful completion of my Ph.D. study, there is support from many people. I would like to express my sincerest gratitude to them and many others whose names are not all enumerated.

First of all, I am extremely grateful to my advisor Dr. Henning Mortveit, for his constant patience, insightful advice, persistent encouragement and professional mentorship throughout my academic endeavors for the last five years. I cannot remember how much time he has spent on discussing with me on every detail of my work and helping me developing presentation and writing skills. Dr. Mortveit is a rigorous researcher, responsible mentor and concerning friend. It is my honor and fortune to be one of his students. I would like to thank Dr. Madhav Marathe, the co-chair of my committee, who introduced me to the cutting-edge research opportunities and provided keen research insight during my pursuit of the degree. I would like to thank the rest of my committee: Dr. Yang Cao, Dr. Xinwei Deng, Dr. S. S. Ravi and Dr. Anil Vullikanti for their invaluable discussion, suggestions, and inputs.

I acknowledge and appreciate the support of my colleagues in the Network Dynamics and Simulation Science Laboratory (NDSSL). I would like to address my special thanks to Dr. Abhijin Adiga, who has worked closely with me from the theoretical work to various research projects. He offered thoughtful ideas on our work, which plays significant roles in our co-authored paper. I thank all the lab mates and collaborators in NDSSL including Dr. Jiangzhuo Chen, Dr. Stephen Eubank, Hilton Galyean, Dr. Sandeep Gupta, Dr. Christopher Kuhlman, Michael Levet, Dr. Bryan Lewis, Dr. Achla Marathe, Joseph Mcnitt, Dr. Eric Nordberg, Dr. Yihui Ren, Dr. Samarth Swarup, Dr. Srinivasan Venkatramanan, and Dawen Xie.

In addition, I would like to thank my fellow graduate students: Maksudul Alam, Hasanuzzaman Bhuiyan, Jose Cadena, Shuyu Chu, Arindam Fadikar, S M Shamimul Hasan, Wei Huang, Qijun He, Fang Liu, Yifei Ma, Nidhi Parikh, Guanhong Pei, Gaurav Tuli, Huadong Xia, Jae-Seung Yeom, Ming Yi, Yao Zhang, for making my stay in NDSSL and Virginia

Tech more enjoyable. I thank the NDSSL, Biocomplexity Institute and Computer Science Department staffs for their support: Erin, Sandi, Sharon, Katie, Sarah and many others. I am also intellectually indebted to my former Master degree advisor Dr. Chuandong Li, who has opened the academic gate for me.

Last but not the least, I would like to express my deepest gratitude to my family for their unconditional understanding and support. When I experienced my hardest time and overwhelmed with frustrations, my mother was always there offering trust and encouragement. Without her faithful love, I would not have been able to stand at this place. I would like to express special thanks to my girlfriend Xiangyu Gao. Without your company, I would have graduated sooner, but with much less enjoyable memories. Finally, I wish my loving grandmother rest in peace. "Twinkling star in the sky, ever bright and peaceful shine." (*Demi-Gods and Semi-Devils*)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Overview

Uncertainties are pervasive in simulation models regardless of their underlying mathematical formalism or actual implementation due to the incomplete knowledge, inexact initial and boundary conditions, and randomness of model parameters. In order to guarantee the quality of models, it is essential to understand how well the real system is captured, what the influential parameters are, how uncertainties propagate trough the model and how model discrepancies can be reduced. This understanding can be gained from model *verification and validation* (V & V) which involves the assessment of the fact that the model implemented is actually what one intended to implement (verification) and assessment that the model mimics the system it was intended to capture (validation) [45].

Earlier, many of the computer simulation models were concerned with numerical methods (e.g. solving differential equations), focusing largely on regular structures with more uniform discretization or grids [19, 39, 114]. Network-based computer simulation models, on the other hand, involve highly irregular structures. Typically, they will represent the interactions of the modeled systems in a direct manner using a graph where edges encode actions and interactions among agents which are represented by vertices. A formal basis for network-based simulation models is *graph dynamical systems* (GDSs). GDSs are discrete dynamical systems and are sometimes referred to as *generalized cellular automata*. A GDS can be represented by a 4-tuple (G, K, F, W) consisting of a graph $G(V, E)$ with vertex set V and edge set E; a vertex state space K; a set of vertex functions F governing the state transition of each vertex; and an update scheme W orchestrating the ordering in which the vertex functions are executed. GDS serves as a formal framework of network-based simulation models. An important theme in the research of discrete dynamical systems is

examining the robustness of system dynamics with respect to perturbations [16, 17, 58, 60, 100, 135, 148]. Perturbations can exist in every component of GDS including graph structure, vertex function, vertex state and update scheme. Usually, their forms and effects are uncertain. Mathematical, computational and algorithmic analysis of the stability of GDS framework under uncertain perturbations will stand as a theoretical foundation of the validation of actual network-based simulation models.

Generally speaking, model validation techniques for all models, whether equation-based or network-based, are the same. The goal is to ensure that the model as implemented is indeed the model that was designed and intended. A simulation model can be represented as follows:

$$y = f(x_1, \ldots, x_p; \theta_1, \ldots, \theta_q) = f(X; \theta),\ X \in \mathcal{T},\ \theta \in \mathcal{S}, \tag{1.1}$$

where $X = (x_1, \ldots, x_p)$ is the model input, $\theta = (\theta_1, \ldots, \theta_q)$ is the model parameter, y is an output variable, f is the system function that may or may not have an explicit analytic formula, $\mathcal{T}$ and $\mathcal{S}$ are the model input space and model parameter space, respectively. For computer simulations, any model represented by Equation 1.1 is usually regarded as a "black box", that is to say, the actual mathematical form is unknown (structural uncertainties). Knowledge about model inputs X and model parameters θ can also be incomplete or imprecise (input uncertainties and parameter uncertainties). *Uncertainty quantification* (UQ) is the process that captures all these uncertain factors of the model, propagates them through the model and represents them adequately in the simulation outcomes [103]. *Sensitivity analysis* (SA) in the context of UQ studies how uncertainties of the model output can be apportioned among the inputs. Both of them are important components of model verification and validation [138]. Since the scale of modern network-based models grows fast with increasing computational demand, a concrete *design of experiment* (DOE) with economic size is essential to improve the efficiency of UQ/SA analysis, save computational time and resources [55, 141]. As a matter of fact, model validation including uncertainty quantification, sensitivity analysis, and experimental design will last throughout the lifetime of the model.

By nature, network-based simulation models often require large data sets as well as a large variety of datasets; these requirements place heavy demands on both data management and computation. As an example, an infectious disease simulation experiment for the United States uses a synthetic population network with 280 million nodes [167]. Naturally, the data volume will continue to increase as infrastructure systems such as transportation and power grid models are integrated for more comprehensive analyses. Experiments involving data sets of this scale will bring even greater challenges to data management and

computing infrastructures.

When performing model validation, the challenges will be intensified since it is usually accompanied by experimental designs involving a large number of cells, each with multiple replicates. Management of a large number of experiment instances along with the associated input/output/configurations is nontrivial. Moreover, it is often desirable that model validation can be done by a third-party in addition to or rather than by the modelers themselves to guarantee the credibility, a process sometimes referred to as independent verification and validation (IV&V) [45]. For this, it is highly desirable to ensure the experimental results are reproducible. However, the involved process of conducting such analyses complicates matters. There is a lack of standardized model configurations as well as ways to actually invoke the simulation models, which makes it challenging to even execute the workflows, not even considering the possible statistical designs involved. The challenge is compounded by the need to integrate domain experts (who typically lack modeling and computing skills), and the needs for advanced expertise covering programming, high-performance computing, statistical analysis, and data management.

Motivated by above-mentioned aspects, in this dissertation, we focus on three major research themes: First, we designed and developed a computational framework GENEUS, which is a comprehensive environment supporting simulation model execution, experimental designs, data management, uncertainty quantification and sensitivity analysis for a variety of network-based simulation models. Second, we explored stability theory for a collection of graph dynamical systems with respect to uncertain perturbations to every GDS component. This forms our major theoretical research contribution. Last but not least, we conducted UQ/SA/DOE studies in several application scenarios by using GENEUS. These case studies involve simulation models with different forms, computational scales and data requirements. We showed that GENEUS can meet various application requirements, facilitate model validation procedures, reduce manual effort and improve human productivity.

1.2 Motivating Study

In 2012, we conducted the sensitivity analysis of a simulation model capturing the transportation system in a large urban setting. The work that we present in [2] is a part of a project employing disaggregated network-based simulation methods to model the collective human behavior in the aftermath of a large-scale nuclear detonation in Washington D.C. The entire simulation model, where the transportation module is one of the components, seeks to answer questions such as how human behavior and various policies affect the number of lives saved.

Validation of such a model is highly challenging for several reasons. First, it covers many aspects such as initial data, ability to reproduce known outcomes, and preservation of functional invariants. Since this is a hypothetical scenario, it hard to fully reproduce the ground truth data. However, it is still useful to investigate the model parameters which can help understand the model itself. In [2], we focus on how the ambient traffic density parameters affect the travel times and route choices of the individuals of the population in our model. These parameters are not easily estimated, particularly in the given context, and they directly influence travel times and routes which in turn impact the health and behavior of the individuals and vice versa. Second, this is a data and computationally intensive simulation system. The simulations were run on a large 60 node multi-core cluster. One complete run (around 120 iterations) takes about 35 hours and requires a few terabytes (TB) of space. Therefore, a full factorial design is not feasible for simulations of such a scale. Third, management of the simulation workflow and manipulating the simulation data are nontrivial tasks even for those who developed the model. It is highly challenging for other researchers who are not familiar with the simulation model and data specification to repeat the experiment.

These difficulties we faced in this study actually motivated our work. In the next few sections, we summarize the research challenges and several major research questions that will be covered in this thesis.

1.3 Challenges

Validation of network-based simulation model is challenging both in theory and in practice. In method, UQ/SA/DOE techniques are still under development; in theory, stability and validation of the mathematical model of the network-based model is generally hard; in practice, performing UQ/SA/DOE and model validation is non-trivial. The major challenges of our research are summarized as follows.

1.3.1 Technical Challenges in Uncertainty Quantification.

Modeling uncertainty is challenging.
Generally speaking, there are two types of uncertainties: aleatory uncertainty (statistical uncertainty) and epistemic uncertainty (systematic uncertainty), where the former is owing to the natural randomness in a model and the latter is due to lack of knowledge. How to characterize different model uncertainties appropriately is non-trivial because of the limited information available and insufficient measurements. When a large number of

uncertain parameters are involved and the correlation structure is complex, it is difficult to model the uncertainties precisely.

Propagating uncertainty is challenging.
Propagation of uncertainties often involves a large number of simulation runs. Many simulation models are complex to configure and computationally demanding. How to manage the experimental design, simulation workflow and the associated data sets are challenging.

Identifying quantities of interest (QoI) is challenging.
UQ and SA are with respect to a specific quantity of interest (QoI), which is a measurand of simulation results. However, in many cases, QoIs are not directly observable from raw simulation outcomes. Identifying an appropriate QoI and extracting it from the output data are crucial but challenging tasks.

Developing uncertainty quantification theory is challenging.
A lot of theoretical techniques have been developed for modeling uncertainties, propagating uncertainties and quantitatively evaluating uncertainties . However, mathematical, algorithmic and computational techniques for uncertainty quantification are still evolving and there exist many open questions.

1.3.2 Theoretical Challenges in Stability of Graph Dynamical Systems.

Rigorous stability analysis is challenging.
Many stability theories are for finite discrete dynamical systems with particular graph structure, node function, and update scheme. However, it is hard to determine the system dynamics for a GDS in general. The effects of perturbations to GDS components is still not well understood.

Exploring the phase space efficiently is challenging.
Brute-force computation of the phase space of graph dynamical system is not feasible in most scenarios. Computational complexity results suggest that such an exploration is unlikely to exist.

1.3.3 Application Challenges in UQ/SA/DOE.

UQ/SA/DOE studies are computationally challenging.
Large network-based simulation models usually involve tremendous amount of data and require extensive computing resources. How to manage the complex simulation workflow

and efficiently executing the simulation model is challenging.

Performing UQ/SA/DOE in practice is challenging.
It requires knowledge including statistical analysis, simulation model configuration, programming for *high-performance computing* (HPC), data management, domain expertise and so on. It is challenging especially for those without a full set of required skills. In addition, the lack of standardization of data formats, configuration specifications, model invocation and execution mechanisms make it hard to generalize the simulation and experimentation workflow.

1.4 Research Questions and Objectives

In the following subsections, we articulate the major research questions and objectives.

1.4.1 Computational Framework for UQ/SA/DOE

There are many types of tools for supporting uncertainty quantification, sensitivity analysis, experimental design and other V & V related studies [1, 160, 163]. Some tools are specific to a particular domain and most of them were designed for models based on ordinary and partial differential equations (ODE and PDE). ODE and PDE based models are ubiquitous in all sciences and their applications. However, with the increase in computational power during the recent decades [79], network-based models have become a very viable alternative. The question is:

- Can we design and implement a domain-independent, extensible, easy-to-use computational framework for supporting uncertainty quantification, sensitivity analysis and experimental design of network-based simulation models?

To address this question, our research goal is to design and construct a computational model validation framework that possesses the following properties: (i) it can accommodate a variety of network-based simulation models in different application domains, (ii) the framework can handle large-scale simulation workflow including model execution, big data management, efficient analysis, etc., (iii) the framework is supposed to be loosely-coupled such that it can be extended by adding new functions, algorithmic implementations and simulation models, (iv) it has clearly defined interfaces such that the startup time for using the software and the required computing knowledge are minimal.

1.4.2 Validation and Stability of Graph Dynamical Systems

At a core mathematical level, the notion of sensitivity with respect to perturbation and initial condition is a key for validation. Naturally, sensitivity can be addressed for each system component for a given fixed initial condition: (i) perturbation of the vertex states, (ii) perturbation of the vertex functions, (iii) perturbation with respect to an ensemble of networks and (iv) perturbation with respect to an ensemble of update schemes. A central question for validation in all these cases is the robustness of the system dynamics under perturbations. We will address following research questions:

- What are the criteria for a specific graph dynamical system that lead to only fixed points as its limit cycles? How do update scheme and network structure affect the limit cycle structure? How to quantify the sensitivity with respect to the node state perturbations? Can we calibrate the parameters of one or multiple GDS component to achieve the desired system dynamics?

A prerequisite for properly addressing model validation of network-based simulation models is a precise formal framework in which they can be represented. Graph dynamical system serves as such a mathematical framework. Therefore, developing a theory for the purpose of answering the questions above is crucial to address the model validation network-based simulation models.

1.4.3 Validation of Network-based Simulation Models

Network-based simulation models have been widely used in many urban computation scenarios [2, 6, 126, 136]. Following are some related research questions of such kind simulation models:

- What are the most influential model input parameters? How do input uncertainties propagate through the simulation model and affect the output? Is there a way to efficiently explore the parameter space that saves computing resources without losing representativeness?

We address these questions in different application scenarios: (i) devise the most effective intervention strategies for containing a pandemic outbreak, and quantify the uncertainties in a disease model; (ii) develop a network-based approach for modeling the seasonal flow of agricultural produce and examine its role in the dispersal of pest spread.

1.5 Research Contributions

1.5.1 GENEUS Software Tool

What is GENEUS

GENEUS is a software package with a multitude of functions including uncertain parameter specification, experimental design, model execution management, data access and registrations, sensitivity analysis, surrogate modeling, model calibration and so on. It is domain-independent in that it can accommodate various simulation models with different data, configuration and execution requirements ranging from theoretical graph dynamical systems to practical large-scale simulation models. It handles the model execution and data management issues in a transparent manner so that the domain experts do not need to concern themselves with HPC and big data management issues. With an environment like GENEUS, one can contribute robust and tested implementations for statistical analysis. Otherwise, a team would be challenged to do this and possibly be prone to faulty algorithms and implementations. Specifically, it has following features:

- By design, GENEUS supports a large number of simulation models through the introduction of a standardized configuration format and streamlines the model execution workflow though its model wrapper component.
- It integrates a large number of experimental designs, including adaptive designs, as well as many UQ/SA methods though its plug-in design.
- GENEUS incorporates a standardized specification of DOE/SA/UQ; thus it supports analysis in a model-independent manner.
- The GENEUS registry records all simulation data dependencies and supports scientifically reproducible experiments.
- The registry has a built-in management system with digital library functions for systematically exploring simulation outcomes and cells of experimental designs.
- GENEUS is a flexible system that can function in a standalone manner through its server and can also be embedded within other applications.
- GENEUS is extensible and allows developers/domain experts to contribute to its UQ/SA/DOE algorithm library.

- GENEUS provides domain experts and other researchers lacking the full set of skills (e.g., modeling, computing, data management, and statistical analysis) robust access to its UQ/SA/DOE methods in support of for example model validation.

Who will use GENEUS?

Any individual modeler, experimentalist, or research group can use GENEUS to design experiments, quantify uncertainties, and assess model parameters for network-based computer simulation models. GENEUS provides researchers access to UQ/SA/DOE methods with robust and tested implementations without requiring detailed expertise in modeling, statistics, or computing. Even for groups having all the skills, GENEUS can help save time, guard against mistakes and improve productivity.

What is the state-of-the-art?

Existing tools including DAKOTA, PSUADE, and UQ-PyL are designed for traditional equation-based simulation models which often involve smaller data sets than the network-based simulation models. DAKOTA [1] provides comprehensive parameter studies, uncertainty quantification, and optimization algorithms running on supercomputer platforms. PSUADE [160] is a C++ based open-source software package. It runs on Linux-like systems only through command line interfaces. UQ-PyL [163] is a Python UQ tool-kit with a graphical user interface that improves the user experience. All of the above-mentioned systems incorporate comprehensive UQ/SA/DOE methods and solutions. We believe each individual software or a combination of them can facilitate the model validation life cycle. However, to the best of our knowledge, most of them lack a standardized model invocation mechanism, a data management scheme for handing large amount datasets in diverse formats, and support for repeatable experiments. Even if fundamental UQ/SA/-DOE methods are the same, big data and complex simulation workflow management are challenging tasks. Motivated by this, GENEUS is designed as a generic system that can accommodate a variety of computer simulation models through the standardized model configuration format and model specific wrapper code. It provides many UQ/SA/DOE methods and handles big data management and model execution in computer simulation experiments transparently. All these features make GENEUS a flexible and easy-to-use system.

Why GENEUS instead of other tools?

Comparing with existing tools, GENEUS has several distinguishing features. First, GENEUS generalizes the configuration format for network-based computer simulation models. By design, it has an XML-based model configuration format such that different simulation models can be initialized uniformly. A model wrapper will handle the actual configuration translation and model execution. This can reduce the chance of errors owing to the manual model configuration. It can also allow for automatic experimental design, model execution, and analysis. Second, GENEUS has a central registry system that records data dependencies of simulation models and experimental details such that experiments can be easily repeated. In addition, simulation outcomes are saved in the registry where they can be easily retrieved through the registry API, in this way, analysis can be performed independently of model execution. This could accelerate the research life cycle. Finally, GENEUS is a loosely-coupled extensible ecosystem: each component can function independently without awareness of other parts. Such design makes GENEUS easy to maintain and evolve. Methods and algorithms are implemented as plug-ins such that GENEUS UQ/SA/DOE library can be extended. All these features make GENEUS a distinguishing ecosystem that can facilitate UQ/SA/DOE workflow in the context of model validation.

1.5.2 Stability Theory of Graph Dynamical Systems.

As mentioned earlier, perturbations can be made to every component of graph dynamical system, and the consequences of the perturbations are uncertain. In this thesis, we derived several sufficient stability criteria with respect to perturbations to system components.

Dynamic bi-threshold system

We considered a generalized dynamic bi-threshold system, where the up- and down threshold values are mutable as the system dynamic evolves. The changeable threshold function can be viewed as a perturbation to the vertex functions. We proved that under certain conditions, the sequential systems only have fixed points as limit sets and the periodic orbits have maximal size 2 under the synchronous update method, regardless of how the up- and down-thresholds evolve.

Block-sequential threshold dynamical systems

We studied the attractor structure of standard block-sequential threshold dynamical systems. In a block-sequential update, the vertex set of the graph is partitioned into blocks, and the blocks are updated sequentially while the vertices within each block are updated in parallel. We studied how the underlying graph structure influences the limit cycle structure of block-sequential systems. We derived a sufficient condition on the graph structure so that the system has only fixed points as limit cycles. We also identify several well-known graph families that satisfy this condition.

Activity in Boolean networks

We study the perturbation to vertex states. We first extended the notion of *activity* [145] which measures the probability that a perturbation in an initial state produces a different successor state than that of the original unperturbed state. Then, we took into account a collection of actual graph structures and determined the activity for them.

1.5.3 Applications of GENEUS in Different Domains

To illustrate the design concept and usability of GENEUS, we explored different application scenarios and successfully incorporated various simulation models with GENEUS.

Modeling interventions for pandemics

We developed a large experimental design for testing the effect of different intervention strategies. In this study, we incorporated GENEUS with EpiFast [31], which is a stochastic simulation model for disease propagation over large contact networks.

Modeling spread of invasive species

We conducted sensitivity analysis for modeling the commodity flow in the context of invasive species spread in Nepal by using GENEUS. These case studies prove the usage of GENEUS and its application of UQ/SA/DOE in actual network-based simulations.

1.6 Dissertation Organization

The remainder of the dissertation is organized as follows. In Chapter 2, we further articulate some basic concepts and review the existing related research results. In Chapter 3, we present the design and technical details of GENEUS system. In Chapter 4, we present the theoretical results in stability of a collection of graph dynamical systems with respect to various perturbations. In Chapter 5, we show the application case studies by using GENEUS for model validation. Finally, we will conclude the dissertation with discussion and several open questions.

Chapter 2

Literature Review

2.1 Uncertainty Quantification, Sensitivity Analysis, and Experimental Design

Forward uncertainty propagation investigates how uncertain outputs propagate from the variability listed in the sources of input uncertainties. As Roy et al. state in [137] , "Uncertainty is classified as either (i) *aleatory* – the inherent variation in a quantity that, given sufficient samples of the stochastic process, can be characterized via a probability density distribution, or (ii) *epistemic* – uncertainty due to lack of knowledge by the modelers, analysts conducting the analysis, or experimentalists involved in validation." Many tools such as DAKOTA [1] uses probability distribution functions and interval analysis to capture aleatory uncertainties and epistemic uncertainties, respectively. Another widely adopted representation of input uncertainties is spectral methods such as *polynomial chaos expansions* (PCE) [52] which employ multivariate orthogonal polynomials to characterize particular input probability distributions. An important advantage of PCE is that it forms a functional relationship between responses and their uncertain inputs rather than estimating point probabilities. For many other methods of modeling input uncertainties, see [26, 27, 75] and references therein.

In [124], Owhadi et al. propose an *optimal uncertainty quantification* framework. They state, "given a set of assumptions and information about the problem, there exist optimal bounds on uncertainties: these are obtained as values of well-defined optimization problems corresponding to extremizing probabilities of failure, or of deviations, subject to the constraints imposed by the scenarios compatible with the assumptions and information." Such framework often involves extremely large problem space and propagating the input

uncertainties is a challenge. Uncertainty propagation often involves sampling the input parameter space according to particular probability distributions. Monte Carlo (random) method is straightforward but usually requires a large number of sample points to achieve a desired convergence rate, thus is not always feasible especially when the simulation model is computational demanding. Nowadays, computer experiments often adopt sparse space-filling sampling techniques that seek to make the samples spread uniformly within the parameter space with economic sample size [55]. As an example, *Latin Hypercube Sampling* (LHS) [112] is a popular method adopted by many experiment practitioners. It works for an arbitrary number of dimensions, where each sample point is the only one in each dimension. However, it is not guaranteed to provide good uniform properties in an n-dimensional hypercube, which is its major known drawback. Many variants of LHS can remedy this by adding more restrictions. For instance, orthogonal array based LHS [158] with strength r is able to achieve a uniformity over the r-dimensional sub-space. Other criteria-based spacing-filling designs are also developed, see [141] and its references. A more advanced experimental design technique known as the *adaptive design* [43, 44, 47, 164] (or active learning) uses information from previous experimental data. Adaptive design starts with an initial experimental design(e.g. LHS), then adaptively chooses samples in order to optimize particular criteria, e.g., finding most uncertain region within the parameter space. It usually constructs a *statistical surrogate model* (meta model) [43, 110] that can be used as a predictor of next sample points of interest. A surrogate model is also widely used as an alternative emulator of an expensive simulation [38, 77].

Sensitivity analysis (SA) measures how uncertainty in the output of a mathematical model or system (numerical or otherwise) can be apportioned to uncertainties in its inputs [74, 84, 97, 140]. SA can be divided into two groups: *local sensitivity analysis* and *global sensitivity analysis*. The local SA monitors the changes of the response by varying one parameter at a time while making other parameters constant, thereby it is also referred as *one-at-a-time* method (OAT/OFAT) [119]. The sensitivity is then measured by partial derivatives of the output Y with respect to the input parameter X_i : $|\frac{\partial Y}{\partial X_i}|$. On the other hand, the global SA investigates not only the individual parameter effect but also the interactions among them [78, 138, 140, 153]. This kind of sensitivity analysis decomposes the output variance in terms of input parameters and their increasing order of interactions as

$$V(X) = \sum_i V_i(x_i) + \sum_{i<j} V_{i,j}(x_i, x_j) + \cdots + V_{1\ldots s}(x_1, \ldots, x_s). \tag{2.1}$$

Note that this variance decomposition method can evaluate the influence of individual parameters (first term right-hand side) as well as higher order interacting effects (remaining terms). However, it is worth noting that effects of high order interactions are neglected in many situations since they are either insignificant or computationally expensive to

evaluate.

Given experimental results of simulations, *inverse uncertainty quantification* estimates the discrepancy between computational models (*bias correction*) and real systems; it also estimates and reduces parameter uncertainties (*parameter calibration*) [161]. Generally speaking, this is a harder task than forward uncertainty propagation because it requires ground truth data, which is often limited or unavailable. On the other hand, inverse problems often need an iterative process to update the analysis results according to the new incoming information. In *Bayesian calibration* [77, 95], uncertainties in input parameters are captured by a "prior" distribution. If priors are updated when physical experiment results or historical ground truth data are available, then one obtains a posterior distribution after the calibration process. The calibrated model is expected to be more credible and accurate.

Uncertainty quantification, sensitivity analysis and design of experiment remain core methodologies in model verification and validation and is a very active research area with many open questions.

2.2 Graph Dynamical Systems

Graph dynamic system (GDS) is a finite discrete dynamical system that generalizes the concept of *Boolean networks* (BN) [92, 135, 147] and *cellular automata* (CA) [64, 83, 165]. It is designed precisely to capture relationships or interactions (graph edges) between agents(graph vertices) of network-based simulation models and to allow for rigorous formal analysis [117]. Research and applications of graph dynamical systems are often compared with models based on ordinary and partial differential equation (ODE and PDE). ODE- and PDE-based models are ubiquitous in all sciences and their applications. However, with the increase in computational power during the last decades [80, 167], graph dynamical system models have become a very viable alternative. In particular, for large network-based systems, graph dynamical systems represent a natural, and perhaps more suited approach for modeling and analysis.

There is often a split between *finite* and *infinite* graph dynamical system. The infinite case is concerned with the systems where the network is infinite, examples include *infinite cellular automata* [17, 42] and *interacting particle systems* [104]. For the finite system, a useful way to classify the work is through (i) the update scheme, (ii) the structure of the network, (iii) the vertex state set, and (iv) whether it is deterministic or stochastic. For *parallel update*, one has *generalized cellular automata* as defined in the above-mentioned literature. The asynchronous update schemes includes sequential update [22] and more general block sequential update scheme [5, 118]. A second split goes for the structure of the network. A large portion of

theory is concerned with regular lattice. Most works on finite cellular automata fall into this category. The split of vertex state set is generally Boolean and non-Boolean systems with the Boolean systems being those for which the state set is $\{0, 1\}$. Stochastic systems are those which include one or more stochastic aspect (e.g., functions, network structure, update mechanism). For example, *Probabilistic Boolean networks* are constructed as for Boolean networks, but the vertex functions are chosen in a stochastic manner [147–149].

A concern that is often raised regarding the analysis of finite dynamical system models is that of *computational tractability* [23, 25, 90, 156]. As a simple example, consider the question of determining if a Boolean network has a fixed point. This is equivalent to an SAT-type problem and, in its general form, is computationally hard. There are libraries such as NetworkX [120] and GaLib [59] exposing a collection of graphs and computational algorithms that allow for conducting computational experiments.

As a formal representation of network-based simulation models, validation and stability of graph dynamical system remains a fundamental question and has drawn extensive attentions [9, 17, 58, 108, 148]. In [148], Shmulevich and Kauffman define the notion of *activity* and *sensitivity* with respect to the perturbation of a vertex state, which essentially measures the short-term stability of the system. Other works [98, 100] consider the long-term stability properties such as stability of attractors and stability with respect to the domain of attraction. Generally, these stability studies concentrate on perturbations to all aspects of the system components: network structure vertex state, vertex function, and update scheme. Much of our work will also address the question that how various sources of perturbations impact the system dynamics.

2.3 Related Existing Frameworks

To the best of our knowledge, there exist systems that cover subsets of features of our proposed framework, but they lack the design needed for integrating all parts. Examples of subsets include (i) methods for model validation, uncertainty quantification, sensitivity analysis and experimental design of computer simulation models (ii) standardized simulation initialization and management, and (iii) scientific data registry system. In the following, we review central tools and methods from each of these categories.

DAKOTA (Design Analysis Kit for Optimization and Terascale Application) [1] developed by Sandia National Laboratories, provides comprehensive parameter studies, uncertainty quantification, and optimization algorithms running on supercomputer platforms. PSUADE (Problem Solving Environment for Uncertainty Analysis and Design Exploring) [160] is a C++ based open-source software package. It runs on Linux-like systems

only through command line interfaces. UQ-PyL (Uncertainty Quantification Python Laboratory) [163] is a Python UQ tool-kit with a graphical user interface that allows more friendly user experience. UQLab (Uncertainty Quantification in MATLAB) [111] is an MATLAB-based software for uncertainty quantification. MUQ [127] is a collection of tools for constructing models and a collection of UQ-focused algorithms for working on those models. This toolkit is developed by uncertainty quantification group at MIT. It has multiple UQ algorithms implemented, but has limitations in applications for external simulation models. We believe each individual software or a combination of them can facilitate the model validation life cycle. Unfortunately, most of them lack a standardized model invocation mechanism, a data management scheme for handing large amount datasets in diverse formats, and support for repeatable experiments.

As indicated in [68], scientific data and workflow management is a major challenge in simulation science, especially for those using network-based models with complex structures and vast data throughput. Systems like Kepler [13] provide comprehensive solutions for scientific workflow and data management. Other systems including SAFE [131] and JAMES II [54] focus on simulation experiment automation. However, they are either domain-specific or not related to model validation.

Various related statistical design libraries are also available, many of which are written in Python or R, such as pyDOE [133] and AlgDesign [10]. Other options include commercial software, for example, SPSS and JMP. Frameworks like DAKOTA have internal algorithm library and can be used externally through the interface they provide. Scikit-learn [130] is a comprehensive machine learning package in Python, which incorporates various statistical surrogate model implementations that can be used as models of the model in adaptive sampling and Bayesian calibration problems.

Finally, There are standalone data registry systems [61], but again, they do not directly support UQ/SA/DOE for networked-based simulations.

To the best of our knowledge, many existing tools are aimed at simulation models based on differential equations or some other algebraic forms [1, 160, 163], while network-based simulation models have a fundamental difference in the sense that they always behave like a "black box" such that their underlying dynamics are difficult to describe directly, i.e. the function f in Equation 1.1 does not have an explicit form, thereby the model validation process is not exactly the same with numerical models. In addition, many tools focus on the specific application domain, such as biology [54] and computational fluid dynamics, instead, we do not presume any application area. Finally, few system has a complete mechanism to record the experiment workflow and data flow as we do, as a result, it is hard to archive the experiment results or repeat a completed experiment. We remark

our system is domain-independent and mainly concentrates on network-based models. Moreover, the entire experiment procedures are stored in a registry system such that they can be recovered in an easy way.

Chapter 3

GENEUS: a General Framework for Experimental Design, Uncertainty Quantification, and Sensitivity Analysis

3.1 Introduction

Many existing frameworks for uncertainty quantification (UQ), sensitivity analysis (SA) and design of experiment (DOE) are designed for traditional equation-based simulation models with compact model descriptions, which generally involve less data throughput. On the contrary, the inherent data and computationally intensive nature of network-based computer simulation models gives rise to fundamental challenges when it comes to executing typical experimental designs. In particular, this applies to model validation. Manual management of the complex simulation workflows along with the associated data will often require a broad combination of skills and expertise. Examples of skills include domain expertise, mathematical modeling, programming, high-performance computing, statistical designs, data management as well as the tracking all assets and instances involved. This is a complex and error-prone process for the best of practices, and even small slips may compromise model validation and reduce the human productivity in significant ways.

In this chapter, we present a framework called GENEUS that addresses the challenges of model validation just mentioned. The components of our framework form an ecosystem consisting of (i) model unification through a standardized model configuration format, (ii) simulation data management, (iii) support for experimental designs, and (iv) methods for uncertainty quantification, and sensitivity analysis, all ultimately supporting the process of model validation. It is an extensible design where domain experts from e.g. experimental

design can contribute to the collection of available algorithms and methods. Additionally, our solution directly supports reproducible computational experiments and analysis, which in turn facilitates independent model verification and validation.

In this section, we discuss challenges for designing such as a system, motivations that guide the design, and our major contributions. In Section 3.2, we present the overall system architecture, and the detailed system implementation will be elaborated in the following sections.

3.1.1 Challenges and Motivations

Computational challenges for network-based simulation models

Network-based simulation models (also known as agent-based simulation models) typically represent the interactions of the modeled systems using a graph where nodes denote agents and edges encode actions among them. In this way, they can provide understanding into both local interactive behaviors and global system dynamics. Although they give a more direct representation of modeled systems, network-based simulation models did not become widespread until modern high- performance computing (HPC) techniques emerged because such type of models often involves large data sets and require intensive computational resources. Even with contemporary HPC equipment, experiments with network-based models can cost a lot of time and resources. For instance, an epidemic propagation experiment performed on Chicago synthetic population network, which contains 9.04 million nodes, will produce 954 GB data. The execution of this experiment will take more than 10 hours by using 48 cores of a high-performance computing cluster. The problem scale will continually expand if larger scenarios such as global pandemic simulation are considered. As a matter of fact, how to efficiently manipulate the large data set associated with the network-based model and improve the computational efficiency remains major challenges in this area.

When performing uncertainty quantification and sensitivity analysis, this challenge will become severer for the reason that UQ/SA is usually accompanied with the experimental design which will create multiple simulation runs and replications. Computational demand will accordingly increase with expanding experiment runs and replications, which will give rise to fundamental computational challenges.

Big data challenges for uncertainty quantification.

Quantifying uncertainty requires a large number of model evaluations, except for the tremendous resulted computational overheads mentioned earlier, this will bring great challenges in big data management. Many existing frameworks provide sophisticated solutions and algorithms such as sampling methods, surrogate modeling, adaptive experimental design, inverse model calibration and so on. Examples include DAKOTA, PSUADE, UQ-PyL, etc., see Chapter 2. Unfortunately, many of these existing tools tend to overlook the big-data management challenges, therefore, are not suitable for supporting UQ/SA in network-based simulation models with large data sets.

Motivated by this, we designed our framework such that it can handle UQ/SA problems as well as data management issues. This will alleviate the technical difficulties from the user's side such that they are able to concentrate on the UQ/SA analysis.

Configuration standardization challenges for simulation models.

A simulation workflow is a collection of methods (programs, scripts) with matching invocation configurations applied to existing data sets (input). It will, in turn, gives rise to new data sets which are the simulation outcomes. A major obstacle one comes across is dealing with complex model configurations. A simulation model may have one or more configuration files in heterogeneous formats. For example, EpiFast [31] has one main configuration file and several sub-configuration files in different formats and structures. It is generally hard to manipulate the unstructured model configurations without a standard. Operations such as locating the statistical factors, varying a group of parameters, maintain and reuse existing configuration elements and so on are done manually are easy to result in errors. In addition, to enable general experimental design for a diversity of simulation models, it is essential to design a standardized configuration format for uniform model invocation.

Moreover, in simulation science, it may be easy to write a model configuration for a particular simulation model. However, repeated model configuring is tedious, easily error-prone and likely to have limited choices in experimental design methods. A standardized configuration format not only prevents such human-initiated errors but also provides a possibility for allowing general experimental design method.

Compatibility challenges for general UQ/SA environments

A general environment is supposed to be compatible, extensible and easy to integrate. The standardized model configuration format allows a uniform model invocation, however, the actual execution of the simulation model requires a translation from the standardized configuration format into the model-native format. In addition, UQ/SA problems are often very domain/application-specific. How to handle the diverse model and analytical requirements is still challenging.

3.1.2 Contribution

Motivated by above mentions points, we have developed GENEUS - a general computational environment for supporting uncertainty quantification, sensitivity analysis, and experimental design of network-based simulation models. We summarize the key features of GENEUS as follows:

GENEUS introduces a standardized model configuration format

We have designed a standardized XML-based configuration specification that fuses heterogeneous configurations into an integral hierarchical-structured format. It incorporates a tree architecture, where a leaf node denotes an atomic configuration parameter specified by a key-value pair and an inner node represents a group of parameters with all its descendants belongs to the same group. The value of an inner node will be a registry ID that refers to a data object. With this hierarchical design, it is possible to vary both a single configuration parameter and a group of parameters at the same time. This design enables assembling arbitrary parameters into a complex parameter which will be recorded in a central registry system. Any individual parameter or group of parameters can be easily accessed and reused if future experiment requires. We also provide APIs for parameter navigation within the standardized configuration file such that one can get access to any parameter regardless of its actual location. There will be a *model wrapper* code that converts between the standardized format and the model-native format, all in a transparent and automatic manner.

GENEUS addresses data management challenges for UQ/SA

Conducting UQ/SA study often needs to iterate through the experiment cells and perform analysis over the map from the model input to the simulation outcome. For network-

based simulation models, the input/output can be simple scalar values or arbitrary complex objects. To manage the experiment cells and their associated data sets, GENEUS incorporates an internal registry system to record the simulation workflow and the data dependencies. This central registry system is built upon a relational database which allows queries of experimental data set such that fast analysis is accelerated for UQ/SA studies.

Another important purpose of such central registry system is for supporting repeatable experimental design. Since detailed simulation procedures are recorded in a database, one can easily extract the required information for repeating the experiments without having to reprogram or reconfigure the model. This feature also allows for independent model verification and validation that is usually for satisfying the third-party's requirement.

GENEUS streamlines uncertainty quantification workflow.

Many existing UQ/SA frameworks put emphasis on uncertainty quantification algorithm implementation and tend to overlook the model execution and data management issues. This is resulted from the fact that many equation-based simulation models are easy to configure and of small scale in terms of data volume. They often separate the analysis and the model execution and focus on the former. However, in our opinion, analysis and model execution are inseparable especially for large-scale network-based simulation models because in many cases an analytic strategy is restricted by the limited computational resources and thus cannot be performed without a sophisticated execution budget plan.

GENEUS is a framework that takes into account the model execution and data management challenges. It exposes possible UQ/SA solutions to domain experts and let them focus on the analysis and experimental design problems instead of the intimidating big-data issues.

GENEUS is a general and extensible framework.

GENEUS incorporates a *model wrapper*, which is a program dealing with model level requirements, such as configuration file translation, execution environment setup, data preprocessing and so on. The model wrapper serves as an adapter between GENEUS and simulation models. Construction of a model wrapper needs programming effort from the user's side, however, this is a one-time effort. We provide a developer's manual that can help the users create customized model wrapper code.

GENEUS has a UQ/SA/DOE algorithmic library which contains a list of common analytic methods. However, a general framework is supposed to be extensible and compatible with customized methods. Considering this, we implemented these methods as plug-ins and

GENEUS supports the fast addition of new algorithms.

Last but not least, GENEUS is a general environment in the sense that it can be easily integrated with other systems including simulation models, similar UQ/SA frameworks, or other tools that only needs partial functions of GENEUS. GENEUS has a loosely-coupled design: each component is unaware of other units and can function independently. This design allows GENEUS to work as a driving engine for streamlining UQ/SA/ED workflow, also, to work as an embedded tool for external systems.

3.2 System Overview

Figure 3.1 illustrates the overall architecture design of GENEUS system. It has a loosely coupled design with several major components. First, a *model configuration module* generalizes configuration format for a variety of simulation models. It allows uniform model invocation and supports for general experimental design. This module consists of a suite of XML/XSD grammar with associated supporting code. With the standardized simulation model configuration format, it is possible to conduct experimental design in a uniform manner because model-specific constraints are handled transparently by a *model wrapper* code, which is the second important component. Model wrapper processes model-native requirements, in particular, it translates the standardized configuration file into the model-compatible format and handles model execution requests. Third, GENEUS has an internal experiment management system. The key of the experiment management system is a central registry built upon a relational database, in which study and experiment instances, associated configurations and data sets are indexed. Operations related to simulation data management are done through a collection of registry APIs. Finally, GENEUS has an algorithm library that contains many uncertainty quantification, sensitivity analysis, and experimental design methods. Each method is implemented as a plug-in such that adding and updating method can be easily managed. It is worth noting that all of the core components are standalone, and are not aware of other components. In this way, each single component can provide services for external applications, GENEUS can also function as a self-contained software package for any individual researcher or research group conducting UQ/SA/DOE studies.

In addition to the core components, we assume several supporting infrastructures. First, a physical data storage portion will handle the actual data. It will deal with arbitrary combinations of the hardware and the storage methods (flat files, DBMS, RDF, etc.) and will interact with the resource management module via generic APIs. Second, platforms such as Simfrastructure [33] support a general environment for high-performance computing

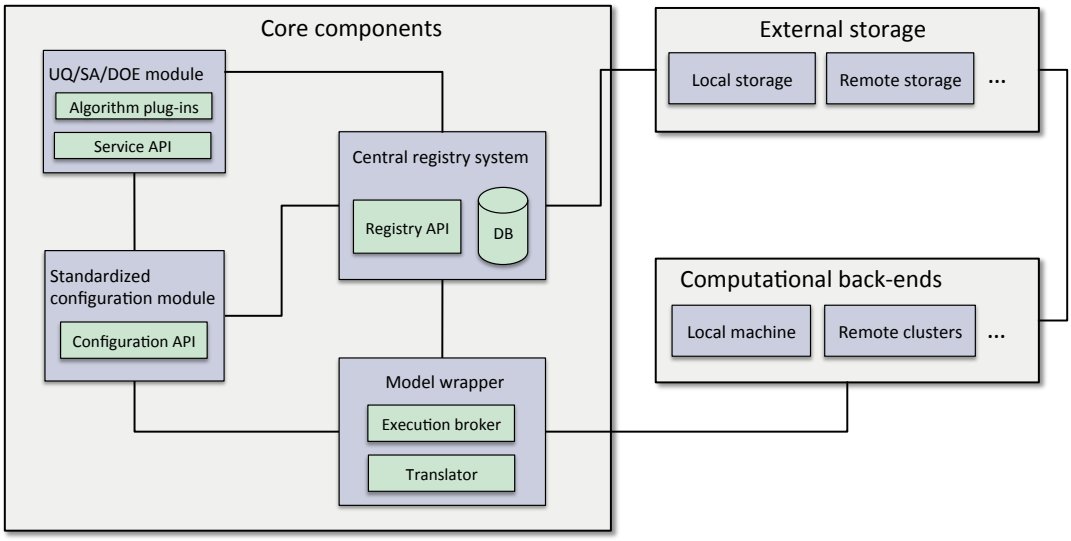


Figure 3.1: *An overall system architecture.*

(HPC). It is worth noting that although the supporting infrastructures are necessary, we do not assume any particular form of them such that this framework remains independent from the platform.

The design concepts and implementation details will be elaborated in the following sections.

3.3 Standardized Model Configuration

3.3.1 Background and Motivation.

Designing a standardized configuration specification is challenging.

Felfernig states in [56]: "Informally, configuration can be seen as a special case of design activity, where the artifact being configured is assembled from instances of a fixed set of well-defined component types which can be composed conforming to a set of constraints." This definition is informal but clearly points out that software configuration consists of two major elements: (composed) configuration components and a set of constraints. For functionality, the configuration provides approaches to initialize the program, to control of the program behavior and to expose the program execution details. In the context of computer simulation, configurations can be simulation model parameter settings or

program runtime configurations. In this dissertation, configuration refers to the former if not explicitly indicated and the program/software usually refers to the simulation model.

Nevertheless, designing model configurations is a non-trivial task. First of all, Model configurations can be as simple as a single command line argument, they can also have very complex structures with intricate dependencies among the configuration entries. A simple format with key-value pairs as configuration entries is easy to develop, however, many requirements such as varying a group of parameters simultaneously is difficult to fulfill because of the inherent design limitation. Configurations with complex structures, on the other hand, are challenging for the domain expert to properly work with, therefore make the model hard to behave in a preconceived manner. Secondly, a standardized model configuration is not only essential for uniform model invocation mechanism, but also a requirement of the general experimental design. Experimental design involves frequently inspecting model parameters and setting up their values. Navigation through an unstructured file(s) is nettlesome and error-prone, thereby hamper appropriate experimental design practices. As a result, standardization of model configuration is an essence for a general environment like GENEUS. However, different simulation models have tremendous diversity in configuration format, it is non-trivial to design a standardized, customizable, experimental design-oriented configuration format.

Features that are essential to a comprehensive standardized model configuration

A standardized model configuration is supposed to support highly variant models and allows for the design of customized functions, but still, has a uniform structure from the view of high-level callers or users. It would benefit from having following features:

(i) *Standardized structure representation:* a standardized configuration structure is not only easy to maintain from the system administrator's side but also easy to learn and work with for users. There are many possible choices of configuration architectures such as main/sub-configuration structure, function-oriented configuration, hierarchical configuration format and so on. Regardless of the design pattern, a standardized configuration format is supposed to capture both simple and complex configuration instances. In a nutshell, The standardized structure represents the common knowledge base for various simulation models.

(ii) *Customizable components:* to incorporate with different simulation models, the standardized configuration format is supposed to capture model-specific requirement such as particular constraints on configuration entries. This represents the domain-specific knowledge of simulation models.

(*iii*) *Standardized interfaces:* each model-specific configuration format has its own proprietary characters. This makes the integration of such configuration formats tricky for higher-level applications such as experimental design module. Standardized configuration interfaces provide services including creating the configuration, adding/deleting configuration entries, locating experiment parameter, setting parameter value, grouping entries, registering configuration entries and so on. The standardized interfaces will encapsulate the low-level operations as services such that the efforts of development and maintenance are reduced. In addition, they also provide a generic approach for developing a user's interface.

(*iv*) *Reduced development and maintenance efforts:* the development of configuration specification is conducted in cooperation between technical experts and domain experts. The development procedure can be very challenging and expensive, as a result, the effort for constructing/modifying such standardized configuration specification is an important evaluation metric of system usability.

(*v*) *Reusable configuration entries:* As stated in (*iv*), resources required to develop and maintain configuration specifications are substantial. In many cases, an experiment instance changes a small part of the model configuration and most remaining configuration components are unchanged. Modern model development and experiment setting are quite modular, so as configuration specifications should be. This gives rise to the reusability consideration of model configurations. A reusable configuration format is supposed to have a fairly modular design and support for convenient configuration component registration and retrieval.

3.3.2 Design Concept

Model-driven architecture (MDA) is a software design pattern launched by the Object Management Group (OMG) in 2001 [154]. One of the main aims of MDA is to separate the platform-independent model (PIM) and the platform-specific model (PSM). In the context of simulation model configuration, this essentially means a separation of common configuration knowledge and model/domain specific knowledge. We adopt the MDA design philosophy: GENEUS incorporates a formal simulation model configuration grammar defining the common configuration architecture. Each simulation model has its own configuration specification complying to the grammar that represents the model-native constraints.

In our design, we generalize the structure of configuration specification as a tree architecture. Basically, there are two types of configuration elements: simply elements and complex elements. A simple element is an atomic configuration entry represented by a leaf node,

and a complex element captured by a tree branch is composed of a collection of elements (simple/complex) underneath the branch. This is a fairly modular design such that one can assembly arbitrary complex configuration elements as needed. Each complex configuration element is registered in a central registry system (see Section 3.4.2), in this way, it can be easily accessed and reused by other applications. For each simulation model, there will be a base configuration file complying to this standardized structure and account for model-specific requirements or constraints. This base configuration file serves as a template representing the model/domain-specific knowledge.

In addition, the design of composed configuration elements allows for varying a group of parameters simultaneously. This has a lot of practical applications. For example, in [159], the researchers seek to determine if the same influenza vaccination strategies would have the same level of effectiveness when applied to two different US metropolitan areas, Miami and Seattle, which have very distinct demographic properties. In this experiment, vaccination intervention is applied to the population with particular demographic properties such as age-group, occupation etc. This involves grouping demographic features and applying different intervention strategies to different demographic groups. By using our standardized configuration format, each demographic group can form a complex configuration object which is registered with a unique ID. As a result, the creation of a demographic group is a one-time effort. For studies like [159], human productivity can be significantly improved because the model configuration is managed in a reusable manner and the chance of errors is reduced.

The standardized configuration format is implemented in extensible markable language (XML), this will be elaborated in the next section. The model-specific constraints are captured by XML Schema Definition (XSD) and Schematron which is a rule-based validation language for making assertions about the presence or absence of patterns in XML. We also provide APIs for many operations such as creating configuration specification, edit configuration file, parameter navigation, parameter setup, export individual simple/composed parameter object, import parameter object from existing file, etc. In this way, users will not see the XML files directly, but use the interfaces we provide.

Finally, this architecture is able to capture many complex configuration structures. As mentioned earlier, the model configuration can be significantly diversified, both in format and in structure. Common configuration format/language includes XML, JSON, INI, YAML, plain text and so on. Translation of GENEUS standardized configuration format is model-specific and will be handled by the model wrapper. Our design is aimed to capture many diverse configuration structures. The simple element in the standardized configuration specification captures key-value pairs in a straightforward way. For many other complex types such as section element in JSON and INI, list elements in YMAL are

treated as complex elements and represented by tree branches. As a concrete example, Figure 3.2 gives a typical EpiFast configuration file. EpiFast configuration consists of one main configuration file and multiple sub-configuration files which are referenced in the main configuration file, see entries marked by the red boxes in Figure 3.2. Typically, experiment parameters can lie in any configuration file and it is difficult to locate them for experimental design or setting up parameter values. In the standardized configuration format, these sub-configuration files are treated as tree branches. With the parameter navigation API we provide, fast parameter locating is possible. As a matter of fact, this tree structure and simple/composed elements provide an approach to constructing many complex configuration specifications.

```
ConfigVersion = 2009
ContactGraphFileFormat = EFIG6Bb
ContactGraphFile = LBR_430-socnet.efi
Transmissibility = 3.07537688442e-05
InfectiousPeriodFormat = DISTRIBUTION
InfectiousPeriodFile = Infectious.Period.Distribution
IncubationPeriodFormat = DISTRIBUTION
IncubationPeriodFile = Incubation.Period.Distribution
EpidemicSeedType = RANDOM_SEEDS_FIRST_DAY
EpidemicSeedNumber = 5
EpidemicSeedFile = subpop_all.txt
SimulationRandomSeed = 7654321
InterventionFile = Intervention
IterationNumber = 10
LogFile = log/EFL
OutputFile = out/EFO
OutputLevel = 2010
SimulationDuration = 200
```

Figure 3.2: *An example EpiFast main configuration file. Entries marked by red boxes denote references to sub-configuration files.*

3.3.3 XML/XSD-based Configuration Specification Grammar

The standardized configuration format is implemented in extensible markable language (XML). XML has many known advantage as a configuration language such as its good

cross-platform capability, human- and machine-readable features. However, comparison of configuration languages and other domain-specific languages is out of the scope of this thesis. The basic unit of the XML configuration specification is the parameter element, which is defined by a complex XSD element type shown in Figure 3.3. One can observe that the parameter element has a recursive definition, which means one parameter element can have one or multiple other parameter elements as its sub-elements. This essentially defines the composed/complex configuration element. We assign several attributes to the parameter element including parameter key, data type, exposed flag for marking experiment factor, optional flag, and default value. The complete XSD grammar is given in Appendix A.

```
<xsd:complexType name="parameterType">
 <xsd:sequence>
  <xsd:element name="value" type="xsd:string"/>
  <xsd:element name="parameter" type="parameterType"
    minOccurs="0" maxOccurs="unbounded"/>
 </xsd:sequence>
 <xsd:attribute name="key" type="xsd:string" use="required"/>
 <xsd:attribute name="type" type="xsd:string" use="required"/>
 <xsd:attribute name="exposed" type="xsd:boolean" use="required"/>
 <xsd:attribute name="optional" type="xsd:boolean" use="required"/>
 <xsd:attribute name="default" type="xsd:string" use="optional"/>
</xsd:complexType>
```

Figure 3.3: *Formal XSD definition of parameter type.*

Figure 3.4 gives an exemplary XML configuration file that corresponds to the EpiFast configuration shown in Figure 3.2. Here "InterventionFile" is a sub-configuration file that is represented by a tree branch of the main configuration root, and "school_closure_compliance" is an exposed statistical factor located in the intervention file; it is a leaf node in the standardized format.

Manipulation of such standardized XML configuration is handled through XPath operations wrapped by a Python code.

It is worth noting that this standardized configuration format is designed for a variety of simulation models instead of any particular one. There will be a model wrapper code that converts to and from the model-native format, all in a transparent and automatic manner. Figure 3.5 shows how to use the standardized configuration format to initialize

```
<parameters>
 <parameter key='ConfigureVersion' type='int'
  exposed='false' optional='false'>
   <value>2009</value>
 </parameter>
 <parameter key='ContactGraphFormat' type='string'
  exposed='false' optional='false'>
  <value>Infectious.Period.Distribution</value>
 </parameter>
 ...
 <parameter key='InfectiousPeriodFile' type='string'
  exposed='false' optional='false'>
  <value>EFIG6Bb</value>
  <ID>dataset_112</ID>
 </parameter>
 ...
 <parameter key='InterventionFile' type='string'
  exposed='false' optional='false'>
  <value>Intervention</value>
  <ID>dataset_114</ID>
  <parameter key='InterventionFile/school_closure_compliance'
   type='float' exposed='true' optional='false'>
    <value>0.5</value>
  </parameter>
 </parameter>
 ...
</parameters>
```

Figure 3.4: *Corresponding XML standardized configuration file for EpiFast shown in Figure 3.2. Some configuration entries are omitted.*

experiments. The user will select factors, levels, experimental design type though the user interfaces at a standardized level.

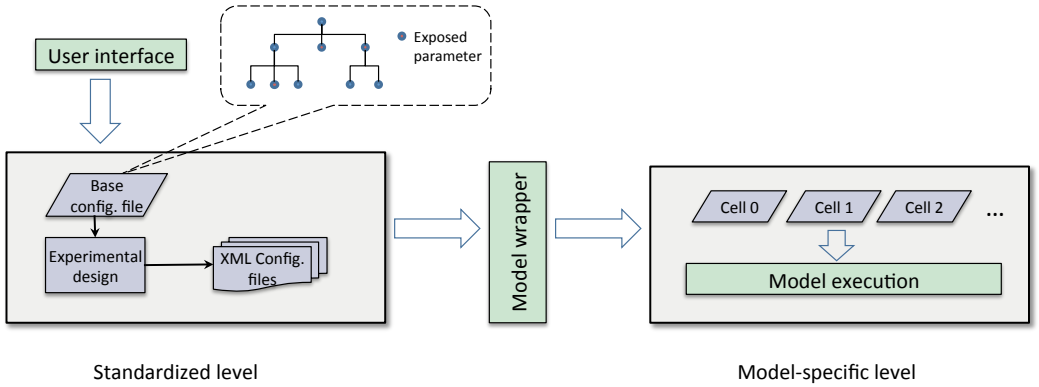


Figure 3.5: *Using standardized configuration format to initialize experiments.*

3.4 Central Registry System

3.4.1 Motivation

The lack of experiment management scheme reduces human productivity.

A computer simulation experiment is a workflow consists of following steps: (i) select experiment parameters and simulation responses; (ii) design experiment; (iii) execute simulation model with specific parameter configuration; (iv) collect experiment results and perform the analysis.

In many situations, this pipeline is handled independently by different people and on different platforms. For example, selection of model parameter and responses needs inputs from domain experts; model configuration and execution are usually done by computer scientists; and data analysis is accomplished by data scientists and statisticians. On the other hand, model configuration, model execution, and data analysis do not necessarily happen on the same machine. As a result, computer simulation experiments often involves collaboration among people with various background and distributed computing resources. If big data sets are involved, data manipulation including data preprocessing, data transportation, data access, etc. will intensify the challenge.

In [28], Bengtsson et al state, “Our industrial partner often experiences disturbances in

production and material handling processes where parts are missing or lost. This problem causes delays in the manufacturing processes and gets even more complicated by the global supply chain. To minimize the disturbances, the industrial partner is investigating potential investments in new tracking technology." Indeed, in practical simulation engineering, many efforts are spent on preparing input data from different sources, configuring the simulation model, setting up the experiment, collecting and processing simulation outcomes and tracking the simulation workflow. Usually, they are done manually, errors are easy to happen and the lack of efficient management of computer simulation experiments becomes a major reason for low human productivity.

Systematical experiment management supports for repeatable simulation experiments.

Sargent writes [142]"Independent verification and validation (IV&V), uses a third (independent) party to decide whether the simulation model is valid. The third party is independent of both the simulation development team(s) and the model sponsor/user(s). The IV&V approach should be used when developing large-scale simulation models, whose developments usually involve several teams. This approach is also used to help in model credibility, especially when the problem the simulation model is associated with has a high cost. " As a result, a credible simulation experiment is expected to be repeatable. Due to the complexity of large-scale simulation models and third party personals' limited knowledge about the model, the cost of model validation is usually quite significant, especially when extremely high model confidence is required.

As a matter of fact, repeatability becomes a basic requirement in experimental science [14]. IV&V researchers need documentation to understand experimental results. To support easily repeatable experiments, a management system is essential to index and record distributed data sources, model configurations, simulation outcomes and so on.

3.4.2 Registry Design

Our solution for experiment management is to integrate a central registry system in GENEUS. The registry is a central place to record any digital objects including data classes, datasets, configuration files, simulation model specification, etc. Data type, data class, and data set are fundamental concepts for data management. On a specification level, using the notion of data types, one can specify templates or definition for data sets. These definitions are called data classes. Generally, with this data architecture, arbitrary complex data object such as synthetic contact networks, input/output data sets, model configuration files, simulation workflow specifications, experiment instance can be regarded as data sets with

respect to specific data classes definition. They will be managed systematically through the registry system.

The actual registry system is built upon a relational database management system. It records simulation model, individual UQ/SA/DOE study instance, each experiment run and its associated configuration file, input data sets, model parameter values, and simulation outcomes. Note that study instances are the basic units in GENEUS, and they are very problem-specific. For example, an Ebola disease parameter sensitivity analysis study uses EpiFast as the underlying simulation model; it has a collection of experiment runs that are resulted from particular experimental design method; each experiment run has multiple replications. The registry system records all the intricate relations among studies, experiment runs, replications, along with their associated configuration specifications, dependent data sets, execution environment and so on. This will allow systematical book-keeping and easy recovery of any completed experiments. Table 3.1 gives detailed database schema design.

On top of the relational database, we provide high-level services such as register, delete, update, retrieval of any recorded digital object. The benefits of this design have two folds: first, for those who need to repeat a completed study, she/he can obtain all necessary information such as simulation model specification, study location and working directory, experimental design information, input/output data sets, and a map from model parameter values to response values. This information can help the researcher to easily repeat the experiment without many efforts. Second, the high-level services allow fast data access which is very useful for particular analyses like the adaptive experimental design (or adaptive design for short). Adaptive design often involves frequent data inspection of the previous experiment runs, with registry services like retrieval and summarize, this adaptive procedure can be automated.

3.5 UQ/SA/DOE Capabilities

Quantifying uncertainties is problem-specific: it can be as simple as drawing scatter plots that display response values against input parameters; it can also mean in-depth analysis with intensive function evaluations. In GENEUS, we provide a multitude of methods for uncertainty quantification, sensitivity analysis as well as the design of experiment. They are summarized in the following sections.

Table name	Description	Data	
		Role	Name
model	Simulation model information.	Primary key Attribute	model_id model_name
study	Study instances. A study instance consists of a collection of experiment instances.	Primary key Foreign key Attribute	study_id model_id study_name, time, system, working_directory, description
experiment	Experiment instances.	Primary key Foreign key Attribute	experiment_id study_id experiment_name, time, system, cell_directory, description
base_configuration	Standardized configuration template. Each study has one base configuration file.	Primary key Foreign key Attribute	base_config_id study_id, model_id base_config_name, system, location, description
configuration	Configuration file for each experiment instance.	Primary key Foreign key Attribute	config_id experiment_id config_name, system, location, description
data_class	Data class definition.	Primary key Attribute	data_class_id data_class_name, format, description
data_set	Physical datasets. Each dataset belongs to a specific data class.	Primary key Foreign key Attribute	data_set_id data_class_id data_set_name, format, system, location, description

Table 3.1: *Detailed database schema (core tables) for central registry system.*

Design type	Design Method
Traditional designs	Full-factorial design [115] Fractional-factorial design [115] Plackett-Bruman design [132] Box-Behnken design [36] Central-Composite design [115]
Space-filling designs	Monte-Carlo design [141] Latin Hypercube sampling (LHS) [112] Orthogonal array design (OA) [73] OA-based Latin hypercube design [158]

Table 3.2: *Integrated experimental design methods.*

3.5.1 Design of Experiment Methods

As mentioned earlier, computationally expensive computer simulations usually prevent exploring the parameter space exhaustively. The efficient design of experiment is essential for performing UQ/SA studies. Traditional DOE methods such as factorial design consider a small number of factors and levels, which is not applicable for modern computer experiments. Space-filling design techniques such as Latin hypercube sampling, and orthogonal array etc. are developed and become more and more popular in the past decades because one of their important features is that they are able to cover as much as the experimental space with a small number of sample points. They can thereby save considerable computational time and resource comparing with random Monte Carlo method and traditional experimental designs.

In GENEUS, we integrate both classical and space-filling design methods such that one can choose the design type flexibly according to the actual study, see Table 3.2 for a list of implemented DOE methods. Many design algorithms are derived from pyDOE [133], which is an open source python DOE package.

3.5.2 Parameter Study and Sensitivity Analysis

Researchers often want to test individual parameter effect as well as joint effects of two or more parameters (interactions) with or without uncertainties' present. The uncertainty of the output can be characterized by using a few basic statistical quantities including

moments and confidence levels, which uniquely determine the probability distribution. In GENEUS, these statistical quantities can be automatically calculated and they can be delivered as a collection of diagrams.

We will integrate both local and global sensitivity analysis methods, specifically, the Morris One-at-A-Time (MOAT) for studying individual parameter and variance-based methods including ANOVA, Sobol sensitivity and Fourier Amplitude Sensitivity Test (FAST). Many sensitivity analysis methods require tens of thousands of model runs to achieve an accurate approximation of the variance, which is very difficult when considering the overhead. Therefore, we developed metamodels as surrogates of the actual simulation models for sensitivity study, this will be elaborated in the next section.

3.5.3 Surrogate Models and Adaptive Experimental Design

Surrogate models are regarded as "models of models". They are widely used in regression, machine learning to interpret existing data and identify the underlying functional relations. In our context, they can be used as an alternative to the real simulation models. For example, a UQ study may require 1000 model runs to obtain the desired accuracy, however, the computational budget can only afford 500 runs. Then, one can consider to fitting a surrogate model by using the data from the initial 500 runs and evaluate the surrogate model instead. Another important use of the surrogate model is to guide the experimental design, specifically, help to predict next potential sample point to experiment. This means selecting samples adaptively based on information from previous design in a sequential manner. The example purposes of adaptive design include locating the most uncertain area of the parameter space or achieve a particular optimization goal such as maximizing the minimal distance between the sample points.

The simplest surrogate model is the linear model which assumes a linear relationship between the input and output. If the data present an observable nonlinear pattern, one can choose higher order polynomials as underlying model, however, there is a risk of over-fitting. As a matter of fact, there are a lot of studies on model selection and finding a trade-off between the interpretation power and the over-fitting problem(Occam's razor principle) [57]. It is worth noting that among the many choices of surrogate models, Gaussian process (GP) [134] is very popular within the UQ society. GP assumes every finite collection of random variables follow a multivariate normal distribution. It can be uniquely determined by a mean function and a variance function. In the Gaussian process model, the correlation of data points is stronger if they are closer to each other, which makes sense in many application scenarios. Other surrogate models include support vector machine(SVM), regression trees, nearest neighbors and so on.

Category	Method
Sensitivity analysis	Fourier Amplitude Sensitivity Test (FAST) [139] Morris one at a time(MOAT) [116] Sobol Sensitivity Analysis [152] Surrogate model-based sensitivity analysis [123] Analysis of variance (ANOVA) [115]
Surrogate modeling	Gaussian process [134] Generalized linear model [141] Random forest [37] Support vector machine (SVM) [151]
Calibration & parameter optimization	Bayesian method [77] Genetic algorithm [113] Simulated annealing [82] Surrogate model-based optimization [77]

Table 3.3: *UQ/SA-related algorithms.*

In GENEUS, we incorporate a collection of surrogate models, see Table 3.3. They can be used for different purposes such as supporting model calibration and adaptive experimental design.

3.5.4 Model Calibration

As stated earlier, the inverse problem is an important category of UQ studies. Given physical experiment data and computer simulation results, this type of UQ method assesses the discrepancy between reality and simulation models (bias correction) and estimates the values of unknown parameters (parameter calibration).

In our work, we adopt the Bayes' view, i.e. regard probability as the confidence level of an event. The model calibration process then relies on the Bayesian framework. We assign initial probability distributions to unknown parameters based on the estimation of the experts or pioneer studies and then update the parameters against available data, after each iteration, we obtain a posterior of the model parameters. Usually, this process also

employs a surrogate model for the iterative function evaluations. In GENEUS we provide a pipeline for supporting a broad range of model calibration workflows.

3.6 GENEUS Workflow

Previous Sections (3.3, 3.4 and 3.5) present the detailed design and implementation of major components. In this section, we will introduce the typical GENEUS workflows as an illustration of usability of the software package, see Figure 3.7. It is worth mentioning that GENEUS documentation and complete API description is distributed with the software.

Experimentation workflow

Experimentation workflow is the major process running in GENEUS whenever a simulation model execution is required. As is shown the pipeline marked by blue arrows in Figure 3.7, one can first define the model parameters through the parameter specification section in GENEUS configuration file, which is also in XML format, see Figure 3.6 as an example. Generally, there are two types of parameters: a continuous parameter is given by its range (upper and lower bound), number of levels and the probability distribution it follows, the model configuration module will discretize the parameter into levels according to the distribution, whereas a discrete parameter is directly given by a list of values. Experimental design is specified by giving necessary information such as the design type, the number of sample points and replications and so on via the experimental design API. Then, the experimental design module will crank out a collection of corresponding experiment cells containing standardized configuration files which will be translated to executables by the model wrapper for model execution. The input/output data sets and model configuration files will be recorded by the central registry such that analysis module could fetch the necessary data only through the registry API.

Model configuration workflow

The model configuration workflow is the associated operations for the standardized model configuration format. It involves creating a new configuration template for a simulation model; editing existing configuration file by adding, deleting and changing any configuration elements; save a configuration element (either an atomic element or a tree branch representing a group of elements) to a file or register it to the central registry; load a

```
<param_spec>
 <continuous_parameters>
  <parameter name="temperature_threshold">
   <data_type>float</data_type>
   <lower_bound>15</lower_bound>
   <upper_bound>25</upper_bound>
   <distribution>UNIFORM</distribution>
   <interval>5</interval>
  </parameter>
  ...
 </continuous_parameters>
 <discrete_parameters>
  <parameter name="kappa">
   <data_type>int</data_type>
   <values>
   <value>100</value>
   <value>500</value>
   <value>1000</value>
   </values>
  </parameter>
  ...
 </discrete_parameters>
</param_spec>
```

Figure 3.6: *A snapshot of parameter specification in case study 5.3*

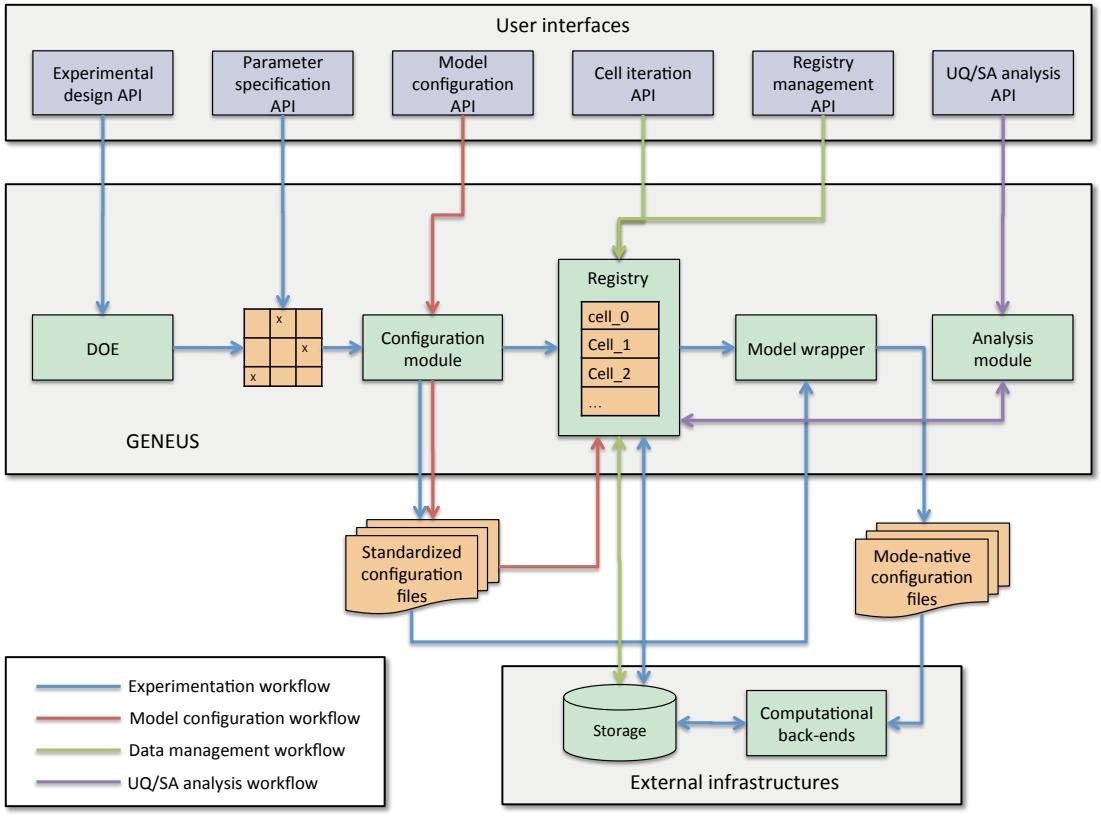


Figure 3.7: *Typical GENEUS workflows.*

configuration element into a existing configuration file. These operations are done through the model configuration API and can assemble arbitrary configuration specification.

Data management workflow

The data management workflow handles all the data requests such as finding the location of the input/output data sets for a simulation experiment. In addition, it realizes a powerful function: iterating through the experiment cells. For example, by using the cell iteration API, one can extract the outcome or other information of a simulation experiment corresponding to an arbitrary combination of parameter settings. This will facilitate easy post-simulation analysis such that the analysis workflow is independent of the model execution procedure. The registry management API handles normal database operations such as inserting/deleting data set and so on.

UQ/SA analysis workflow

With the above-mentioned design, UQ/SA analysis can be independent with the model execution if only it has been finished and the results have been registered. For different UQ/SA problems, it is possible to specify the customized analysis via the UQ/SA analysis API, for example, choosing the sensitivity analysis type and metrics.

In Figure 3.7, the box on the top gives common APIs available in GENEUS, this can also illustrate the capability and usability of GENEUS system.

3.7 Reduction in Human Effort

Figure 3.8 provides turn-around time for the UQ studies of several simulation models: graph dynamical system calculation tool, pest diffusion model, and EpiFast. These times were estimated by using a constructive cost model (COCOMO). COCOMO was developed by Barry Boehm to predict the work effort and development time required for a software development project [34]. It estimates the human effort based on the size of the project represented by the lines of source code. The model parameters are derived from fitting a regression formula using data from 163 historical projects. It is important to note that COCOMO is size oriented, therefore, this method may not cover the effort required for system design. Many researches have reservations with respect to the use of this model. However, here we present a starting point for additional and more in-depth studies regarding human productivity.

We split the UQ/SA study into six phases: model configuration, data preparing, experimental design, model execution, postprocessing, and analysis. Figure 3.8 shows that GENEUS is able to reduce human effort significantly in model configuration, data preparing, experimental design, postprocessing, and analysis. However, the model execution time will depend largely on the complexity of the simulation model itself, thus, the turn-around time cannot be reduced in this phase. It is worth mentioning that for each simulation model, development of the model wrapper will require some time, however, this is an one-time effort that is only required when new simulation model is integrated. Even if considering the model wrapper development time, the total turn-around time can still be reduced by using GENEUS system.

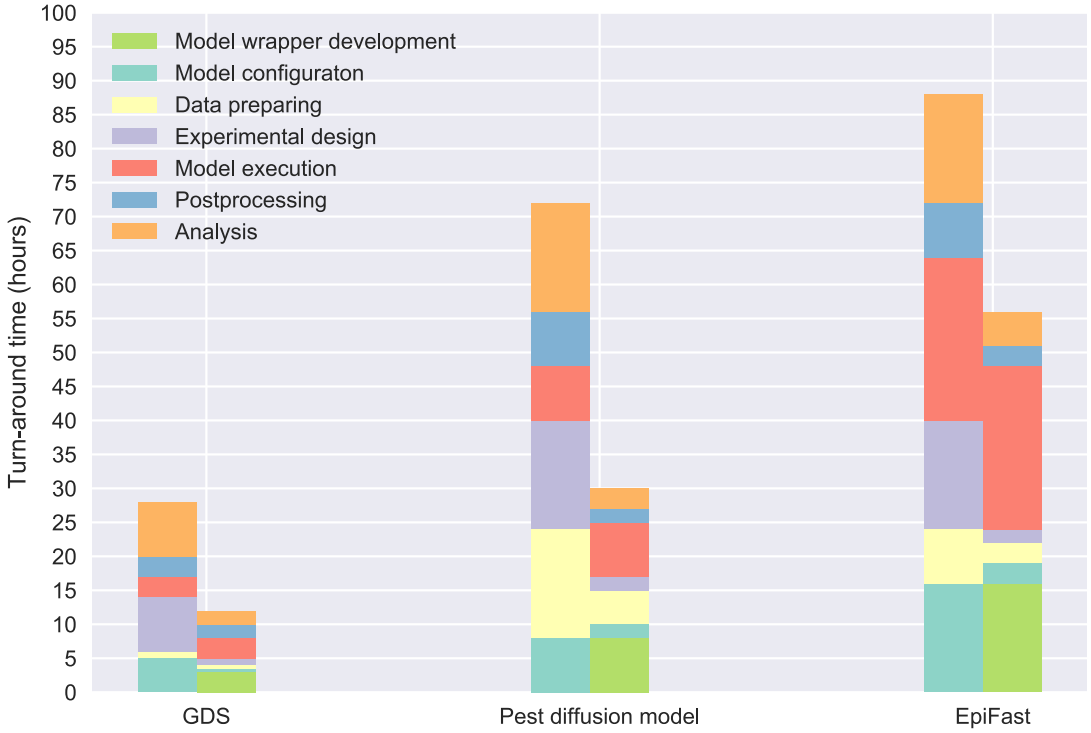


Figure 3.8: *Human effort reduction by using GENEUS for GDS calculation tool, pest diffusion model, and EpiFast.*

3.8 Summary

In this chapter, we present the design philosophy and technical details behind GENEUS. As a summary, GENEUS is a computational framework for supporting uncertainty quantification, sensitivity analysis and experimental design. Apart from the core UQ/SA/DOE capabilities, it has several distinguishing features: (i) By design, GENEUS supports a large number of simulation models through the introduction of a standardized configuration format and streamlines the model execution workflow though its model wrapper component. (ii) It integrates a large number of experimental designs, including adaptive designs, as well as many UQ/SA methods though its plug-in design. (iii) GENEUS incorporates a standardized specification of DOE/SA/UQ thus it supports analysis in a model-independent manner. (iv) The GENEUS registry records all simulation data dependencies and supports scien- tifically reproducible experiments. (v) The registry has a built-in management system with digital library functions for systematically exploring simulation outcomes and cells of experimental designs. (vi) GENEUS is a flexible system that can function in a standalone manner through its running server and can also be embedded within

other applications. (*vii*) GENEUS is extensible and allows developers/domain experts to contribute to its algorithm library. (*viii*) GENEUS provides domain experts and other researchers lacking the full set of skills (e.g., modeling, computing, data management, and statistical analysis) robust access to its UQ/SA/DOE methods in support of for example model validation.

However, there are currently some limitations of GENEUS: integration of new simulation models still requires extra efforts. Specifically, the model wrapper module needs to be provided by the user. Moreover, the execution and computational efficiency of this framework depends largely on the simulation model itself and in most cases, overshadows the overall running time. However, the increase in human productivity and reduction in inadvertent errors is expected to be significant. We have not yet had the time to conduct a formal study or survey to quantify this, but from our own hands-on experience with this class of simulation models, this saving is tremendous. Moreover, the standardized format has several additional advantages waiting to be addressed. For example, it supports automated and instant GUI construction (through the standardized configuration file) which in turn provides domain experts immediate ways to specify model configurations and conduct studies.

To summarize, GENEUS provides a novel solution for supporting model validation of large-scale network-based simulations. Any individual modeler, research group, or third party can benefit from using this system to conduct the model validation studies as well as other analyses.

Chapter 4

Stability of Graph Dynamical Systems

4.1 Introduction

4.1.1 Background

A large range of complex phenomena such as biological social and technical systems can be modeled as discrete dynamical processes taking place over networks [53, 92, 93]. Many dynamical system models have been proposed to capture such systems. Examples include cellular automata (CA) which were originally introduced by von Neumann [162], Boolean networks (BN) proposed by Kauffman in [92, 93], Random Boolean Networks (RBN), see [144, 146], automata networks [63], and polynomial dynamical systems (PDS), see [85, 86]. Graph dynamical systems (GDSs) generalizes the concepts of cellular automata and Boolean networks and focuses on discrete dynamics evolving as

$$x(t+1) = F(x(t)), \qquad \text{where} \qquad F\colon K^n \longrightarrow K^n \tag{4.1}$$

is a map on a suitable space K^n. In this thesis, we take $K = \{0, 1\}$ unless stated otherwise. System states are denoted by $x = (x_1, \ldots, x_n) \in K^n$.

The focus of the work in this thesis is on systems of the form (4.1) that are constructed from four components. The first is a finite graph X with vertex set V and edge set E. Each vertex v has a state x_v from a finite set K, which is the second component. Thirdly, for each vertex v there is a vertex function f_v that is used to map $x_v(t)$ to $x_v(t+1)$ based on the vertex states in the 1-neighborhood of v (including x_v) and thus governs the *local dynamics*. Finally, an update method that governs how the vertex functions assembly to a map F of the form (4.1) to produce the *global dynamics*. From the point of view of modeling, the

vertex state captures the state of a particular agent such as the health state in an epidemic model. The vertex function encodes the evolution of the agent's state as a function of its own state and those of its neighbors in the graph X. The update scheme governs how the collection of agent states is updated collectively in a time step. As a matter of fact, GDS represent a powerful mathematical and computational model for network-based simulation models, applicable to for example epidemic spread on social contact networks [31, 53], transportation system modeling with vehicles and humans as agents [2, 6] and systems in computational biology [21].

A general theme in the research of finite discrete dynamical systems is the stability of system dynamics with respect to system components and initial conditions. Assuming that one has a precise mathematic model, one needs an additional notion to address V & V: *equivalence*, which evaluates the similarity of different system dynamics. For a given measure, stability assessments will allow one to analyze system robustness under perturbations of system components including (i) vertex state, (ii) vertex function, (iii) graph structure and (iv) vertex state update scheme. In this chapter, we summarize our research results on stability theory of graph dynamical systems, which serves as an important motivation and theoretical guidance for the design of GENEUS system. All results are published work, therefore, we only summarize the main propositions, theorems, and corollaries here, while the detailed proofs are referred to our published papers [4, 5, 7, 166]. In Section 4.5, we present how to use the GENEUS framework to fast configuring a GDS computational tool and exploring a large system configuration space. At the end of this section, we also give an example application of GENEUS to a theoretical GDS problem.

4.1.2 General Terminology

We first introduce the necessary definitions and terminology associated the GDS model and its extensions. A *graph dynamical system* is specified as a four-tuple $S = (G, K, F, W)$, where $G(V, E)$ is an undirected *graph* with vertex set V and edge set E representing the underlying network, and we assume that G has n vertices (also referred to as nodes); K is a finite domain denoting the *vertex state space*. Here we consider $K = \{0, 1\}$ in this thesis; Next, $F = \{f_1, f_2, \ldots, f_n\}$ is a collection of *functions*, with $f_i \colon K^n \longrightarrow K$ being the *transition function* associated with vertex v_i for $1 \leq i \leq n$. Note that f_i will in general depend nontrivially only on the states of vertex i's closed 1-distance neighborhood denoted by $n[i]$, and let $x[i] \in \prod_{i' \in n[i]} K_{i'}$ be the corresponding projection of the system state $x = (x_1, x_2, \ldots, x_n)$ onto $n[i]$. Hence, we note that this type of transition function captures the local interaction implied by the underlying graph structure; finally, the *update scheme* W governs how the functions f_i are composed to produce the dynamics of the

system as a whole. For example, in the *synchronous update* scheme, all vertices compute their local function value and update their state synchronously at each time step. For the *sequential update scheme*, a total order $w = (w_1, w_2, \ldots, w_n) \in W$ is specified. The vertices compute their local function values and update their states in the order specified by w. With above terminology, we have the following definition.

Definition 4.1. *The sequential graph dynamical system map* $F_w \colon K^n \longrightarrow K^n$ *is the composition*

$$F_w = F_{w_n} \circ F_{w_{n-1}} \circ \cdots \circ F_{w_2} \circ F_{w_1} \ . \tag{4.2}$$

The associated synchronous graph dynamical system map is defined by

$$F(x_1, x_2, \ldots, x_n) = \big(f_1(x[1]), f_2(x[2]), \ldots, f_n(x[n])\big) \ . \tag{4.3}$$

A *system state* (also called *configuration* in some literature) x of a GDS is an n-vector $(x_1, x_2, \ldots, x_n)$, where $x_i \in K$ is the value of vertex v_i for $1 \leq i \leq n$. If a GDS has a (one step) transition from system state x to system state x', we say that x' is a *successor* of x and x is a *predecessor* of x'.

For graph dynamical system map F, the *phase space* of F with map $\phi \colon K^n \longrightarrow K^n$ is a directed graph $\Gamma(\phi)$ with vertex set K^n (all possible system states) and all edges $(x, \phi(x))$ for $x \in K^n$. A state on a cycle in $\Gamma(\phi)$ is a *periodic point* of ϕ, whereas a state on a cycle of length one is a *fixed point*. The set of such points are denoted by $\mathrm{Per}(\phi)$ and $\mathrm{Fix}(\phi)$, respectively. We also refer to $\omega(x)$, the *limit set* of $x \in \{0,1\}^n$, which is the set of periodic points reachable from x. It is worth noting that the phase space can capture the system dynamic of a particular GDS. However, for a GDS over a domain K, where the underlying graph has n vertices, the number of nodes in the phase space is $|K|^n$, therefore, exhaustively exploring the phase space structure is usually not feasible. In this chapter, we will give our major contributions on the stability of the phase space structure for several classes of graph dynamical systems.

4.2 Dynamic Bi-threshold GDS

4.2.1 Background and Motivation

Many real-world phenomena can be described by threshold functions. For example, if the contact duration of two individuals reaches a specific time, disease transmission can happen; if the number of one person's friends who believe an opinion reaches a particular threshold, this person will be influenced and tend to adopt the opinion. As a matter of fact, this class of functions is widely adopted in modeling biological systems [46, 89, 91], social

behaviors [41, 94], pandemic spread [150], and pest control [29]. For many applications, the threshold values for a vertex state transition from 0 to 1 and from 1 to 0 are not necessarily the same. For example, a smoker tends to gain addiction easier than to quit smoking. Therefore, in [99], Kuhlman et al extended the standard Boolean threshold function to *bi-threshold function* $\theta_{v,k^\uparrow,k^\downarrow}(x_1,\ldots,x_N)\colon \{0,1\}^N \longrightarrow \{0,1\}$ defined by

$$\theta_{v,k^\uparrow,k^\downarrow}(x_1,\ldots,x_N) = \begin{cases} 1, & \text{if } x_v = 0 \text{ and } \sum_{j=1}^N x_j \geqslant k^\uparrow \\ 0, & \text{if } x_v = 1 \text{ and } \sum_{j=1}^N x_j < k^\downarrow \\ x_v, & \text{otherwise,} \end{cases} \tag{4.4}$$

where the integers $k^\uparrow$ and $k^\downarrow$ are called the *up- and down-thresholds*, respectively. Here v is a designated vertex and $N = d(v)+1$ where $d(v)$ is the *degree* of v in X. Note that one will get the standard threshold system when the up- and down- thresholds are the same. In our work, we further extend the bi-threshold system to a more general system: the up- and down-thresholds may depend on the vertex and we write $(k_v^\uparrow)_v$ and $(k_v^\downarrow)_v$ for the vertex-indexed sequence of up- and down-thresholds. Note that the threshold values will evolve through time instead of stay static. We call a graph dynamical system with such time-varying bi-threshold vertex function *dynamic bi-threshold system*.

The motivation for introducing this type of system is that for many applications, the up- and down-thresholds are not static, but are instead governed by the dynamics itself. In the case of epidemics such as malaria [150], for example, the acquired partial immunity will generally increase upon exposure to the parasite. From the point of view of modeling, this can be captured through an increase in the threshold $k^\uparrow$ upon the transition from susceptible (S or 0) to infected (I or 1).

On the other hand, from the view of model validation, the dynamical bi-threshold system provides a more flexible model that can be calibrated. One can view the change of the up- and down-threshold values as a type of function perturbation, and our major contribution in this research is to give the sufficient condition on the dynamics of the up- and down-threshold that ensure the stability of synchronous and sequential system. This work is published in [166], we will first give some necessary definitions in Section 4.2.2 and then summarize the results in Section 4.2.3.

4.2.2 Preliminaries

In contrast to maps F whose vertex functions are static bi-threshold functions as in (4.4), we here consider systems where the up- and down-thresholds $k_v^\uparrow$ and $k_v^\downarrow$ of each vertex v

may evolve with time. Let $T_v^{\uparrow}$ and $T_v^{\downarrow}$ be the set of permissible thresholds defined by

$$T_v^{\uparrow} = \{a_v, a_v + 1, a_v + 2, \ldots, b_v\} \text{ and } T_v^{\downarrow} = \{a'_v, a'_v + 1, a'_v + 2, \ldots, b'_v\} \ . \tag{4.5}$$

for integers $a_v \geq 0$, $a'_v \geq 1$, $b_v \leq d(v) + 1$ and $b'_v \leq d(v) + 2$. We include both thresholds in the state of each vertex which gives us vertex states of the form $(x_v, k_v^{\uparrow}, k_v^{\downarrow})$. For the case where the up- and down-threshold values change only upon a transition of x_v, we also keep track of how many times x_v has transitioned from 0 to 1 or from 1 to 0. For this reason, we include a *transition index* in the vertex state. In this paper, we therefore consider vertex states of the form

$$M_v = \{0, 1\} \times T_v^{\uparrow} \times T_v^{\downarrow} \times T \tag{4.6}$$

where T is a suitable space capturing the transition index. For convenience, we also introduce $M_v^* = \{0, 1\}^{d(v)+1} \times T_v^{\uparrow} \times T_v^{\downarrow} \times T$. We write vertex states as $s_v = (x_v, k_v^{\uparrow}, k_v^{\downarrow}, \tau_v)$. The *system state* is $s = (s_1, s_2, \ldots, s_n) \in M = \prod_v M_v$. We can now define the notion of *dynamic bi-threshold systems*:

Definition 4.2. *A dynamic bi-threshold system is a GDS map where each vertex function is of the form* $f_v \colon \prod_{v' \in n[v]} M_{v'} \longrightarrow M_v$ *and defined by*

$$f_v(s[v]) = (\theta_{v,k_v^{\uparrow},k_v^{\downarrow}}(x[v]),\ g_v(x[v], k_v^{\uparrow}, k_v^{\downarrow}, \tau_v),\ h_v(x[v], k_v^{\uparrow}, k_v^{\downarrow}, \tau_v))$$

for some functions $g_v \colon M_v^* \longrightarrow T_v^{\uparrow} \times T_v^{\downarrow}$ *and* $h_v \colon M_v^* \longrightarrow T$ *governing the evolution of thresholds and transition index, respectively.*

We will consider the case where g_v and h_v only depend on $x[v]$ through x_v and $\theta_{v,k_v^{\uparrow},k_v^{\downarrow}}(x[v])$.

Transition-based dynamic threshold systems. We start with some preliminary observations. First, the state of any vertex v initially transitions as $\ldots, 1, 0, 1, 0, 1, \ldots$ along every trajectory. By this we do not imply that the vertex state transitions in every iteration, rather we only record *when* a transition occurs. Eventually a vertex state may become fixed as the orbit may reach a fixed point or a periodic orbit where x_v is constant. If we assume that the thresholds $k_v^{\uparrow}$ and $k_v^{\downarrow}$ may only change upon a transition of x_v we obtain the *transition-based dynamic threshold systems*. For this class of systems one may pre-compute the thresholds $k_v^{\uparrow}$ and $k_v^{\downarrow}$ along a trajectory. The transition number may also be pre-computed from its initial value. Clearly, this depends on the fact that x_v can only take on values 0 and 1. If the vertex state space has size 3 or larger, then this is no longer that simple.

For transition-based system, the function h_v would typically return $\tau_v + 1$ whenever the state of vertex v transitions and would return τ_v otherwise. Note that we may also use iteration number (time) as transition index. In this case, the function h_v would always return $\tau_v + 1$. The above definition thus captures non-autonomous graph dynamical

systems. In some cases we may disregard the transition index altogether. If that is the case, we will simply set $T = \{0\}$. When we do consider the transition-index, we will generally be concerned with trajectories starting at $\tau_v = 0$.

4.2.3 Results

Preliminary observation

In this section, we consider the long-term dynamics of GDS maps with dynamic bi-threshold functions under parallel and sequential update methods. Our first result is concerned with the case when limit sets of dynamic and static bi-threshold systems coincide.

Proposition 4.3. *Let F be a dynamic bi-threshold system. If both components $k_v^{\uparrow}$ and $k_v^{\downarrow}$ of each function g_v in Definition 4.2 are monotone along every trajectory, then every limit set of F coincides with that of a static bi-threshold GDS map with the same update scheme.*

Under the stated conditions, the up-threshold of each vertex is either monotonically non-increasing or monotonically non-decreasing. The same holds for the down-thresholds. Since the threshold state spaces $T_v^{\uparrow}$ and $T_v^{\downarrow}$ are finite, and the up- and down-thresholds are bounded, there exists an integer $r \geq 0$ for any state s such that the up- and down-thresholds are constant on the set $\{F^{r+t}(s)\}$ for $t \geq 0$. This in turn implies that the dynamics of F eventually coincides with that of a static bi-threshold system. This essentially proves Proposition 4.3.

Even though it may be clear, we remark that the static bi-threshold GDS maps referenced in the proof generally depends on the particular state s, or more precisely, the orbit through s.

For the class of transition based dynamic threshold systems, we may assume, without loss of generality, that $k_v^{\uparrow}$ only changes upon the transition where x_v is mapped from 0 to 1. This is clear since $k_v^{\uparrow}$ is of no relevance when $x_v = 1$. The same holds for $k_v^{\downarrow}$ for transitions where x_v is mapped from 1 to 0. Consequently, for maps g_v of the form

$$g_v(x[v], k_v^{\uparrow}, k_v^{\downarrow}, \tau_v) = \begin{cases} (k_v^{\uparrow} + \Delta_{01}^{\uparrow}, k_v^{\downarrow}), & \text{if } x_v \text{ is mapped from 0 to 1,} \\ (k_v^{\uparrow}, k_v^{\downarrow} + \Delta_{10}^{\downarrow}), & \text{if } x_v \text{ is mapped from 1 to 0,} \\ (k_v^{\uparrow}, k_v^{\downarrow}), & \text{otherwise,} \end{cases}$$

where $\Delta_{01}^{\uparrow}$ and $\Delta_{10}^{\downarrow}$ are constants, we may apply Proposition 4.3 to conclude that the long-term dynamics coincide with static bi-threshold systems.

Limit sets of synchronous dynamic bi-threshold systems

The following theorem characterizes the periodic orbit size when we project from the full state $((x)_v, (k_v^{\uparrow})_v, (k_v^{\downarrow})_v, (\tau_v)_v)$ onto the $x = (x_v)_v$-component for synchronous dynamics.

Theorem 4.4. *Let F be a synchronous, transition-based dynamic neural network map with constant transition-index. Then the projection of every periodic orbit of F onto its $x = (x_1, \ldots, x_n)$-component has period at most two. In particular, if $k_v^{\uparrow}$ or $k_v^{\downarrow}$ has period $p > 2$ on a periodic orbit, then the x-component on this orbit is fixed.*

The proof is basically a strengthening of Theorem 3.1 in [99] which in turn extended a result of Goles and Olivos [62] on neural networks. The detailed proof is provided in our published paper [166] and thus omitted here for the sake of brevity.

Stability of sequential dynamic bi-threshold systems

In transition-based dynamic threshold systems, if x_v can take on only two values, $k_v^{\uparrow}$ and $k_v^{\downarrow}$ can be pre-computed along a trajectory as a function of the transition index τ_v, which we may denote by $k_v^{\uparrow}(\tau_v)$ and $k_v^{\downarrow}(\tau_v)$. For this class of threshold systems we may without loss of generality assume that $k_v^{\uparrow}$ only changes upon the transition where x_v is mapped from 0 to 1. This is clear since $k_v^{\uparrow}$ is of no relevance when $x_v = 1$. The same holds for $k_v^{\downarrow}$ for transitions where x_v is mapped from 1 to 0. Therefore, the map g_v can be expressed as

$$g_v(x[v], k_v^{\uparrow}, k_v^{\downarrow}, \tau_v) = \begin{cases} (k_v^{\uparrow}(\tau_v + 1), k_v^{\downarrow}(\tau_v)), & \text{if } x_v \text{ is mapped from 0 to 1,} \\ (k_v^{\uparrow}(\tau_v), k_v^{\downarrow}(\tau_v + 1)), & \text{if } x_v \text{ is mapped from 1 to 0,} \\ (k_v^{\uparrow}(\tau_v), k_v^{\downarrow}(\tau_v)), & \text{otherwise.} \end{cases} \tag{4.7}$$

Theorem 4.5. *Consider a transition-based dynamic SDS. If there exists a $\tau_{\max}$ such that for all vertices v and $\tau \geq \tau_{\max}$, $\Delta_v(\tau) = k_v^{\downarrow}(\tau) - k_v^{\uparrow}(\tau) < 2$ then, it has only fixed points as limit sets and from any initial configuration, a fixed point is reached in at most $(n\tau_{\max} + 1)(3m + 2n + 1)$ time steps where, m and n are the numbers of edges and vertices in the underlying graph X respectively.*

The proof of Theorem 4.5 uses a method involving potential functions to study the dynamics of SDS. This was introduced by Barrett et al. [23] for bounding the transient lengths in the case of standard threshold sequential dynamical systems. Since then, this technique has been used to characterize the dynamics of general static bi-threshold systems [99] and block-sequential systems [118], but see also [66]. Here we also omit the detailed proof which is given in [166].

4.2.4 Discussion

In this section we show our result published in [166], where we have extended bi-threshold systems to include dynamically evolving thresholds. This provides a more realistic venue for modeling complex phenomena, such as adaptive biological, social and technical systems. We have shown the remarkable result that synchronous, transition-based, dynamic bi-threshold GDS maps may only have periodic orbits of size 1 or 2.

It is worth noting that this extension of dynamic bi-threshold system can be viewed as a perturbation to the vertex functions, we derived sufficient stability conditions with respect to this type of perturbations such that the GDS maps may only have fixed points or periodic orbits of length 2.

4.3 Graph Structure and Update Scheme

4.3.1 Background and Motivation

As mention in Section 4.1, in the synchronous update scheme, every vertex function is applied simultaneously, while in a sequential update scheme, vertices are updated one by one according to a total order defined on the vertex set. A generalization of these schemes is the *block-sequential update*. Here, the vertices are partitioned into blocks, and vertices within the blocks are updated synchronously while the blocks themselves are updated sequentially.

Two interesting questions which have been repeatedly addressed in the past are: given a graph dynamical system, (i) what is the maximum possible length of a limit cycle? (ii) what conditions lead to only fixed points as limit sets? There are several notable previous results concerning synchronous and sequential threshold systems. Goles and Olivos [62] and Barrett et al. [23] independently, using different methods, showed that sequential threshold systems exhibit only fixed points as limit cycles. In [62, 63], it was shown that for synchronous update there can be limit cycles of length at most two. Their result is applicable to the more general case of weighted threshold functions. Kuhlman et al. [99] considered these questions regarding bi-threshold systems. They showed that, while synchronous systems can have limit cycles of length at most two, sequential systems can have arbitrarily long limit cycles.

Motivated by these results, we study how the underlying graph structure influences the limit cycle structure of block-sequential systems. Mortveit [118] showed that if the blocks

are of size at most 3, then there will be only fixed points. The author also conjectured that the limit cycle length can be at most two for arbitrary block size. However, this was disproved recently by Goles and Montealegre [65]. Unlike the sequential or synchronous cases, these systems can have arbitrarily long limit cycles. In [63], the more general setting of weighted threshold functions were studied. They gave a sufficient condition for the system to have only fixed points.

We derive a sufficient condition on the graph structure so that the system has only fixed points as limit cycles. We also identify several well-known graph families that satisfy this condition. Related to the main theme of this thesis, this work actually investigates how graph structure and update scheme will affect the stability of the system dynamics because fixed points are steady states in the phase space that once reached will never leave. In this Section, we summarize our results published in [5] and illustrate our research contribution in block-sequential threshold systems.

4.3.2 Preliminaries

Let X be a simple undirected graph on n vertices with vertex set $V[X]$ and edge set $E[X]$. Let $\mathcal{B} = \{B_1, B_2, \ldots, B_m\}$ be a block partition of $V[X]$. For $S \subseteq V[X]$, let $\deg_S(v)$ denote the number of neighbors of v in the graph induced by S, and let $\deg(v)$ be its degree in X. Recall that x_v denotes the *vertex state* of v. Since we are considering Boolean systems, $x_v \in \{0, 1\}$. Let $x = (x_1, x_2, \ldots, x_n)$ be the system state. Let $n[v]$ denote the sorted sequence of the closed neighborhood of v, and let $x[v]$ denote the restriction of x to $n[v]$.

Every vertex is assigned a *threshold function* $f_v : \{0, 1\}^{\deg(v)+1} \longrightarrow \{0, 1\}$ defined as follows:

$$f_v(x[v]) = \begin{cases} 1, & \text{if } \sum_{v' \in n[v]} x'_v \geq k_v, \\ 0, & \text{otherwise,} \end{cases} \tag{4.8}$$

where $k_v \in \mathbb{N}$ with $k_v \geq 1$ is the *threshold* of v. For a block B and system state x, the map $F_B(x) : \{0, 1\}^n \longrightarrow \{0, 1\}^n$ is given by

$$(F_B(x))_v = \begin{cases} f_v(x[v]), & \text{if } v \in B, \\ x_v, & \text{otherwise.} \end{cases} \tag{4.9}$$

The block-sequential map $F : \{0, 1\}^n \longrightarrow \{0, 1\}^n$ is defined as

$$F = F_{B_m} \circ F_{B_{m-1}} \circ \cdots \circ F_{B_1}\,. \tag{4.10}$$

The two special cases of the block-sequential update scheme are sequential and parallel update schemes. The sequential update corresponds to each vertex belonging to a distinct

block, i.e., for $i = 1, \ldots, n$, $|B_i| = 1$ and therefore, $m = n$. In the parallel update, there is only one block, i.e., $m = 1$ and $B_1 = V[X]$.

4.3.3 Results

Main result

In this work [5], we examine standard threshold systems from a structural perspective. Our main objective was to identify conditions on the underlying graph structure which lead to only fixed points. Our main result is given below.

Theorem 4.6. *Let X be a simple graph with vertex set $V[X]$ and edge set $E[X]$. Let $\mathcal{B}$ be a block partition of $V[X]$. If every block $B \in \mathcal{B}$ satisfies Condition (4.11) below, then, any block-sequential standard threshold system induced by $\mathcal{B}$ for any update order on the blocks has only fixed points as limit sets. Also, the transient length is at most $(|E[X]| + |V[X]| + 1)/2$.*

$$\text{For any non-empty } B' \subseteq B \text{ and any assignment } y \text{ of vertex states for } B',\ \|B'\| - 2|\Lambda_{B'}(y)| - |B'| < 0, \text{ where, } \|B'\| \text{ is the number of edges in the subgraph induced by } B' \text{ and } \Lambda_{B'}(y) = \{\{u, v\} \in E[X] \mid u, v \in B', \text{ and } y_u = y_v\}. \tag{4.11}$$

An interesting feature of Theorem 4.6 is that Condition (4.11) only applies to the individual blocks and is independent of the connections between the blocks. The proof uses the potential function argument introduced in [23]. We build on the framework provided by [118] and extend the results of that paper. We note that Condition (4.11) is not a necessary condition. Consider any graph with arbitrary block partition where each vertex has threshold 1. This is a progressive threshold system, i.e., a vertex will never transition from 1 to 0. Hence, it has only fixed points even though the blocks may not satisfy Condition (4.11).

Block decomposition

In the graph theory literature, a *block* is a maximal connected subgraph without a cut vertex [50]. Every block can either be a maximal 2-connected subgraph, an edge, or an isolated vertex. Since the term "block" has already been used to mean something else in this paper, we will henceforth refer to maximal connected subgraphs as *subblocks*. Every graph can be decomposed into subblocks. Since they satisfy maximality, any two subblocks overlap in at most one vertex, which, if it exists, is a cut vertex of the graph. This is illustrated with an example in Figure 4.1(a). Let C be the set of cut vertices and $\mathcal{S}$ be the

set of subblocks. The *block graph* is the bipartite graph on the vertex set $C \cup \mathcal{S}$ where for $c \in C$ and $S \in \mathcal{S}$, $\{c, S\}$ is an edge if and only if $c \in S$. It can be easily shown that the block graph is a tree. See Figure 4.1(b) for the block graph of the example.

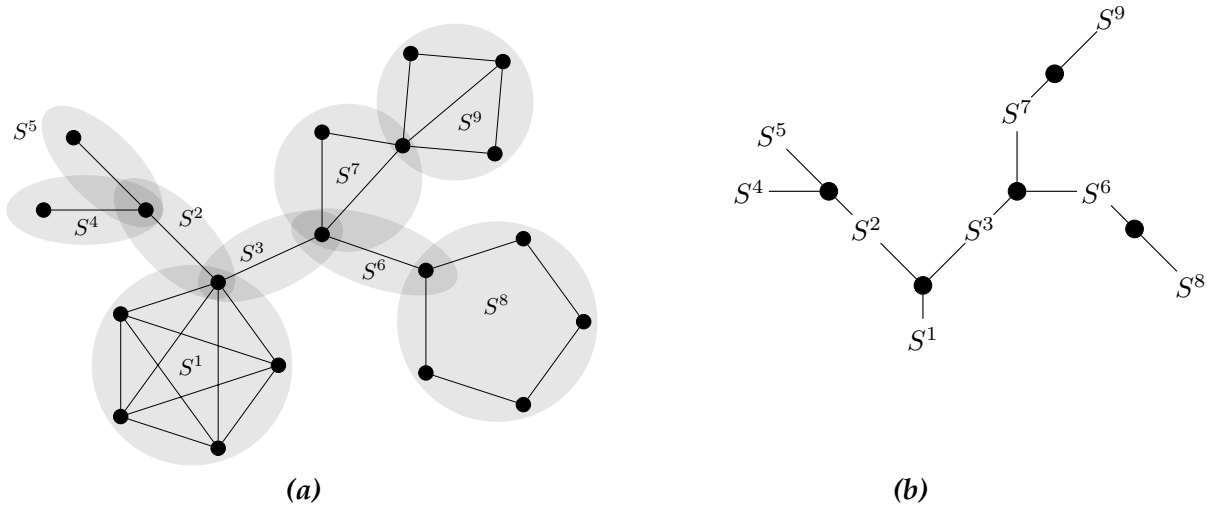


Figure 4.1: *An example of (a) a block decomposition and (b) the corresponding block graph.*

Theorem 4.7. *Let block B be such that all of its subblocks satisfy Condition* (4.11)*. Then, B satisfies Condition* (4.11) *too.*

Simple graph classes which satisfy Condition (4.11)

We will show that some graph classes such as trees, odd cycles, and complete graphs satisfy Condition (4.11). Even though these are very simple graphs, to the best of our knowledge, these results have not been obtained before using any other method. Throughout this section, B corresponds to a block in X and $B' \subseteq B$. The collection of graph classes are given by following propositions.

Proposition 4.8. *If B induces following graph classes in X, then, it satisfies Condition* (4.11)*:*

(i) Tree.
(ii) Odd cycle.
(iii) Clique.
(iv) Block B of size 4 *other than the* 4*-cycle, see Figure 4.2.*
(v) Wheel graph with odd cycle.

Not that a wheel graph is formed by connecting a single vertex to all vertices of a cycle.

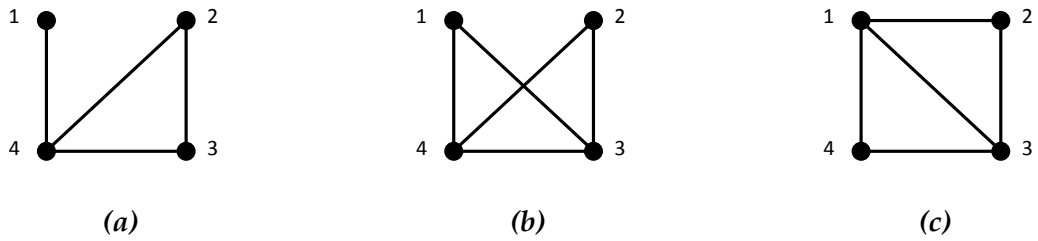


Figure 4.2: *Possible connected graphs (up to isomorphism) of size* 4 *excluding trees and* 4*-cycle.*

Remark 4.9. *Note that one can configure an example corresponding to a* 4*-cycle* (C_4) *for which not satisfies Condition* (4.11) *(See Figure* 4.3*(a)). Moreover, this configuration corresponds to a limit cycle of length* 2*. So, a natural question to ask is whether graphs which do not have a* C_4 *as a vertex-induced subgraph not satisfy Condition* (4.11) *for all* x*. Unfortunately, the answer is no. Figure* 4.3*(b) is an example where an induced-*C_4*-free graph has a limit cycle of length two.*

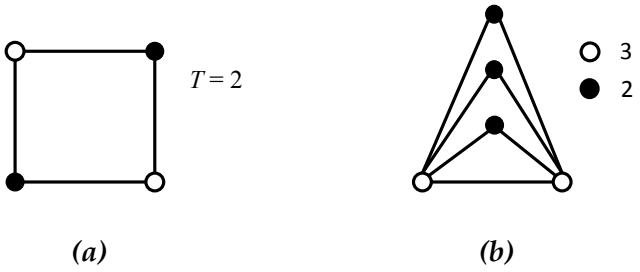


Figure 4.3: *Configurations lead to limit cycles of length two: if the black vertices are in state* 0*, then the white are in* 1*, and vice versa. (a) Block* C_4 *where all vertices have threshold* 2 *and (b) an induced-*C_4*-free graph with the black vertices having threshold* 2 *and white vertices* 3*.*

Remark 4.10. *Note that there exists a wheel graph with an even cycle corresponding to a limit cycle of length* 2*, see Figure* 4.4*. In this example, suppose the threshold value of the central vertex is* 3 *and all other vertices have threshold value* 2*. Then, one can verify that the central vertex will remain* 0 *and the other vertices will change states alternatively in pairs, i.e. this configuration leads to a length* 2 *limit cycle.*

4.3.4 Discussion

In [5], we studied the limit cycle structure of standard threshold dynamical systems with block-sequential update. We identified a sufficient condition for the system to have only

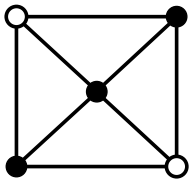


Figure 4.4: *Configuration over a wheel graph with an even cycle which leads to limit cycle of length* 2. *Black vertices are in state* 0 *and white are in state* 1. *The threshold value for the central vertex is* 3, *and* 2 *for other vertices.*

fixed points as limit sets. There are several possibilities to consider for the future. Even though the condition depends only on the blocks and not the graph as a whole, it seems to be restrictive. One direction to explore is to find more general conditions which take into account edges between the blocks too. Another direction would be to look at bi-threshold systems.

It is worth noting that this study actually examines the stability criteria under the perturbation to the GDS graph structure and update scheme. It gives sufficient conditions such that graph dynamical systems with particular graph structures will only have fixed points as limit sets.

4.4 Activity and Sensitivity of Graph Dynamical Systems

4.4.1 Background and Motivation

The study of stability and the response to perturbations of graph dynamical systems is central to increased understanding of their dynamical properties. We discussed effects of perturbations of the graph structure, vertex functions and update scheme in above sections. In this section, we summarize our work in [4], where we consider noise applied to vertex states. Specifically, we want to know the following:

What is the probability that $F(x)$ and $F(x + e_i)$ are different?

Here i is a vertex while e_i is the i^{th} unit vector with the usual addition modulo 2. In [145], Shmulevich and Kauffman considered Boolean networks over regular graphs with $f_j = f$ for all vertices j, that is, a common vertex function. They defined the notion of *activity* of f with respect to its i^{th} argument as the expected value of the Boolean derivative of

f with respect to its i^{th} variable. Under their assumptions, this may give a reasonable indication of the expected impact of perturbations to the i^{th} variable under the evolution of F. However, this approach does not consider the impact of the dependency graph structure. We remark that Layne et al. compute the activities of nested canalyzing functions given their canalyzing depth, extending results in [145] on canalyzing functions, see [102].

This question of sensitivity has also been studied when x is restricted to attractors in order to assess the stability of long-term dynamics under state noise. The notion of threshold ergodic sets (TESs) is introduced in [135] and studied further in [100, 106]. The structure of TESs captures long-term stability under state perturbations of periodic orbits and the resulting mixing between attractors that may happen as a result.

Other tools for analyzing the sensitivity of vertex noise includes Lyapunov exponents, see for example [16, 17], although this is perhaps mostly relevant or suited for the infinite case such as cellular automata over (infinite) regular lattices. Also, the notion of *Derrida diagrams* has been used to quantify how *Hamming classes* of states separate on average [48, 58] after one or k transitions under F. Derrida diagrams, however, are mainly analyzed through numerical experiments via sampling. Moreover, analyzing how Hamming classes of large distance separate under F may not be so insightful – it seems more relevant to limit oneself to the case of nearby classes of vertex states.

Returning to the original question, we note that $F(x)$ and $F(x + e_i)$ may only differ in the coordinates j for which f_j depends on x_i. The answer to the question therefore depends on the vertex functions in the 1-neighborhood of i in the dependency graph X, and therefore the structure of the induced subgraph of the 2-neighborhood of i in X with the omission of edges connecting pairs of vertices both of distance 2 from i. We denote this subgraph by $X(i; 2)$, see Figure 4.5.

To determine the activity of vertex i one will in general have to evaluate F over all possible states of $X(i; 2)$, a problem which, in the general case, is computationally intractable. We will address three cases in this paper: the first case is when $X(i; 2)$ is a tree, the second case is that of elementary cellular automata (a special case of the former case), and the case where X is a regular, square, 2-dimensional lattice. Work for other graph classes is in progress and we comment on some of the challenges in the Summary section. Here we remark that a major source of challenges for analytic computations is the introduction of *type-3* and *type-4* edges as illustrated in Figure 4.5. Lack of symmetry adds additional challenges. The activity of a vertex can be evaluated analytically using the inclusion-exclusion principle, and type-3 and type-4 edges impact the complexity of the combinatorics.

A goal of the work on activity started here is as follows: when given a network X and vertex functions $(f_i)_i$, we would like to rank the vertices by activity in decreasing order.

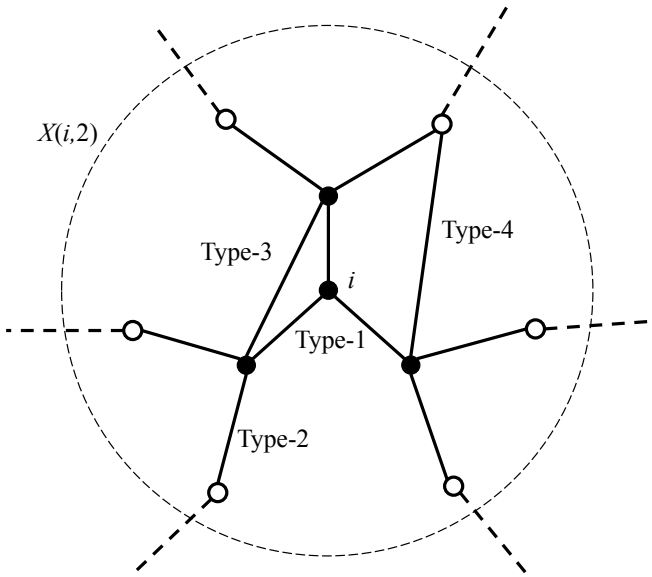


Figure 4.5: *The subgraph $X(i;2)$ of X induced by vertex i and its distance ≤ 2 neighbors. Vertices belonging to the closed 1-neighborhood $n[i]$ of i are marked black. Type-3 edges (relative to i) connect neighbors of i, while type-4 edges connect neighbors of $j \in n'[i]$ through a common neighbor different from i. Here $n'[i]$ is $n[i]$ with i omitted. Edges connecting vertices of distance 2 from i in X do not belong to $X(i;2)$.*

Just being able to identify for example the ten (say) vertices of highest activity would also be very useful. This information would allow one to identify the vertices for which state perturbations are most likely to produce different outcomes, at least in the short term. We will first introduce necessary definition and then summarize our main result in [4].

4.4.2 Preliminaries

In [4] we analyze *short-term stability of dynamics* through the function $\alpha_{F,i}\colon K^n \longrightarrow \{0,1\}$ defined by

$$\alpha_{F,i}(x) = \mathbb{I}[F(x+e_i) \neq F(x)] \tag{4.12}$$

where $\mathbb{I}$ is the indicator function and e_i is the i^{th} unit vector with $1 \leq i \leq n$. In other words, $\alpha_{F,i}(x)$ measures whether perturbing x by e_i results in a different successor state under F than $F(x)$.

Definition 4.11. *The* activity of F with respect to vertex i *is the expected value of* $\alpha_{F,i}$ *using the*

uniform measure on K^n*:*

$$\bar{\alpha}_{F,i} = \mathbb{E}[\alpha_{F,i}] \ . \tag{4.13}$$

The activity of F *is the vector*

$$\bar{\alpha}_F = (\bar{\alpha}_{F,1}, \bar{\alpha}_{F,2}, \ldots, \bar{\alpha}_{F,n}) \ , \tag{4.14}$$

while the sensitivity of F *is the average activity* $\bar{\alpha} = \sum_{i=1}^{n} \bar{\alpha}_{F,i}/n$.

For a randomly chosen state $x \in K^n$, the value $\bar{\alpha}_{F,i}$ may be interpreted as the probability that perturbing x_i will cause $F(x + e_i) \neq F(x)$ to hold. This activity notion may naturally be regarded as a measure of sensitivity with respect to initial conditions.

From (4.12) it is clear that $\bar{\alpha}_{F,i}$ depends on the functions f_j with $j \in n[i]$ and the structure of the distance-2 subgraph $X(i;2)$, see Figure 4.5. The literature (see, e.g. [102, 145]) has focused on a very special case when considering activity. Rather than considering the general case and Equation (4.12), they have focused on the case where X is a regular graph where each vertex has degree d and all vertex functions are induced by a common function $f : K^{d+1} \longrightarrow K$ through $f_v(x[v]) = f(x[v])$. In this setting, activity is defined with respect to f and its i^{th} argument, that is, as the expectation value of the function $\mathbb{I}[f(x + e_i) \neq f(x)]$ where $x \in K^{d+1}$. Clearly, this measure of activity is always less than or equal to $\mathbb{E}[\alpha_{F,i}]$. Again, we note that this simpler notion of activity does not account for the network structure of $X(i;2)$.

4.4.3 Results

Preliminary results

For the evaluation of $\bar{\alpha}_{F,i}$ we introduce some notation. In the following we set $K = \{0, 1\}$, write N_i for the size of $X(i;2)$, and $K(i) = K^{N_i}$ for the projection of K^n onto the set of vertex states associated to $X(i;2)$. For $j \in n[i]$, define the sets $A_j(i) \subset K(i)$ by

$$A_j(i) = \{x \in K(i) \mid F(x + e_i)_j \neq F(x)_j\} \ .$$

These sets appear in the evaluation of $\bar{\alpha}_{F,i}$, see Proposition 4.12. For convenience, we also set

$$A_j^m(i) = \{x \in A_j(i) \mid x_i = m\} \ ,$$

for $m = 0, 1$. We write $\bar{A}_j(i) = A_j^0(i)$ and $\binom{n}{k}$ for binomial coefficients using the convention that it evaluates to zero if either $k < 0$ or $n - k < 0$.

The following proposition provides a somewhat simplified approach for evaluating $\bar{\alpha}_{F,i}$ in the general case.

Proposition 4.12. *Let X be a graph and F a map over X as in (4.1). The activity of F with respect to vertex i is*

$$\bar{\alpha}_{F,i} = \Pr(\bigcup_{j\in n[i]} A_j \mid x_i = 0) = \Pr(\bigcup_{j\in n[i]} \bar{A}_j) \ . \tag{4.15}$$

Proposition 4.12 actually gives a way to calculate the activity with respect to an individual vertex and the evaluation of Equation (4.15) can often be done through the inclusion-exclusion principle. As an example, we have following proposition for complete graph.

Proposition 4.13. *If F is the Boolean network induced by the* nor*-function over K_n, where K_n is a complete graph with n vertices, then*

$$\bar{\alpha}_{F,i} = \frac{1}{2^{n-1}},$$

and if F is induced by the k-threshold function over K_n, then

$$\bar{\alpha}_{F,i} = \binom{n-1}{k-1} \Big/ 2^{n-1} \ .$$

In the computations to follow, we will frequently need to evaluate the probability of the union of the A_j's. For this, let B denote the union of A_j's for all $j \neq i$. We then have

$$\Pr\Big[\bigcup_{j\in n[i]} A_j\Big] = \Pr(A_i \cup B) = \Pr(B) + \Pr(A_i \cap B^c) = 1 - \Pr(B^c) + \Pr(A_i \cap B^c) \ , \tag{4.16}$$

where B^c denotes the complement of B.

Activity of elementary cellular automata

Let F be the ECA map over $X = \text{Circle}_n$ with vertex functions given by $f : \{0,1\}^3 \longrightarrow \{0,1\}$, where Circle_n is a *circle graph on n vertices* . We will assume that $n \geq 5$; the case $n = 3$ corresponds to the complete graph K_3 and the case $n = 4$ can be done quite easily. Here we have

$$\bar{\alpha}_{F,i} = \Pr(\bar{A}_{i-1}(i) \cup \bar{A}_i(i) \cup \bar{A}_{i+1}(i)) \ .$$

Applying the definitions,

$$\bar{A}_j(i) = \{x = (x_{i-1}, x_{i-1}, x_i = 0, x_{i+1}, x_{i+2}) \mid \text{ and } f(x[j]) \neq f((x + e_i)[j])\} \tag{4.17}$$

for $j \in n[i] = \{i-1, i, i+1\}$ with indices modulo n.

Proposition 4.14. *The activity for k-threshold ECA is*

$$\bar{\alpha}_{F,i} = \begin{cases} 0, & \text{if } k = 0 \text{ or } k > 3 \ , \\ 1/2, & \text{if } k = 1 \text{ or } k = 3 \ , \\ 7/8, & \text{if } k = 2 \ . \end{cases}$$

Clearly, for $k = 0$ and $k > 3$ we always have $F(x + e_i) = F(x)$ so in these cases it follows that $\bar{\alpha}_{F,i} = 0$. The cases $k = 1$ and $k = 3$ are symmetric, and, using $k = 1$, we have

$$\bar{A}_{i-1} = \{(0,0,0,x_{i+1},x_{i+2})\}, \quad \bar{A}_i = \{(x_{i-2},0,0,0,x_{i+2})\}, \text{ and } \bar{A}_{i+1} = \{(x_{i-1},x_{i-1},0,0,0)\} \ .$$

By the inclusion-exclusion principle, it follows that

$$\begin{aligned} |\bar{A}_{i-1} \cup \bar{A}_i \cup \bar{A}_{i+1}| &= |\bar{A}_{i-1}| + |\bar{A}_i| + |\bar{A}_{i+1}| \\ &\quad - |\bar{A}_{i-1} \cap \bar{A}_i| - |\bar{A}_{i-1} \cap \bar{A}_{i+1}| - |\bar{A}_i \cap \bar{A}_{i+1}| + |\bar{A}_{i-1} \cap \bar{A}_i \cap \bar{A}_{i+1}| \\ &= 3 \times 4 - 2 - 1 - 2 + 1 = 8 \ . \end{aligned}$$

This yields $\bar{\alpha}_{F,i} = 8/2^4 = 1/2$. The proof for the case $k = 2$ is similar to that of $k = 1$ so we leave this to the reader.

Remark 4.15. *If we instead use the* nor*-function, then $\bar{\alpha}_{F,i} = 1/2$ when $n \geq 5$.*

Remark 4.16. *More generally, an ECA rule $f : \{0,1\}^3 \longrightarrow \{0,1\}$ may be represented by an integer $0 \leq r(f) \leq 255$, or simply r. Each triple (x_{i-1}, x_i, x_{i+1}) can be viewed as a binary number $0 \leq j \leq 7$ where x_{i-1} is taken as the most significant digit. Let x^j denote the triple corresponding to the integer j and set $a_j = f(x^j)$. We can then represent f as the 8-tuple $a = (a_0, a_1, \ldots, a_7)$. The associated rule number is $r(f) = \sum_{0 \leq j \leq 7} a_j \cdot 2^j$. From this and through either inclusion/exclusion or exhaustive enumeration (perhaps easier), one can obtain the activity value of all ECA through conditions on the a_i values.*

Remark 4.17. *In [145], activity is defined with respect to the function f instead of F and its i^{th} argument. With this context,*

$$\bar{\alpha}_{f,i}(x) = \mathbb{E}\Big[\mathbb{I}[f(x + e_i) \neq f(x)]\Big] \ .$$

Using their definition of activity for ECA with a function of three Boolean variables, the only possible values for activity of ECA are 0, 1/4, 1/2, 3/4, and 1. The fact that our new notion of activity gives $\bar{\alpha} = 7/8$ for threshold-2 Boolean networks demonstrates that network structure does indeed impact results.

Additionally, we always have $\bar{\alpha}_{f,i} \leq \bar{\alpha}_{F,i}$. As an example, for a circle graph of girth ≥ 5 and threshold-1 functions (i.e., $k = 1$) it can be verified that $\bar{\alpha}_{f,i} = 1/4$. It was shown above that in this case $\bar{\alpha}_{F,i} = 1/2$.

Activity over d-regular trees

A natural starting point for analyzing activity is the case where the graph X is a d-regular tree. The reason is that in this case the sets A_j (or more precisely, the sets $n[j]$) with $j \neq i$ only have vertex i in common. As a result, when we condition on $x_i = m$, the resulting sets are independent. This fact will hold if the girth of the graph is at least 5, so the results we give are applicable to this broader graph class. We remark that a d-regular tree is an infinite tree where all vertices have degree d. Alternatively, one may consider this to be finite trees where all vertices either have degree d or degree 1. In the latter case, our result only applies to vertices whose neighbors all have degree d. We have following propositions for Boolean bi-threshold function (Equation (4.4)) and norfunction.

Proposition 4.18. *Let X be a d-regular graph of girth ≥ 5, and F the Boolean network over X with the bi-threshold vertex functions as in Equation (4.4). Then the activity of F with respect to vertex i is given by*

$$\begin{aligned}\bar{\alpha}_{F,i} =& 1 - \frac{1}{2^{d^2}}\left[2^d - \binom{d-1}{k^{\uparrow}-1} - \binom{d-1}{k^{\downarrow}-2}\right]^d + \\ & \frac{1}{2^{d^2+1}} \sum_{k=k^{\uparrow},k^{\downarrow}} \binom{d}{k-1}\left[2^{d-1} - \binom{d-1}{k^{\downarrow}-2}\right]^{k-1}\left[2^{d-1} - \binom{d-1}{k^{\uparrow}-1}\right]^{d-(k-1)} .\end{aligned} \tag{4.18}$$

Proposition 4.19. *Let X be a d-regular graph of girth ≥ 5 and F the Boolean network over X induced by the* nor*-function. Then the activity of F with respect to i is given by*

$$\bar{\alpha}_{F,i} = 1 - \left(1 - \frac{1}{2^d}\right)^d + \left(\frac{1}{2} - \frac{1}{2^d}\right)^d . \tag{4.19}$$

The detailed proof of Proposition 4.18 and 4.19 can be found in [4, 8].

Activity over square lattices

In this section we consider graphs with girth-4 edges or girth 4. Here the 1-neighborhoods $n[j]$ with $j \neq i$ may intersect, the key aspect we want to address here. As an example, the reader may verify that for threshold-2 functions and Circle_4, the activity of any vertex is 3/4 and not 7/8 as when $n \geq 5$ in Proposition 4.14.

To be specific we take the graph X to be a regular, square 2-dimensional lattice. It may be either infinite or with periodic boundary conditions. In the latter case, we assume for

simplicity that its two dimensions are at least 5. This graph differs from the 4-regular tree by the introduction of type-4 edges: sets A_j and A_{j+1} of $n(i)$, when conditioned on the state of vertex i, are no longer independent. The graph has cycles of size 4 containing i. We first illustrate this case using nor-functions as these allow for a somewhat simplified evaluation as compared to what happens for threshold functions.

Proposition 4.20. *Let X be the 2-dimensional lattice as above where every vertex has degree 4, and let F be the Boolean network over X induced by* nor*-functions. Then the activity of F for every vertex i is $\bar{\alpha}_{F,i} = 1040/2^{12} = 0.254$.*

Proposition 4.21. *Let X be a rectangular grid (torus) in which every vertex has degree 4, and F the map over X with the threshold vertex functions. Then the activity of F with respect to vertex i is given by*

$$\alpha_{F,i} = \left(\frac{1}{2^{12}}\right)\left[\gamma_1 - \gamma_2 + \gamma_3 - \gamma_4 + \gamma_5\right] \ , \tag{4.20}$$

where

$$\gamma_1 = 2^{10} \cdot \binom{4}{k-1} \ , \tag{4.21}$$

$$\gamma_2 = 2^7\left[\binom{3}{k-1}^2 + \binom{3}{k-2}^2\right] + 2^5\binom{4}{k-1}^2 \ , \tag{4.22}$$

$$\gamma_3 = 2^4\left[\binom{3}{k-1}^2\binom{2}{k-1} + 2\binom{3}{k-2}\binom{2}{k-2}\binom{3}{k-1} + \binom{3}{k-2}^2\binom{2}{k-3}\right] \ , \tag{4.23}$$

$$\begin{aligned}\gamma_4 = \Bigg[\binom{2}{k-1}^4 + 4\binom{2}{k-1}^2\binom{2}{k-2}^2 + 4\binom{2}{k-1}\binom{2}{k-2}^2\binom{2}{k-3} + \\ 2\binom{2}{k-2}^4 + 4\binom{2}{k-2}^2\binom{2}{k-3}^2 + \binom{2}{k-3}^4\Bigg] \ ,\end{aligned} \tag{4.24}$$

$$\gamma_5 = \sum_{x\in M} \mathbb{I}\left[x_{j_\ell} + x_{\beta_\ell} + x_{\beta_{\ell+1}} + x_{c\ell} \neq k-1\right] \ , \tag{4.25}$$

where M is the subset of states associated to $X(i;2)$ for which the Hamming norm of its elements projected to the components j_ℓ with $1 \leq \ell \leq 4$ is $k-1$, that is, $|(x_{j_1}, x_{j_2}, x_{j_3}, x_{j_4})|_H = k-1$, and where x_{j_ℓ}, x_{β_ℓ} and $x_{c\ell}$ are the states of the vertices j_ℓ, β_ℓ and c_ℓ as given in Figure 4.6.

The detailed proof of Proposition 4.20 and 4.21 can be found in [4, 8].

We have shown the graph of the activity as a function of the threshold value k in Figure 4.7. Note that for thresholds in the range $1 < k < 5$, the activity for the threshold function is

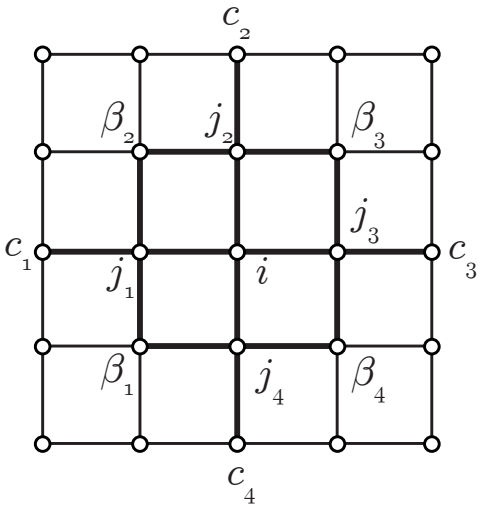


Figure 4.6: *A subgraph of the square lattice including $X(i;2)$ at center.*

Graph k	1	2	3	4	5
4-lattice	0.2539	0.7095	0.9009	0.7095	0.2539
4-regular tree	0.2641	0.7370	0.9046	0.7370	0.2641

Table 4.1: *Activity value (rounded to four decimal places) as a function of the threshold k for the 4-regular square lattice and the 4-regular tree. Thresholds 0 and 6 omitted.*

considerably larger than that for the nor-function given immediately above (Figure 4.7 and Table 4.1.)

Activity over triangular lattices

A triangular lattice is a regular graph with type-3 edges and a girth of 3 as indicated in Figure 4.8(a). In this case, the structure of the intersections among sets A_j (with $j \neq i$) is more involved than for the square lattice case, and type-3 edges are introduced. In principle, the derivations are similar to those of the square lattice case. Rather than giving a similar argument to the proof of Proposition 4.21, we limit ourselves to showing the graph of the activity as a function of the threshold value k, see Figure 4.8(b), but see also Table 4.2 for the approximate values.

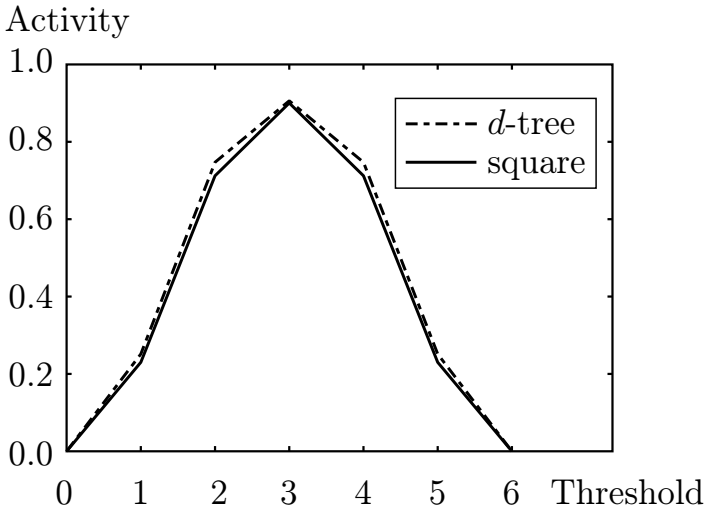


Figure 4.7: *Activity as a function of the threshold value k for the square lattice of Figure 4.6 (solid line). For comparison, the activity for the 4-regular tree is included (dashed line)*

Graph k	1	2	3	4	5	6	7
Triangular lattice	0.0881	0.3963	0.7603	0.9019	0.7603	0.3963	0.0881

Table 4.2: *Activity value (rounded to four decimal places) as a function of the threshold k for the triangular lattice. Thresholds 0 and 8 omitted.*

Activity over the Erdős-Rényi random graphs

Here, we provide an upper bound for the expected activity in $G_{n,p}$ for the uniform k-threshold function using the union bound. While bound obtained this way may not turn out to strong, the work in this section takes the first steps towards analyzing activity over ER random graphs.

First, let $B[n,p]$ denote the binomial random variable and let

$$f(t;n,p) = \Pr(B[n,p] = t) = \binom{n}{t} p^t (1-p)^{n-t} \ .$$

Since we are considering a family of graphs, we find it necessary to redefine some of the terminology developed earlier. Let i be the vertex under consideration. Let $\bar{\alpha}_{F,i}(G)$ denote the activity for $G \in G_{n,p}$. The expected activity $\mathbb{E}[\bar{\alpha}_{F,i}] \coloneqq \mathbb{E}_{G(n,p)}[\bar{\alpha}_{F,i}]$.

Note that, by symmetry $\bar{\alpha}_{F,i}$ is the same for all $i \in \{1,\ldots,n\}$. Next, recall the definition

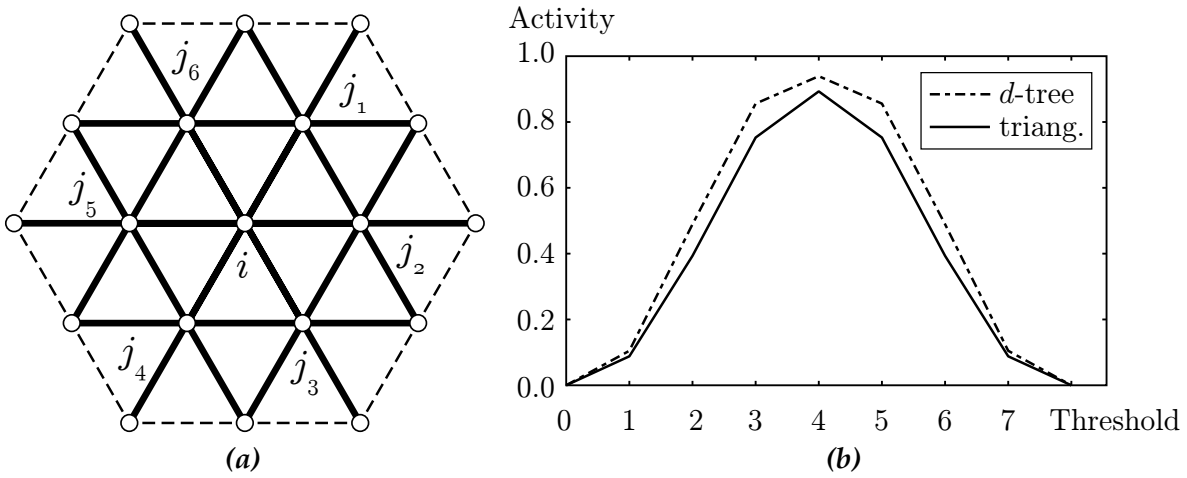


Figure 4.8: *(a) the triangular lattice. (b) Activity as a function of threshold for the triangular lattice (solid line). For comparison, the activity for the 6-regular tree is included (dashed line).*

of $\bar{A}_j(i)$ (or $\bar{A}_j$ in short). Here, we modify it as $\bar{A}_j(G)$ to denote the set of states x which satisfy $x_i = 0$ and $F(x + e_i)_j \neq F(x)_j$ in G.

Proposition 4.22. *If F is the Boolean network induced by the k-threshold function over $G(n, p)$, then $\mathbb{E}[\bar{\alpha}_{F,i}] \leq f(k-1; n-1, p/2) + \frac{(n-1)p}{2}\big(f(k-1; n-1, p/2) + f(k-2; n-2, p/2)\big)$.*

See [4, 8] for the proof of the above Proposition.

4.4.4 Discussion

In this section, we present our work published in [4, 8], where we have introduced an extension of the notion of activity proposed by Shmulevich and Kauffman [145]. This extension takes into account the impact of the network structure when studying $\mathbb{E}\Big[\mathbb{I}[F(x) \neq F(x + e_i)]\Big]$, which estimates how likely the perturbation e_i will cause successor states to diverge.

Naturally, orbits that initially separate may later converge, reflecting that x and $x + e_i$ may belong to the same attractor basin. Nonetheless, this notion of activity provides a measure of sensitivity with respect to initial conditions that account for network structure. Investigating *long-term activity*, that is, the probability that perturbing a state in the i^{th} coordinate will cause x and $x + e_i$ to have different limit sets, is an interesting direction for future research. Involving the attractors of F, we naturally expect this to be challenging work, even in most special cases.

Possibly interesting avenues for further work includes studying asynchronous systems. If the vertex functions are applied sequentially according to for example a permutation update sequence, are there effective ways of relating activity and the permutation? If so, are there principles connecting network structure and update sequence that allows one to minimize or maximize the activity of one or more vertices? Our results also show that the activity for the d-regular trees using threshold functions is larger than those of the corresponding square and triangular grids. Clearly, the combinatorial arguments become more involved with the presence of type-3 and type-4 edges. In general, what can be said about the impact of these classes of edges on activity?

Finally, we considered arbitrary initial states $x \in K^n$. What can be said about activity if we restrict x to be a periodic point? Some initial results on attractor activity are given in [100].

4.5 Application of GENEUS in Graph Dynamical Systems

4.5.1 Exploring GDS System Configurations in GENEUS

As discussed earlier, the system dynamics of a GDS is collectively determined by its components including vertex states, vertex functions, update scheme and graph structure. In many applications, we need to identify a system configuration such that particular dynamical properties are achieved. However, this is generally a hard task due to the fact that, as a discrete finite dynamical system, small changes to the system configuration of a GDS may lead to very different system dynamics. As a result, finding a GDS system with specific dynamical properties involves exploring a large configuration space exhaustively. With the design of standardized model configuration format and experimental design module, GENEUS can facilitate such studies in a systematic manner.

Figure 4.9 gives an example standardized XML configuration file for the GDS computational tool. With this design, graph structure, vertex functions and update scheme are parameterized such that systematical experimental design and fast system configuring are enabled. The dedicated model wrapper will automatically handle the invocation of the GDS calculation tool and evaluate the phase space with respect to specific requirement. In the next section, we will present an application of GENEUS to graph dynamical system.

```
<configuration>
 <parameters>
  <parameter key="ConfigVersion" type="integer"
    exposed="false" optional="false">
   <value>2014</value>
  </parameter>
  <parameter key="graph" type="string"
    exposed="false" optional="false">
   <value>Circ_n_cords/circ_5_2</value>
  </parameter>
  <parameter key="functionSpec" type="string"
    exposed="false" optional="false">
   <value>nonuniform</value>
   <parameter key="functionSpec/f0" type="string"
     exposed="false" optional="false">
    <value>threshold</value>
    <parameter key="functionSpec/f0/thresholdValue"
      type="integer" exposed="true" optional="false">
     <value>1</value>
    </parameter>
   </parameter>
   ...
  </parameter>
  <parameter key="schedule" type="string"
    exposed="false" optional="false">
   <value>sequential</value>
    <parameter key='schedule/permutation' type="string"
      exposed="true" optional="false">
     <value>1-2-3-4-0</value>
     ...
    </parameter>
  </parameter>
 </parameters>
</configuration>
```

Figure 4.9: *An example standardized configuration file for the GDS tool .*

4.5.2 Preservation of Phase Space Structure

In Section 4.3, we have shown that update scheme has a significant effect on the GDS system dynamics. Macauley and Mortveit have proved that sequential graph dynamical system maps F_w and $F_{\sigma_s(w)}$ have the same cycle structure, where $F_{\sigma_s(w)}$ is a cyclic left s-shift of the permutation w [107]. A natural question is: how much of the entire phase space structure including both transient paths and cycles are preserved under the cyclic left shift operation over the vertex states update sequence?

Figure 4.10 gives an example illustrating the concept above: consider the Circle$_4$ graph with nor functions and update sequence $w = (1, 3, 2, 4)$. The red edges represent the preserved potion of the phase space for the left 1-step shifted update sequence $\sigma_1(w) = (3, 2, 4, 1)$.

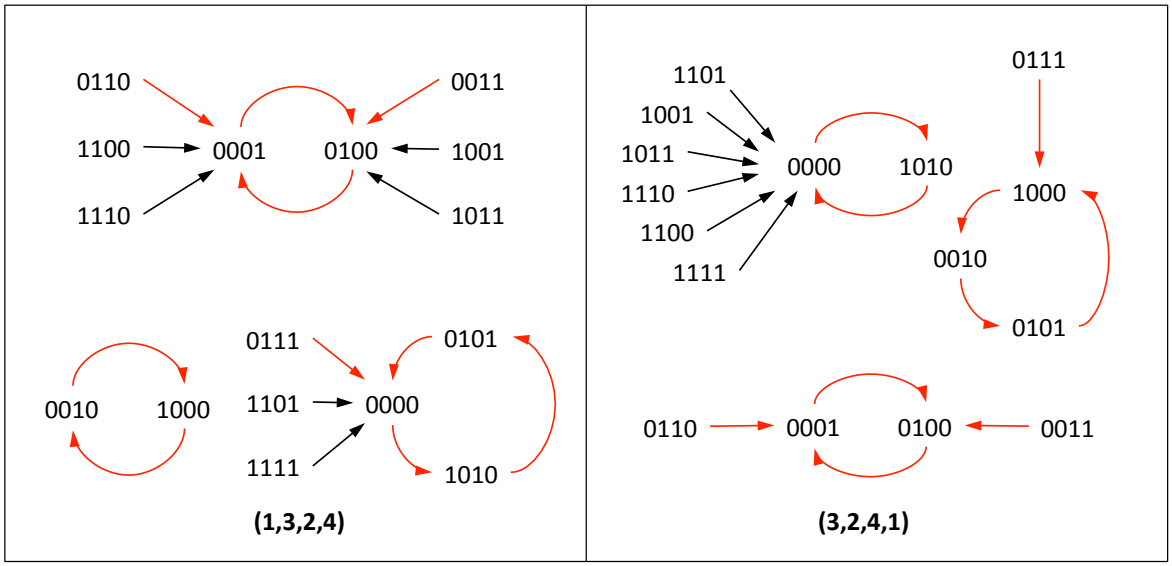


Figure 4.10: *The phase spaces of sequential* nor*-GDS with update sequences* $w = (1, 3, 2, 4)$ *and* $\sigma_1(w) = (3, 2, 4, 1)$.

If we consider the cyclic 1−shift we have following conjugating relationship between F_w and $F_{\sigma_1(w)}$:

$$F_{\sigma_1(w)} \circ F_{w_1} = F_{w_1} \circ F_w. \tag{4.26}$$

As a simple proof, if we expand the left hand side of above equation, by definition we have,

$$F_{\sigma_1(w)} \circ F_{w_1} = F_{w_1} \circ F_{w_n} \circ F_{w_{n-1}} \circ \cdots \circ F_{w_2} \circ F_{w_1} = F_{w_1} \circ F_w.$$

Therefore, F_{w_1} has a significant effect of the map from the phase space $\Gamma(F_w)$ to $\Gamma(F_{\sigma_1(w)})$. The complete structural implications are still not well understood and this is a work in progress. However, we have used GENEUS to search a large number of GDS configurations and found a system where a large portion of the phase space structure is preserved under the left 1-shift operation on its update sequence: If we consider a GDS map F_w shown in Figure 4.11 with update sequence $w = (2, 0, 1, 3, 4)$, then $F_{\sigma_1(w)}$ preserves 93.75% of the phase space structure of F_w.

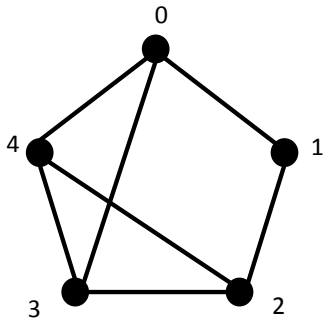


Figure 4.11: Circle$_5$ *graph with 2 diagonal edges. Vertices are associated with Boolean bi-threshold functions: vertex 2 has up- and down-threshold values 4 and 2, respectively; other vertices have up- and down-threshold values 1 and 2, respectively.*

4.6 Summary

The study of stability and the response to perturbations of graph dynamical systems is central to understanding of their dynamical properties. Perturbations may take many forms, with examples including perturbations of the dependency graph, the vertex states, the vertex functions, the update scheme or combinations of these.

In this chapter, we summarized our theoretical results on the stability of graph dynamical systems with respect to various types of perturbations. In Section 4.2, we have extended bi-threshold systems to include dynamically evolving thresholds. This provides a more realistic venue for modeling complex phenomena, such as adaptive biological, social and technical systems. We have shown the result that synchronous, transition-based, dynamic bi- threshold GDS maps may only have periodic orbits of size 1 or 2. In Section 4.3, we studied the limit cycle structure of standard threshold dynamical systems with block-sequential update. In a block-sequential update, the vertex set of the graph is partitioned into blocks, and the blocks are updated sequentially while the vertices within blocks

are updated in parallel. We identified a sufficient condition for the system to have only fixed points as limit sets. We also demonstrate several well-known graph families that satisfy this condition. In Section 4.4, we have introduced an extension of the notion of activity proposed by Shmulevich and Kauffman [145]. We take into account the actual graph structure of the graph dynamical system. The notion of activity measures the probability that a perturbation in an initial state produces a different successor state than that of the original unperturbed state. It captures the notion of sensitive dependence on initial conditions and provides a way to rank vertices in terms of how they may impact predictions. We give basic results that aid in the computation of activity and apply this to graph dynamical systems with threshold functions and nor functions for elementary cellular automata, d-regular trees, square lattices, triangular lattices, and the Erdős-Rényi random graph model.

Graph dynamical system is an important mathematical foundation of network-based simulation models. Theoretical research on the stability and robustness of GDS with respect to external noise or perturbations forms an essential component of model validation of network-based simulation models. The results presented in this chapter therefore serves as the major theoretical contribution of this dissertation.

Chapter 5

Applications in Network-based Simulation Models

5.1 Introduction

By nature, network-based simulation models often require large data sets as well as a large variety of data sets placing heavy demands on both data management and computation. As mentioned in the motivational example in Section 1.2 of Chapter 1, the network of transportation system in Washington D.C. has around 50,000 nodes and 230,000 links. The simulations were run on a large 60 node multi-core cluster. One complete run (around 120 iterations) takes about 35 hours and requires a few TB of space.

When performing model validation, the challenges will be intensified since the model validation process is usually accompanied by experimental designs involving a large number of cells, each with multiple replicates. Management of a large number of experiment instances along with the associated input/output/configurations is nontrivial. Moreover, it is often desirable that model validation can be done by a third-party in addition to or rather than by the modelers themselves to guarantee the credibility, a process sometimes referred to as independent verification and validation (IV&V) [45]. For this, it is highly desirable to make sure the experiment results are reproducible. However, the involved process of conducting such analysis complicates matters. There is a lack of standardized model configurations as well as ways to actually invoke the simulation models, which makes it challenging to even execute the workflows, not even considering the possible statistical designs involved. The challenge is compounded by the need to integrate domain experts (who typically lack modeling and computing skills), and the needs for advanced expertise covering programming, high-performance computing, statistical analysis, and

data management.

These challenges actually motivate and guide the design of GENEUS system. In this chapter, we will illustrate the capabilities of GENEUS by several application case studies. These case studies are actual simulation experiments with large data sets involved. We will show how GENEUS facilitates such large-scale simulation experiment and improves the efficiency in both management and human productivity.

5.2 Model Validation of Networked Epidemics

5.2.1 Scenario

Infectious diseases constitute a major public health challenge all over the world. The 2014 Ebola outbreak in Western Africa and the recent Zika virus have caused hundreds of thousands of deaths and brought significant threat to human lives and society [101]. New infectious diseases are emerging and there is still no cure for many of them. According to a World Health Organization (WHO) report [76], every year transmissible influenza result in an estimated three to five million cases of severe illness and 250,000 to 500,000 deaths. Planning a response to an outbreak of a pandemic is a high public health priority. Due to ethical and practical reasons, controlled experiments are often impossible in epidemiology. Fortunately, computational models allow us to identify the spatial-temporal dynamics of epidemics and can therefore help policy makers forecast the epidemic propagation pattern, plan medical facility placements, and prepare response strategies.

Social contact network-based simulations have been widely used for modeling pandemic outbreaks [30, 101]. Often, this type of simulation model incorporates highly resolved synthetic representations of the person-to-person contact patterns of people in a specific geographical region, which is central to modeling disease transmission. Computational epidemiologists have conducted simulations to test the effectiveness of a set of potentially feasible intervention strategies [3, 71], and made considerable contributions to combating infectious diseases.

5.2.2 Simulation Method

Modeling disease transmission over contact networks

Many simulation tools for modeling infectious disease have been developed [24, 31, 32]. In this case study, we use EpiFast [31] for our illustration experiments. EpiFast is a stochastic simulation model for disease propagation over large contact networks. It has been used in many studies with sophisticated settings to evaluate various dynamic interventions like vaccination and to provide decision support for public health policy makers.

In EpiFast, the person-to-person contacts are formally represented by a network $G(V, E, W)$, where V is the node set, E is the edge set, and W is the edge weight set. In this network, nodes correspond to individuals and edges represent contacts between two end nodes. An edge $e = (u, v)$ with weight $w(e)$ denotes that node u has contact with node v for a duration $w(u, v)$ throughout the day, and during which the disease may transmit from u to v with probability $p(w(e))$. The function p typically depends on specific disease infectivity.

Many stochastic simulations adopt the standard SEIR disease model which is widely used in epidemiology [31]. Each person is in one of the four health states at any time: susceptible (S), exposed (E), infectious (I) and removed (R). A person is in the susceptible state until she/he becomes exposed. The person will remain exposed for an incubation period, during which she/he is not infectious. Then the person becomes infectious and remains so for an infectious period. Finally, she/he becomes removed (or recovered) and remains so permanently. With the SEIR model, the disease spreads in a population in the following way: it can only be transmitted from an infectious node to a susceptible node. On any day, if node u is infectious and v is susceptible, disease transmission from u to v occurs with probability

$$p(w(u, v)) = 1 - (1 - r)^{w(u,v)} , \tag{5.1}$$

where r is the disease transmissibility, the probability of disease transmission for a contact of one unit time. Infections from multiple infectious individuals to a susceptible individual are treated as independent events. Thus, the disease propagates probabilistically along the edges of the contact network.

Targeted layered containment strategies of influenza pandemic

In [71], Halloran et al. developed targeted layered containment strategies of an influenza pandemic happening in the Chicago area by using stochastic, spatially structured, network-based simulation models. To control the pandemic spread, a set of interventions were considered. They combined antiviral treatment and household isolation of identified

cased, prophylaxis and quarantine of their household contacts, closure of schools, social distancing in the workplace. Because these interventions include both targeted and general strategies, they are called targeted-layered containment (TLC) approaches. In [71], the researchers examined different levels of ascertainment of symptomatic influenza cases, and compliance with the interventions and cumulative illness *attack rate* (total number of infected people – past and present) threshold for initiating interventions.

In this case study, we revisit the TLC simulation experiment and use our framework to streamline the study pipeline.

5.2.3 Experiment Description

One uncertain factor of the model is the transmissibility of a disease. This uncertainty also poses challenges for estimating how transmissible a future pandemic will be. Typically, the transmissibility is measured by using the reproductive number, that is, the number of secondary cases for each primary case. In our study, it is characterized by the probability of transmission per unit of contact time between two persons (r in Equation 5.1), which is a real number. As a matter of fact, transmissibility is a dominant factor that affects the cumulative attack rate significantly, and this will be shown in the experiment results.

Interventions for an ascertained case include antiviral treatment for the infected individual, targeted antiviral prophylaxis of household contact, home isolation (ill person is isolated in the home but not isolated from housemates), and quarantine of household contacts (household contacts are all quarantined within the home). General population-wide interventions are summarized as follows:

School closure. All schools, including primary, middle and high schools are closed at a particular threshold cumulative illness attack rate. Students are expected to stay at home with certain compliance levels.

Work closure. At a particular threshold cumulative illness attack rate, workplaces are closed and contacts are reduced by a certain percentage and also by a certain compliance level.

Social distancing. Community social distancing results in fewer public activities. Examples include closing shopping centers, and reducing visits to restaurants and other public locations. After a particular threshold attack rate, people are encouraged to stay at home with a certain compliance.

These interventions are triggered when a threshold fraction of the population becomes infected. In [71], several scenarios were considered based on different levels of the threshold

values which varied from 1% to 0.01%. Different combinations of intervention compliance levels were also examined, but in a coarse manner (such as 30%, 60%, 90%). An advantage of our experimental design module and automated model configuration method is that one is able to run hundreds and even thousands of simulations to explore the parameter space more thoroughly, something that was effectively impossible in the original study.

5.2.4 Experiment Setup

Model configuration

Figure 5.1 shows an example EpiFast main configuration file, which contains references to a collection of sub-configuration files. Typically, experiment parameters can be in either the main configuration file or any sub-configuration file. Sub-configuration files are organized in a rather arbitrary manner, they can be located in different working directories or even on different machines. In practice, it is generally hard to perform experimental designs with such unstructured configuration format. In this particular study, model parameters such as school closure compliance rate, work closure compliance rate and social distancing compliance rate are located in a sub-configuration file called intervention file. Manually configuring the model is tedious and can easily introduce errors.

```
ConfigVersion = 2009
ContactGraphFile = LBR_430-socnet.efi
Transmissibility = 3.07537688442e-05
InfectiousPeriodFormat = DISTRIBUTION
InfectiousPeriodFile = Infectious.Period.Distribution
IncubationPeriodFormat = DISTRIBUTION
IncubationPeriodFile = Incubation.Period.Distribution
EpidemicSeedType = RANDOM_SEEDS_FIRST_DAY
EpidemicSeedNumber = 5
EpidemicSeedFile = subpop_all.txt
SimulationRandomSeed = 7654321
InterventionFile = Intervention
IterationNumber = 10
OutputFile = out/EFO
SimulationDuration = 200
```

Figure 5.1: *An example of the main EpiFast configuration file. Here red boxes denote references to sub-configuration files.*

To facilitate generalized experimental design and uniform model invocation, we devised a standardized configuration format for EpiFast using the tree architecture discussed earlier, where the main configuration file is the trunk and sub-configuration files are treated as branches, see Figure 5.2 for an example. With the associated parameter navigation API, fast parameter locating is possible. In addition, we support saving and loading of parameter tree branches to and from external files which enable customized configuration assembly. The entire configuration specification and all tree branches are also indexed by the central registry system. This has a lot of practical applications, for example, a particular intervention file usually corresponds to a specific intervention strategy and is devised by domain experts. With the standardized configuration format along with the central registry system, complex intervention strategies can be easily archived and reused when necessary without duplication of efforts.

Experimental design

By using the standardized model configuration format, experimental parameters are automatically exposed, along with the user specified parameter space, and it is sufficient for the experimental design module to take samples in a uniform manner. We utilized the Latin hypercube sampling (LHS) plug-in in the algorithm library for parameter sensitivity analysis. The reason we use LHS instead of traditional DOE methods such as factorial design is that exhaustively exploring parameter space for studies of this scale is not practically feasible. For instance, running one TLC study can take fifteen to twenty minutes over the Miami and the Seattle networks; thirty to forty minutes over the Chicago network on a high-performance computing cluster. Clearly, it is not possible to run all combinations of the parameter settings, and therefore, a sparse space-filling experimental design with economic run size is essential for conducting the sensitivity analysis.

After the generation of the experiment configurations, the EpiFast model wrapper prepares the model execution environment and automatically submits job execution requests to the computational back-end. Meanwhile, the registry records the information of the experiment instance such as working directory, input and output data sets, computing platform, execution log and so on. This is useful when the finished experiment instances need to be repeated.

Sensitivity analysis

When all the model execution jobs are finished, the model wrapper will process the simulation outcome and extract quality of interests from the raw output data. The

```
<epifast_cfg>
  <parameter key="ConfigureVersion" type="int"
   exposed="false" optional="false">
    <value>2009</value>
  </parameter>
  <parameter key="ContactGraphFile" type="string"
   exposed="false" optional="false">
    <value>LBR_430-socnet.efi</value>
  </parameter>
  ...
  <parameter key="InfectiousPeriodFile" type="string"
   exposed="false" optional="false">
    <value>Infectious.Period.Distribution</value>
    <registry_id>data_set_112</registry_id>
    ...
  </parameter>
  ...
  <parameter key="InterventionFile" type="string"
   exposed="false" optional="false">
    <value>Intervention</value>
    <registry_id>data_set_114</registry_id>
    <parameter key="school_closure_compliance" type="float"
      exposed="true" optional="false">
      <value>0.5</value>
    </parameter>
    ...
  </parameter>
  ...
</epifast_cfg>
```

Figure 5.2: *Standardized XML configuration file for EpiFast shown in Figure 5.1. Some configuration entries are omitted for succinctness.*

processed data are used to perform parameter sensitivity analysis. Sensitivity analysis is concerned with how changes in the model inputs influence the outputs. Generally speaking, there are two types of sensitivity analyses: local sensitivity analysis examining individual parameter effects (also known as main effects), and global sensitivity analyses which also consider interactive effects among two or more parameters. Both of them can enhance the understanding of a complex model, find aberrant model behavior, identify which input have a significant effect on a particular output and so on. In this study, we first investigated individual parameter effects one at a time and then used two global sensitivity metrics for assessing the parameter effects: Sobol's method [152] which partitions the output variance in terms of input parameters and their increasing order of interactions as

$$V(X) = \sum_i V_i(x_i) + \sum_{i<j} V_{i,j}(x_i, x_j) + \cdots + V_{1\ldots s}(x_1, \ldots, x_s) \ , \tag{5.2}$$

and the delta sensitivity indicator [35] which examines the influence of input uncertainty on the entire output distribution without reference to a specific moment of the output. We will give the sensitivity analysis results with respect to both measurements in the next section.

5.2.5 Results

Experiment #1: transmissibility and intervention threshold

In this experiment, we used synthetic contact networks of the three US cities Miami, Seattle, and Chicago. Miami and Seattle have similar population sizes but distinct demographics, see Table 5.1, whereas Chicago area has much larger population size (9,038,163 vertices) and mixed demographic properties as compared to the other two cities. Thus, we use the simulation over the Chicago network as a control experiment.

Intuitively, cumulative illness attack rate will increase with greater disease transmissibility, and reductions in the attack rate are expected when TLC interventions are installed. The initiating threshold fraction of illness is also an important factor to consider since it determines how stringent the intervention strategy will have to be. We considered three scenarios in this experiment: the baseline scenario without intervention, interventions initiated after 1% and 0.1% of the population has developed symptomatic influenza for the other scenarios, in which, compliance rate for school closure, work closure, and social distancing interventions were fixed at 30%, 50% and 50%, respectively.

The results of the experiment are shown in Figure 5.3. It can be observed that TLC interventions can effectively contain the influenza spread. Scenario 3 with a more stringent

	Miami region	Seattle region
Population size	2,092,076	3,206,897
Age group		
Preschool (0-4 years)	6.74%	6.78%
School-aged (5-18 years)	15.03%	20.33%
Adults (19-64 years)	65.04%	63.08%
Seniors ($\geq$ 65 years)	13.18%	9.80%
Household size		
Small (1 person)	7.96%	10.90%
Medium (2-3 persons)	38.21%	45.76%
Large ($\geq$ 3 persons)	53.83%	43.34%

Table 5.1: *Age and household size composition of the Miami and Seattle populations [159].*

intervention initiating threshold achieves a lower illness attack rate compared with Scenario 2. When comparing the results across cities, Seattle has a higher attack rate than Miami if no intervention is applied. However, when TLC interventions were implemented, Seattle achieves a lower attack rate. This inversion results from the fact that Seattle has a larger portion of school-aged people and the fact that school closure intervention is the most significant factor for controlling the disease spread, see Table 5.1. This is demonstrated in the next set of experiments. Chicago always has a higher attack rate because this area has a larger population size and the cascade effect can result in a pandemic expanding.

Experiment #2: sensitivity analysis of intervention compliances

In the TLC study, we mainly considered three types of interventions: school closure, work closure, and general social distancing. The compliance level of these interventions is a major factor that influences the overall attack rate. We conducted both local and global sensitivity analyses to assess their effects.

In this study, we fixed the disease transmissibility at 5.8×10^{-5}, intervention initiating threshold value at 1%, and we used the Miami network. First, we varied the intervention compliance levels one at a time for 30 different values to evaluate their local sensitivities. Each experiment had 5 replicates. Figure 5.4 gives the OFAT sensitivity analysis result. It is shown that all these three interventions have apparent effects on the overall illness attack rate, but that school closure intervention is more effective than the other two in decreasing the illness attack rate.

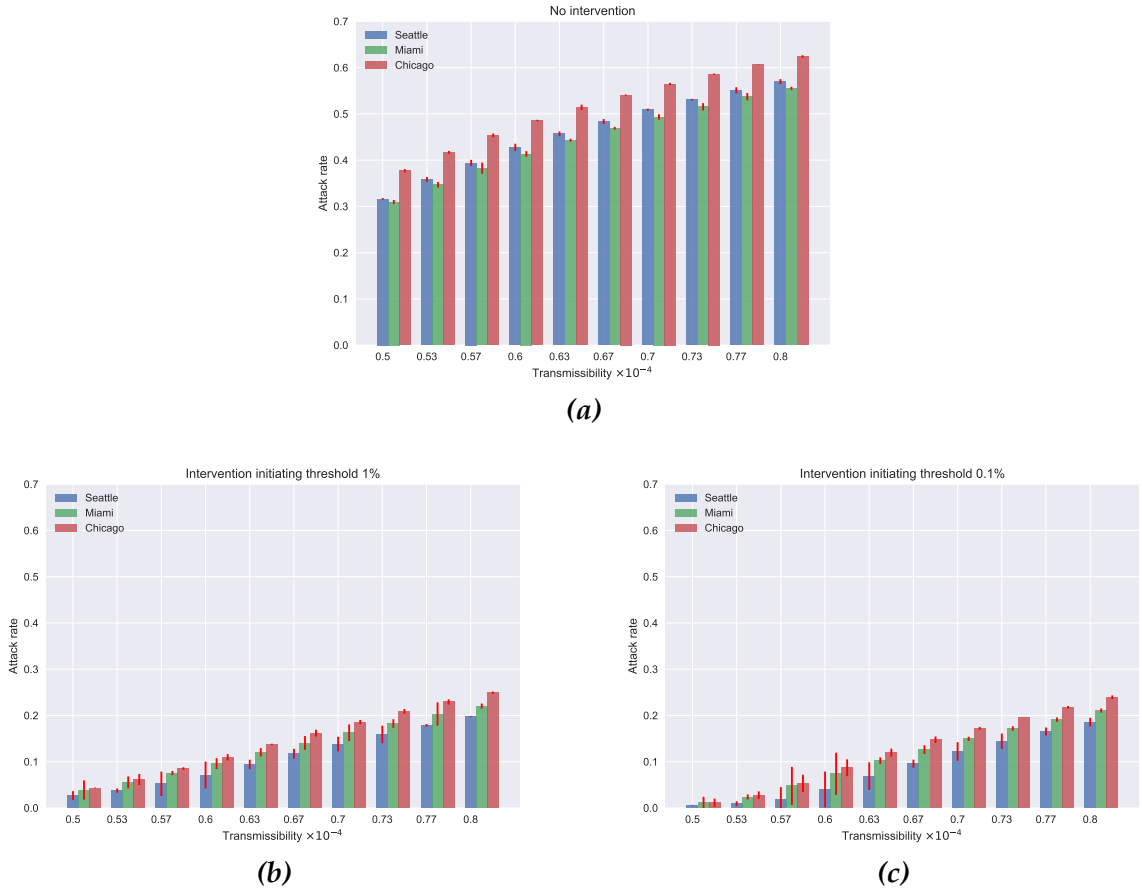


Figure 5.3: *Illness attack rates for three US cities Seattle, Miami and Chicago (a) without intervention, (b) and (c) with TLC interventions for* 1% *and* 0.1% *initiating threshold values, respectively. Each experiment ran with 5 replicates. Error bars denote standard deviation.*

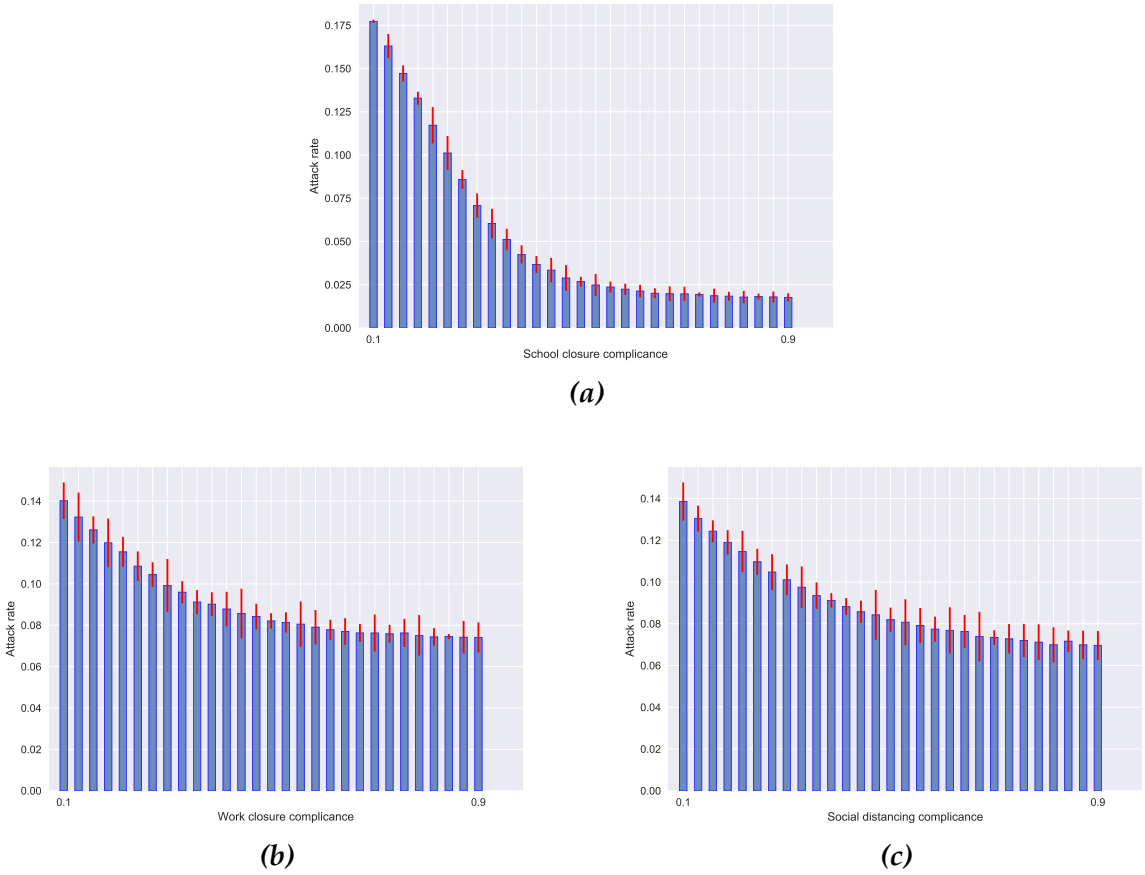


Figure 5.4: *Local sensitivity analysis of different intervention compliance parameters: (a) school closure compliance rate, (b) work closure compliance rate, and (c) social distancing compliance rate. Each parameter is examined for 30 different values with 5 replicates. Error bars denote standard deviation.*

For global sensitivity, we varied the three intervention compliance levels simultaneously for 1000 runs by using the Latin hypercube sampling method. Figure 5.5 is the scatter plot for each dimension of the parameter space. One can observe that school closure intervention shows a stronger pattern whereas the other two are more random, which again indicates that school closure intervention dominates the intervention efficacy.

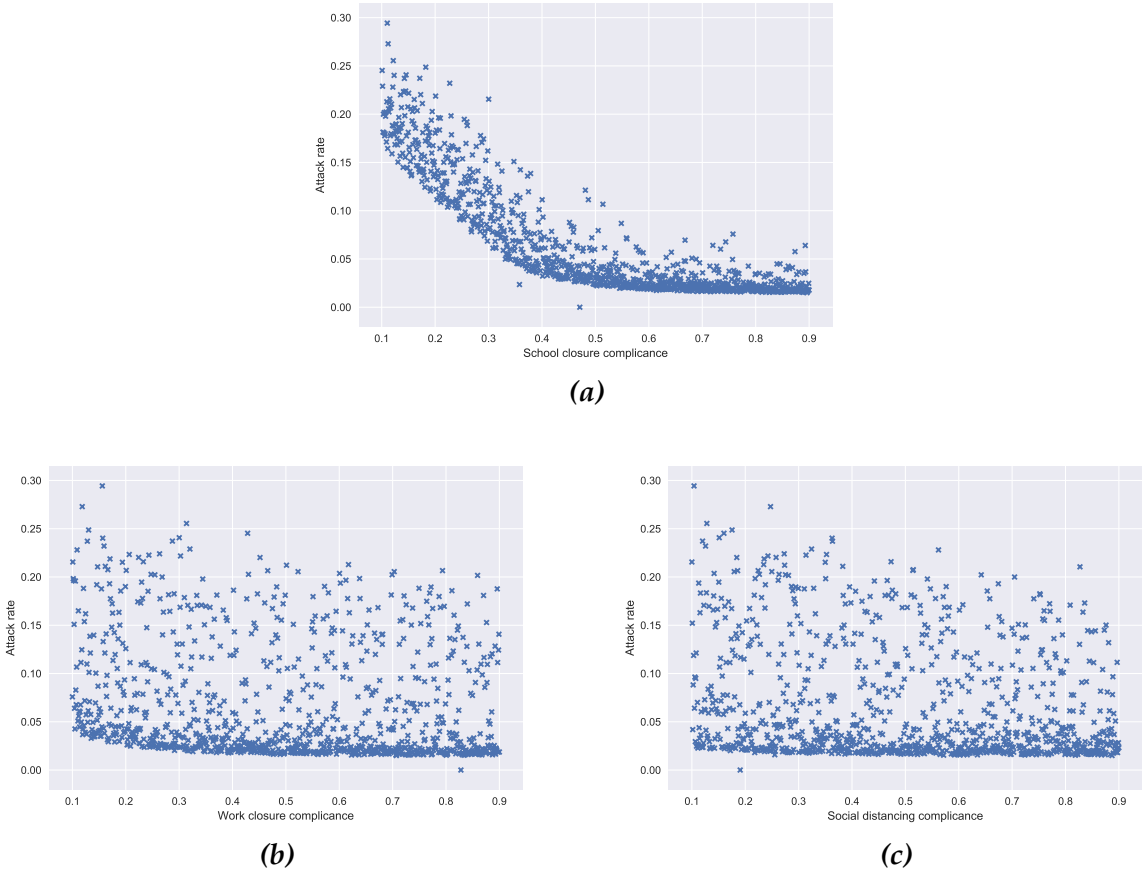


Figure 5.5: *Scatter plot of the global sensitivity analysis for various intervention strategies: (a) school closure, (b) work closure and, (c) social distancing. Here 1000 sample points were evaluated using the Latin hypercube Sampling method.*

Quantitatively, we calculated the sensitivity indices for the three intervention strategies, see Table 5.2. Both the Delta measurement and Sobol's measurement show that school closure is the most effective intervention option. Our experiment results suggest that the combination of various intervention strategies can reduce the illness attack rate significantly.

parameter	Delta measurement	Sobol's measurement
School closure compliance	0.534339	0.836894
Work closure compliance	0.082681	0.055476
Social distancing compliance	0.082046	0.047361

Table 5.2: *Parameter ranking with two sensitivity importance measures.*

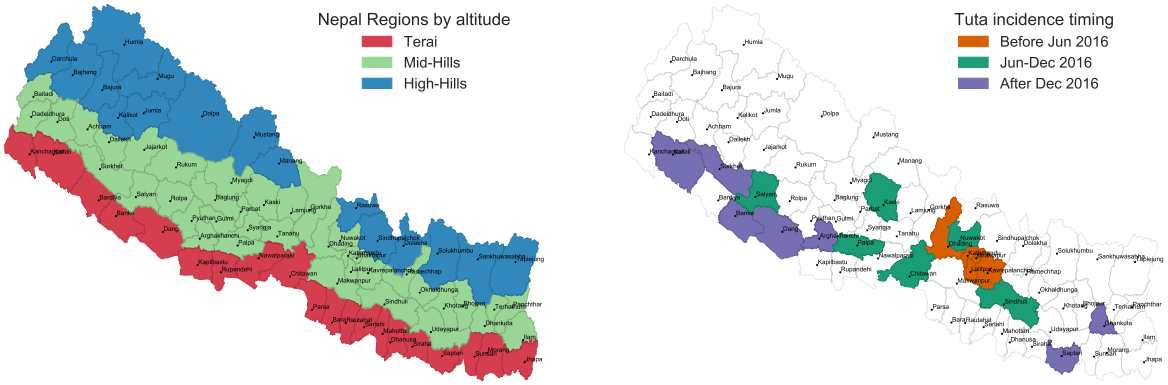
5.3 Sensitivity Analysis of Pest Diffusion Simulation Model

5.3.1 Scenario

Invasive alien species spread is a complex phenomenon driven by various natural and anthropogenic factors. While the knowledge of biology and climate is essential to assess establishment risk and devise sustainable management strategies [155], it is critical to understand human-mediated pathways to prevent introduction and mitigate immediate impact [20, 81].

As is the case with many built infrastructures, trade of goods naturally yields to network representations. Typically, nodes of the network represent locations–ranging from markets to continents depending on the context–connected by transportation infrastructure. The influence of one node on another is captured by an edge with weight that is a function of the transaction volume across that edge. A major challenge in constructing such networks is estimating the temporal flows. The intricate web of supply chain logistics makes it hard to document transactions, and even in economically developed countries, obtaining commodity specific flow data is a challenge [109]. On the other hand, it is comparatively easy to obtain datasets on production, population, and economic indicators at finer spatial resolution, thus allowing the use of spatial interaction models such as the gravity model [72]. Further such an approach is also better suited for data-poor regions.

A representative flow network can yield valuable insights into the phenomenon. The network structure helps identify possible entry points and hubs [121, 157]. Network dynamical processes such as the SEIR (Susceptible → Exposed → Infected → Recovered) model from epidemiology [128] is applied to capture the spatio-temporal evolution of the invasion. Model selection and validation is challenging due to the lack of accurate ground surveillance, since very few countries have the capacity to effectively react to impending and emerging invasions [51]. Therefore, there is a need for a robust modeling framework that can operate with limited observational data.



(a) *(b)*

Figure 5.6: *Nepal geography and T. absoluta incidence. (a) Physiographic division of Nepal. (b) District-level report of* T. absoluta *incidence.*

In this study, we focus on the spread dynamics of *Tuta absoluta* (*Gelechiidae , Lepidoptera*) (Meyrick,1917), a devastating pest of the tomato crop. The region of interest is Nepal, a biodiversity hotspot [96] and largely an agrarian economy, which recently reported *T. absoluta* invasion in 2016 [18]. Indigenous to South America, *Tuta absoluta* or the South American tomato leaf miner was accidentally introduced to Spain in 2006 [49]. Since then, it has rapidly spread throughout Europe, Africa, Western Asia, Indian subcontinent and Central America [40] over the past decade. Since tomato is among the top two traded vegetables (http://www.fao.org), it is strongly suspected that trade played a critical role in *T. absoluta*'s rapid spread. Indeed, on multiple occasions, it has been discovered in packaging stations [122]. Karadjova et al. [88] observed that the spread pattern in Bulgaria was correlated with prime trade routes. With tomato being a commercially important crop [69], *T. absoluta* has had significant global impact. For example, in Turkey, the annual estimated intervention cost is €167M per year [125]. Due to extensive insecticide treatment in Europe [67], insecticidal resistance has been recently observed in populations of *T. absoluta* [70]. Lack of effective natural predators has made integrated pest management quite challenging.

In Nepal, after its introduction in 2016 in Kathmandu district [18], it has continued to spread to several districts (Figure 5.6(b)). Tomato is among the major cash crops that Nepalese farmers depend on for their livelihood. Therefore, stresses such as the *T. absoluta* invasion can have a huge impact on Nepal's agrarian economy and fragile ecosystem.

5.3.2 Modeling Commodity Flow in the Context of *T. absoluta* Spread

Flow network construction

We model the flow of agricultural produce among markets based on the following premise: The total outflow from a market depends on the amount of produce in its surrounding regions, and the total inflow is a function of the population it caters to and the corresponding per capita income. The main assumptions in this model are: (*i*) imports and exports are not significant enough to influence domestic trade, (*ii*) fresh tomatoes are mainly traded for consumption, and (*iii*) the higher the per capita income, the greater the consumption. As discussed in the Methods section, these are fair assumptions in the case of Nepal. The flows are estimated using a doubly constrained gravity model [15, 87].

Figure 5.6(a) gives the physiographic division of Nepal. Within a distance of 180km from south to north, the altitudes range from 67 meters above sea level (masl) corresponding to tropical Terai region bordering India to the mountain region exceeding 3000 masl [11]. Based on the altitude-induced agricultural cycle, we have divided the entire year into two seasons: S1 (June to November) and S2 (December to May). During season S1, mainly the Mid Hills and High Hills contribute to the production, while in S2, Terai region is the major producer. As a result, we have two flow networks, one for each season. The total outflow from each market i, O_i is the amount of produce that arrives at the market in the specified season. The total inflow I_i is proportional to the population catered to by the market and a function of its average per capita income η_i, η_i^γ, where γ is a tunable parameter. The flow F_{ij} from location i to j is given by $F_{ij} = a_i b_j O_i I_j f(d_{ij})$, where, d_{ij} is the distance to travel from i to j, and $f(\cdot)$ is the *distance deterrence function*: $d_{ij}^{-\beta} \exp(-d_{ij}/\kappa)$, where β and κ are tunable parameters. The coefficients a_i and b_j are computed by iteratively solving the system of equations

$$a_i = \frac{1}{\sum_j b_j I_j f(d_{ij})},$$
$$b_j = \frac{1}{\sum_i a_i O_i f(d_{ij})},$$

such that the total outflow and total inflow at each node agree with the input values [87].

Modeling the pest spread

We use a discrete-time SI (Susceptible-Infected) epidemic model on directed weighted networks [128] to model pest dispersal. Each node is either susceptible (free from pest) or infected (pest is present). Henceforth, we use the term "infected" for a node or a region frequently to imply *T. absoluta* infestation at that location. A node i in state I infects each of its out-neighbors j in the network with probability proportional to the flow F_{ij} at each time step t. The infection probabilities are obtained by normalizing flows globally: $\lambda_{i,j} = \frac{F_{i,j}}{\max_{i,j} F_{i,j}}$. The model is based on two assumptions: (*i*) an infected node remains infected and continues to infect its neighbors and (*ii*) the chance of infection is directly proportional to the volume traded. Considering the fact that Nepal was ill-prepared for this invasion and the lack of effective intervention methods, (*i*) is a fair assumption. Historically, *T. absoluta* has spread rapidly in regions where tomato trade has been the highest (parts of Europe and Middle-East for example) thus motivating assumption (*ii*).

$P_I(i, t, f_0)$ denotes the probability that node i is infected by time t given the initial condition f_0 which assigns probability of infection at time step $t = 0$ to each node. In general, computing P_I is hard. Efficient methods have been proposed to estimate this probability [12, 105]. Here, we adopt the *dynamic message passing algorithm* by Lokhov et al. [105], a generalization of the Belief Propagation algorithm [129]

The initial configuration f_0 is chosen to mimic a spatially dispersed seeding scenario. We first select a *central* seed node, and then use a Gaussian kernel with parameter σ around the seed node to assign initial infection probabilities for neighboring markets. A market at a distance d from the seed as measured on the road network, is assigned the infection probability $e^{-\frac{d^2}{2\sigma^2}}$. The kernel accounts for factors such as uncertainty in determining the pest location, the possibility of spread of the pest through natural means as well as interactions between these markets.

Model evaluation through origin inference framework

Whether to establish goodness of fit of the model given the observed spread, or to forecast, we are faced with the challenge of translating t of the SI model to a real-world equivalent time period. Estimating this requires spatio-temporal distribution of the pest at a high resolution. Despite considerable monitoring efforts in Nepal, the incidence reports have poor time information. Monitoring is resource intensive: the placement of traps is largely determined by accessibility and availability of trained personnel. Given these constraints, the pest might remain undetected for a long time. Secondly, in the absence of monitoring,

the pest's presence will become apparent only during the growing season. Therefore, there is a possible delay of several months before it is reported and confirmed. To circumvent this problem, we make use of SI model's monotonic property: For any $t' > t$, $P_I(i, t', f_0) \geq P_I(i, t, f_0)$. Rather than attempt to match the time step to observational data, and thus estimate the absolute infection probabilities, we analyze the spread using relative vulnerabilities of the nodes. For this analysis to be meaningful, it is important that the rank list changes slowly with t with other parameters fixed. We establish that this is indeed true through rigorous sensitivity analysis.

The incidence reports in Nepal and ground surveillance strongly suggest that *T. absoluta* was first introduced to the area around Kathmandu [18]. By December 2016, several areas had reported its presence (Figure 5.6(b)). With this information, we evaluate our model based on the following backward inference problem: given an observation of node states at time t, what is the most likely origin of invasion? This is precisely the source detection problem, which has been addressed extensively in recent years [12, 143]. We examine the likelihood of markets or regions being the source nodes, and in particular, we compare this with the likelihood of the region around Kathmandu being the source (see Figure 5.7). Suppose $\mathcal{O}$ is the observation criteria; it consists of pairs (v, s) where v is a node and s is a state. For each candidate initial condition f_0, ideally, we would like to compute the joint probability of $\mathcal{O}$ at a time step t, $P(\mathcal{O} \mid f_0, t)$. However, there are two issues. Firstly, it is not tractable to compute this probability. Following Lokhov et al. [105], we approximate this joint probability as a product of the marginal probability estimates from the message passing algorithm and define an *energy function* for each tuple (f_0, t) as

$$\phi(\mathcal{O} \mid f_0, t) = -\log\left(\prod_{(i,I)\in\mathcal{O}} P_I(i, t, f_0) \prod_{(i,S)\in\mathcal{O}} (1 - P_I(i, t, f_0))\right).$$

The lower the value of ϕ, the higher the likelihood of f_0 being the initial condition. Secondly, recalling the uncertainty in interpreting time step t, we examined the relative likelihoods of each f_0 and the stability of the ranking across a range of model parameters.

While the intended usage of the origin inference formulation is to determine the source of infection, we have adapted it to compare expected spread in the model with observed data. Our results demonstrate that this framework is in general very useful in finding the likely pathways of introduction of the pest.

Epidemic source inference results.

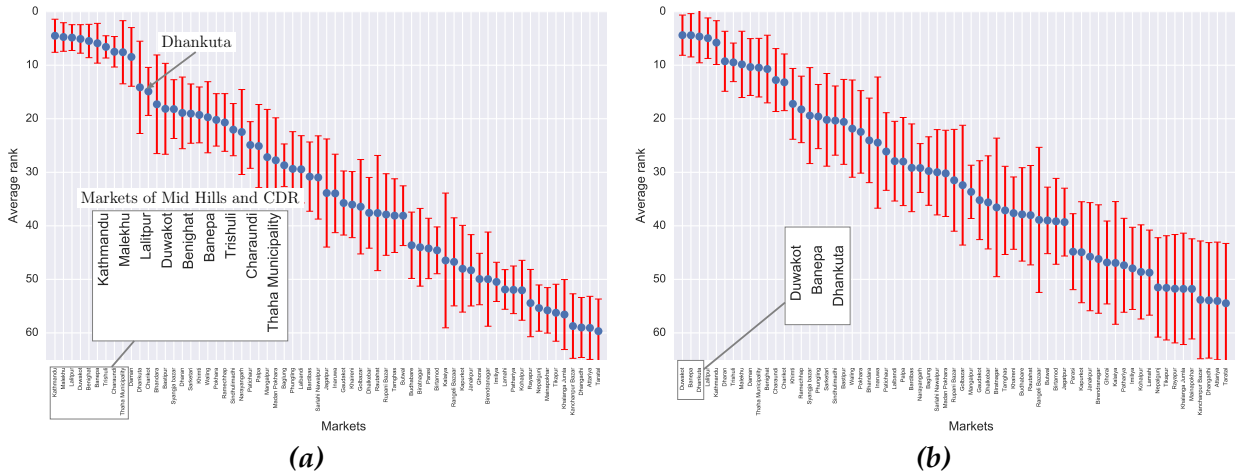
We consider the spread during June-November for model evaluation. Hence, the network corresponds to season S1 for a given set of parameters (β, κ, γ). Our objective was to rank

various starting configurations f_0 based on $\phi(\mathcal{O} \mid f_0, t)$ given $\mathcal{O}$, t and flow network of season S1. Recall that every initial configuration f_0 corresponds to a central node and Gaussian kernel parameter σ, which results in the infection probabilities at time $t = 0$. For a given σ, we evaluated the likelihood of each node being the central node. We considered two criteria based on which the likelihood of each f_0 as the starting configuration was computed: (*i*) $\mathcal{O}_{\mathrm{G}}$: this is the set of all pairs (v, I) where v is a market node that belongs to a district that reported pest presence by December 2016. (*ii*) $\mathcal{O}_{\mathrm{B}}$: this is the set of (v, I) for all nodes v. This is the baseline which assumes no observational data on infections.

The results of the origin inference experiments are shown in Figure 5.7. Firstly, we observed that for both criteria $\mathcal{O}_{\mathrm{G}}$ and $\mathcal{O}_{\mathrm{B}}$, the top few ranks are relatively robust to varying network and model parameters. This is discussed in more detail in Sensitivity Analysis. Also, for both criteria, markets from the Central Development Region (CDR) that belong to Kathmandu and its adjacent districts Lalitpur, Bhaktapur and Kavrepalanchok are among the top ranked nodes. This is in agreement with ground truth, despite the fact that none of these nodes are assigned the highest production in our model. This can be explained by the fact that Kathmandu and nearby districts combined together dominate the production. Also, they are centrally located in the network. Interestingly, for the criterion $\mathcal{O}_{\mathrm{G}}$, Dhankuta, with the highest assigned production has a very low rank (Figure 5.7(a)) and even lower ϕ value compared to the top market in $\mathcal{O}_{\mathrm{G}}$. However, for $\mathcal{O}_{\mathrm{B}}$, it is ranked second (Figure 5.7(b)). This clearly shows that while Dhankuta has the potential to infect a large number of areas, given what has been observed, it is very unlikely that it was the source of infection. Dhankuta reported presence of the pest only towards the end of 2016 (see Figure 5.6(b)). On the other hand, markets close to Kathmandu–Charaundi and Thaha Municipality, have average ranks comparable to the top markets with respect to $\mathcal{O}_{\mathrm{G}}$, but much lower average ranks with respect to $\mathcal{O}_{\mathrm{B}}$. Thus, using the principled approach of epidemic origin inference formulation, one can demonstrate the correlation between trade patterns and observational data on *T. absoluta* incidence.

5.3.3 GENEUS Facilitates Sensitivity Analysis

As described earlier, the simulation model consists of tow sub-models: a gravity model producing the seasonal commodity flow in Nepal and a dynamic message passing-based SI model predicting the pest spread pattern. Each sub-model has several uncertain parameters, see Table 5.3 for summary. It is essential to quantitatively analyze the effect of these uncertainty parameters, which is a challenging task because this is a complex simulation workflow involving many data dependencies and intricate model interactions. For the sake of efficient experimentation, we use GENEUS to streamline the simulation



(a) ***(b)***

Figure 5.7: *Evaluating the spread model using epidemic source inference framework. (a) The average rank of each market based on the likelihood for the criterion O_G for a range of model parameters (see Table 5.4). (b) Same as (a), but for criterion O_B.*

pipeline via the standardized model configuration file (Figure 5.9) and a model specific wrapper code that drives the entire workflow, see Figure 5.8.

Symbol/Abbrev.	Description
S1	Season 1: June to November
S2	Season 1: December to May
β	Power-law exponent of gravity model
κ	Cutoff time of gravity model
γ	Per capita income parameter
σ	Gaussian kernel parameter for spatial seeding
t	Time step for the spread model

Table 5.3: *Notation and abbreviations.*

5.3.4 Sensitivity Analysis Results

We studied the sensitivity of individual market ranks as well as rank lists to network parameters (β, κ, γ), and diffusion model parameters (σ, t) (see Table 5.3 for definitions). The joint parameter effects are shown in Figure 5.7, while the single parameter results are provided in Figure 5.10. We found that the market ranks are more sensitive to spatial

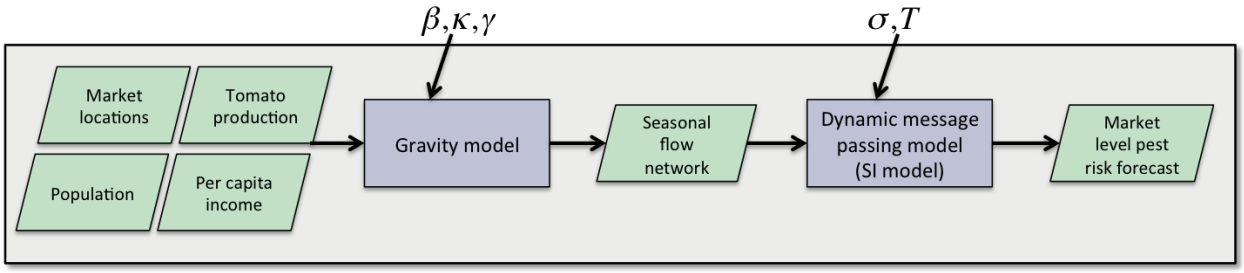


Figure 5.8: *GENEUS streamlines the simulation workflow*

seeding parameter σ and distance exponent β than other parameters. In particular, we observed that the sensitivity was highest when $\sigma = 0$ was included in the analysis. In this case (and in general for very low values of σ), substantial spread occurs only when the seed node is a source. Even if a node is in close proximity to several sources (such as Kathmandu), there is hardly any spread. This is unrealistic in the context of pest and pathogen dispersal. Hence, we restricted σ to be greater than 0 in our analysis. Also, we observe that the variance in rank is small for higher ranked nodes. This can be seen in Figure 5.7, and is more pronounced in the single parameter analyses. This property gives higher confidence in interpreting the results on top markets.

For rank list stability, we used Spearman's rank correlation coefficient (also known as Spearman's rho). It assesses monotonic relationships between rank lists and the value ranges between -1 and $+1$. A positive correlation coefficient indicates a positive relationship and vice versa, and two identical rank lists have a correlation of 1. Here we use the rank list that results from configuration ($\beta = 2,\ \kappa = 500,\ \gamma = 0,\ \sigma = 5,\ T = 10$) as the reference and calculate the Spearman's rho value with respect to it for rank lists induced by other parameter settings. Table 5.4 gives the Analysis of Variance (ANOVA) results. Under 95% confidence level, p-value < 0.05 means that the particular parameter has a significant effect. Therefore, we see that β and σ have significant effects, while others do not. Here, we note that this is despite not considering $\sigma = 0$ in the analysis.

Parameter	Description	Levels	t-ratio	F-value	p-value
β	Power-law exponent of gravity model	[0, 1, 2]	-5.16	26.6059	< 0.0001
σ	Gaussian kernel parameter for spatial seeding	[0, 5, 10, 15, 20]	-3.29	10.8424	< 0.0001
κ	Cutoff time of gravity model	[100, 500, 1000]	-0.42	0.1758	0.6753
γ	Per capita income parameter	[0, 0.5, 1.0]	0.89	0.7970	0.3727
T	Time step for the spread model	[5, 10, 20]	1.14	1.2976	0.2556

Table 5.4: *Analyzing sensitivity to model parameters using ANOVA.*

```
<configuration>
 <parameters>
  <parameter exposed="false" key="gravity_model_cfg"
    optional="false" type="string">
   <value>gravity_cfg_template</value>
   <parameter exposed="true" key="node_attr_file"
     optional="false" type="string">
    <value>../data/market_node_attr_season1.csv</value>
   </parameter>
   <parameter exposed="false" key="network_file"
     optional="false" type="string">
    <value>../data/nepal_market_network.csv</value>
   </parameter>
   <parameter exposed="true" key="beta" optional="false" type="float">
    <value>2</value>
   </parameter>
   <parameter exposed="true" key="kappa" optional="false" type="float">
    <value>500</value>
   </parameter>
   <parameter exposed="false" key="gravity_output"
     optional="false" type="float">
    <value>dummy</value>
   </parameter>
  </parameter>
  <parameter exposed="false" key="message_passing_cfg"
    optional="false" type="string">
   <value>message_passing_cfg</value>
   <parameter exposed="true" key="sigma"
     optional="false" type="float">
    <value>5</value>
   </parameter>
   <parameter exposed="true" key="T" optional="false" type="int">
    <value>10</value>
   </parameter>
  </parameter>
 </parameters>
</configuration>
```

Figure 5.9: *Base configuration file for the pest spread model.*

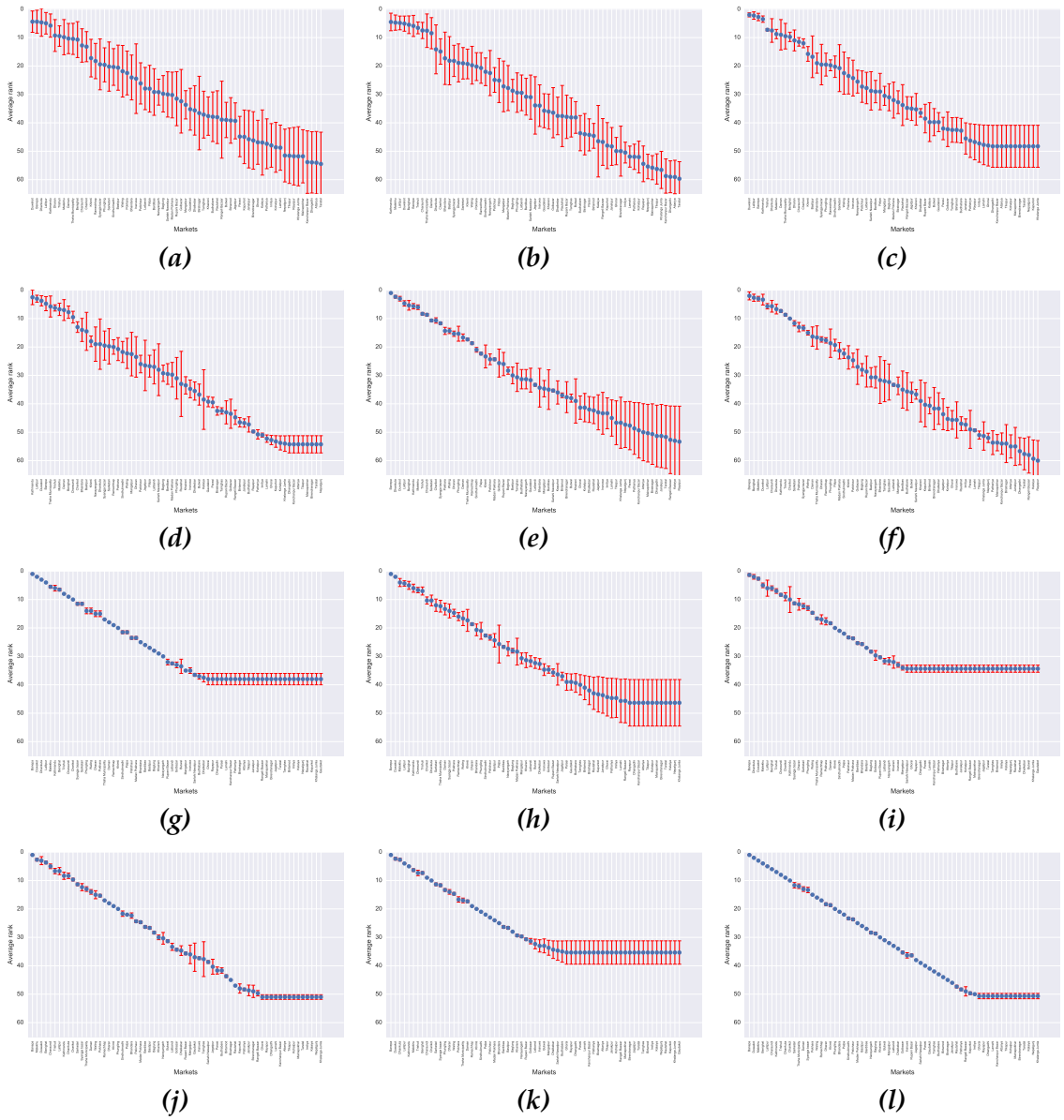


Figure 5.10: *Single parameter effects on stability of individual market ranks. Blue dots represent mean rank for each market and error bars are standard deviation. (a) Joint parameter effect based on O_B. (b) Joint parameter effect based on O_G. (c) Parameter effect of σ based on O_B. (d) Parameter effect of σ based on O_G. (e) Parameter effect of β based on O_B. (f) Parameter effect of β based on O_G. (g) Parameter effect of κ based on O_B. (h) Parameter effect of κ based on O_G. (i) Parameter effect of γ based on O_B. (j) Parameter effect of γ based on O_G. (k) Parameter effect of T based on O_B. (l) Parameter effect of T based on O_G.*

5.4 Summary

Typically, network-based simulation models have complex simulation workflows involving may data dependencies and intricate model interaction. Experimentation and quantitatively analysis of the uncertainties of them are challenging, especially for those who does not have solid computational background. In this chapter, we present two application case studies with large scale network-based simulation models. These case studies can illustrate the usability and the capability of GENEUS system.

Chapter 6

Conclusion

Uncertainty quantification is essential in model validation . To grantee the model credibility, molders are required to identify the uncertainty sources, quantify their effects, control and reduce them. However, it is a non-trivial task due to both of the lack of knowledge and the inherent indeterminate characters of the model. Evaluating model uncertainties often involves comprehensive parameter sensitivity analysis and large experimental design. This brings more challenges for network-based simulation models because their data and computationally intensive feature leads to many technical challenges and prevents running a large number of experiments.

Here we present a novel approach to facilitate the model validation and uncertainty quantification for network-based simulation models in this dissertation. This approach includes a software package – GENEUS that supports for UQ/SA/DOE studies for a range of network-based simulation models, as well as theoretical contributions in the stability theory of fundamental graph dynamical system, which serves as an important mathematical foundation of network-based simulation models.

As a summary of the dissertation. In Chapter 1, we introduce the research background, motivation, challenges, questions, and contributions. In Chapter 2, we review the state-of-the-art in this field and discuss existing related work. Chapter 3 presents the basic design concept and philosophy of GENEUS system as well as implementation details. We remark that the development of the standardized model configuration format, integration of a central registry system and design of the loosely-coupled system architecture make GENEUS a general extensible ecosystem. In Chapter 4, we establish sufficient stability criteria for a collection of graph dynamical systems stability assessments will allow one to analyze system robustness under perturbations of system components including vertex state, vertex function, graph structure and vertex state update scheme. Since graph

dynamical system is an important mathematical foundation of network-based simulation models, stability theory in GDS relates tightly with model validation researches. This serves as an important motivation and theoretical guidance for the design of GENEUS system. Finally, in Chapter 5 we present two application case studies to illustrate the capability and usability of GENEUS.

We admit that there are known limitations GENEUS system. First, extra efforts are still needed such as developing model-specific wrapper code to couple GENEUS with simulation models. Second, GENEUS itself does not depend on the particular platform, however, the simulation models may have certain requirements and dependencies on external infrastructures which need to be resolved before conducting experiments. Finally, the computational complexity depends largely on the simulation model itself, this may remain a challenge when complex model and a large number of experiment runs are involved.

The future research direction includes: enriching GENEUS system by integrating more potential analysis methods; integrating a graphical user interface (GUI) to achieve a better user experience; exploring UQ/SA researches in the context of new application scenarios and theoretic graph dynamical systems.

Appendix A

Standardized Configuration Format Grammar

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://ndssl.vbi.vt.edu"
 xmlns="http://ndssl.vbi.vt.edu"
 elementFormDefault="qualified">
 <xsd:include schemaLocation="registryIdType.xsd"/>
 <xsd:element name="configSpec">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="registryInfo">
    <!-- Information associated with registry-->
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="configurationName" type="xsd:string"/>
       <xsd:element name="methodName" type="xsd:string"/>
       <xsd:element name="description" type="xsd:string"/>
       <xsd:element name="baseConfigFile" type="registryIdType"/>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="configuration">
    <!-- Configuration information -->
```

```
      <xsd:complexType>
       <xsd:sequence>
        <xsd:element name="parameters"
        type="parameterListType"
        minOccurs="0"/>
       </xsd:sequence>
      </xsd:complexType>
     </xsd:element>
    </xsd:sequence>
   </xsd:complexType>
 </xsd:element>

  <!-- Type definition -->

  <xsd:complexType name="parameterListType">
   <xsd:sequence>
    <xsd:element name="parameter"
    type="parameterType" minOccurs="0"
    maxOccurs="unbounded"/>
   </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="parameterType">
   <xsd:sequence>
    <xsd:element name="value" type="xsd:string"/>
    <xsd:element name="parameter"
    type="parameterType" minOccurs="0"
    maxOccurs="unbounded"/>
   </xsd:sequence>
    <xsd:attribute name="key" type="xsd:string" use="required"/>
    <xsd:attribute name="type" type="xsd:string" use="required"/>
    <xsd:attribute name="exposed" type="xsd:boolean" use="required"/>
    <xsd:attribute name="optional" type="xsd:boolean" use="required"/>
    <xsd:attribute name="default" type="xsd:string" use="optional"/>
   </xsd:complexType>
 </xsd:schema>
```

Appendix B

XSD for GENEUS Configuration File

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://ndssl.vbi.vt.edu"
 xmlns="http://ndssl.vbi.vt.edu"
 elementFormDefault="qualified">

 <!-- Include external schema -->
 <xsd:include schemaLocation="registryIdType.xsd"/>
 <xsd:include schemaLocation="doeType.xsd"/>
 <xsd:include schemaLocation="saType.xsd"/>

 <!-- Schema definition -->
 <xsd:element name="geneus_config">
  <xsd:complexType>
   <xsd:element name="study" type="studyType"
    minOccurs="1" maxOccurs="1"/>
   <xsd:element name="param_spec" type="paramSpecType"
    minOccurs="1" maxOccurs="1"/>
   <xsd:element name="model_spec" type="modelSpecType"
    minOccurs="1" maxOccurs="1"/>
   <xsd:element name="model_spec" type="modelSpecType"
    minOccurs="1" maxOccurs="1"/>
   <xsd:element name="doe" type="doeType"
    minOccurs="0" maxOccurs="1"/>
```

```
    <xsd:element name="sa" type="saType"
     minOccurs="0" maxOccurs="1"/>
   </xsd:complexType>
  </xsd:element>

  <!-- Type definition -->
  <xsd:complexType name="studyType">
   <xsd:sequence>
    <xsd:element name="study_name" type="xsd:string"
     minOccurs="1" maxOccurs="1"/>
    <xsd:element name="work_dir" type="xsd:string"
     minOccurs="1" maxOccurs="1"/>
    <xsd:element name="method" type="xsd:string"
     minOccurs="1" maxOccurs="1"/>
    <xsd:element name="base_config" type="registryID"
     minOccurs="1" maxOccurs="1"/>
   </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="param_spec">
   <xsd:sequence>
    <xsd:element name="continuous_parameters" tpye="contParamType"
     minOccurs="0" maxOccurs="1"/>
    <xsd:element name="discrete_parameters" tpye="discreteParamType"
     minOccurs="0" maxOccurs="1"/>
    <xsd:element >
   </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="contParamType">
   <xsd:element name="parameter" minOccurs="0" maxOccurs="unbounded">
    <xsd:complexType>
     <xsd:sequence>
      <xsd:element name="data_type" tpye="xsd:string"
       minOccurs="1" maxOccurs="1"/>
      <xsd:element name="lower_bound" tpye="xsd:float"
       minOccurs="1" maxOccurs="1"/>
      <xsd:element name="upper_bound" tpye="xsd:float"
```

```
        minOccurs="1" maxOccurs="1"/>
       <xsd:element name="distribution"  minOccurs="1" maxOccurs="1">
        <xsd:simpleType>
         <xsd:restriction>
          <xsd:enumeration value="SWEEP"/>
          <xsd:enumeration value="U"/>
          <xsd:enumeration value="N"/>
         </xsd:restriction>
        </xsd:simpleType>
       </xsd:element>
       <xsd:element name="interval" tpye="xsd:int"
        minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:string" use="required"/>
     </xsd:complexType>
    </xsd:element>
   </xsd:complexType>


   <xsd:complexType name="discreteParamType">
    <xsd:element name="parameter" minOccurs="0" maxOccurs="unbounded">
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="data_type" tpye="xsd:string"
        minOccurs="1" maxOccurs="1"/>
       <xsd:element name="value" tpye="xsd:string"
        minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:string" use="required"/>
     </xsd:complexType>
    </xsd:element>
   </xsd:complexType>
  </xsd:schema>
```

Bibliography

[1] Brian Adams, Mohamed Ebeida, Michael Eldred, John Jakeman, Laura Swiler, Adam Stephens, Dena Vigil, and Timothy Wildey. *DAKOTA, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.0 User's Manual*. Tech. rep. Sandia Technical Report SAND2014-4633. Sandia National Laboratory, 2014 (pages 6, 9, 13, 16, 17).

[2] Abhijin Adiga, Henning S Mortveit, and Sichao Wu. "Route Stability in Large-Scale Transportation Models." In: *Workshop on Multiagent Interaction Networks, AAMAS*. Vol. 13. 2012 (pages 3, 4, 7, 46).

[3] Abhijin Adiga and Anil Vullikanti. "Temporal Vaccination Games under Resource Constraints." In: *AAAI*. 2016, pp. 2438–2444 (page 74).

[4] Abhijin Adiga, Hilton Galyean, Chris Kuhlman, Michael Levet, Henning S Mortveit, and Sichao Wu. "Activity in Boolean Networks." In: *Natural Computing* (2016). accepted (pages 46, 57, 59, 63, 64, 67).

[5] Abhijin Adiga, Chris J Kuhlman, Henning S Mortveit, and Sichao Wu. "Effect of Graph Structure on the Limit Sets of Threshold Dynamical Systems." In: *Cellular Automata and Discrete Complex Systems*. Springer, 2015, pp. 59–70 (pages 15, 46, 53, 54, 56).

[6] Abhijin Adiga, Madhav V Marathe, Henning S Mortveit, Sichao Wu, and Samarth Swarup. "Modeling Urban Transportation in the Aftermath of a Nuclear Disaster: The Role of Human Behavioral Responses." In: *Transportation Research: Part C (International Journal of Transportation)*. 2013 (pages 7, 46).

[7] Abhijin Adiga, Hilton Galyean, Chris J Kuhlman, Michael Levet, Henning S Mortveit, and Sichao Wu. "Network Structure and Activity in Boolean Networks." In: *Cellular Automata and Discrete Complex Systems*. Springer, 2015, pp. 210–223 (page 46).

[8] Abhijin Adiga, Hilton Galyean, Chris J Kuhlman, Michael Levet, Henning S Mortveit, and Sichao Wu. "Network Structure and Activity in Boolean Networks." In: *Cellular Automata and Discrete Complex Systems*. Springer, 2015, pp. 210–223 (pages 63, 64, 67).

[9] Abhijin Adiga, Chris J Kuhlman, Henning S Mortveit, and Anil Vullikanti. "Sensitivity of Diffusion Dynamics to Network Uncertainty." In: *J. Artif. Intell. Res.(JAIR)* 51 (2014), pp. 207–226 (page 16).

[10] *AlgDesign. Available at* `https://cran.r-project.org/web/packages/AlgDesign/index.html`. Last accessed: June 2017 (page 17).

[11] Mubarik Ali. "Dynamics of vegetables in Asia: a synthesis." In: *Dynamics of vegetable production, distribution and consumption in Asia. Asian Vegetable Research and Development Center. AVRDC publication no. 00-498* (2000), pp. 1–29 (page 87).

[12] Fabrizio Altarelli, Alfredo Braunstein, Luca Dall'Asta, Alejandro Lage-Castellanos, and Riccardo Zecchina. "Bayesian inference of epidemics on networks via belief propagation." In: *Physical Review Letters* 112.11 (2014), p. 118701 (pages 88, 89).

[13] Ilkay Altintas, Chad Berkley, Efrat Jaeger, Matthew Jones, Bertram Ludascher, and Steve Mock. "Kepler: an extensible system for design and execution of scientific workflows." In: *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*. IEEE. 2004, pp. 423–424 (page 17).

[14] Todd R Andel and Alec Yasinsac. "On the credibility of manet simulations." In: *Computer* 39.7 (2006), pp. 48–54 (page 33).

[15] James E Anderson. "The gravity model." In: *Annual Review of Economics* 3.1 (2011), pp. 133–160 (page 87).

[16] Jan M Baetens and Bernard De Baets. "Phenomenological study of irregular cellular automata based on Lyapunov exponents and Jacobians." In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 20.3 (2010), p. 033112 (pages 2, 58).

[17] Jan M Baetens, Pieter Van der Weeën, and Bernard De Baets. "Effect of asynchronous updating on the stability of cellular automata." In: *Chaos, Solitons & Fractals* 45.4 (2012), pp. 383–394 (pages 2, 15, 16, 58).

[18] Ajaya Shree Ranta Bajracharya, Ram Prasad Mainali, Binu Bhat, Sanjaya Bista, PR Shashank, and NM Meshram. "The first record of South American tomato leaf miner, Tuta absoluta (Meyrick 1917)(Lepidoptera: Gelechiidae) in Nepal." In: *J. Entomol. Zool. Stud* 4 (2016), pp. 1359–1363 (pages 86, 89).

[19] Masako Bando, Katsuya Hasebe, Akihiro Nakayama, Akihiro Shibata, and Yuki Sugiyama. "Dynamical model of traffic congestion and numerical simulation." In: *Physical Review E* 51.2 (1995), p. 1035 (page 1).

[20] Natalie Clare Banks, Dean Ronald Paini, Kirsty Louise Bayliss, and Michael Hodda. "The role of global trade and transport network topology in the human-mediated dispersal of alien species." In: *Ecology letters* 18.2 (2015), pp. 188–199 (page 85).

[21] C. L. Barrett, K. Bisset, S. Eubank, V. S. A. Kumar, M. V. Marathe, and H. S. Mortveit. "Modeling and Simulation of Large Biological, Information and Soci-Technical Systems: An Interaction-Based Approach." In: *Modeling and Simulation of Biological Networks*. Vol. 64. Proceedings of Symposia in Applied Mathematics. American Mathematical Society, 2007, pp. 101–148 (page 46).

[22] Chris L Barrett, Henning S Mortveit, and Christian M Reidys. "Sequential dynamical systems." In: *Artificial Life and Robotics* 6.4 (2002), pp. 167–169 (page 15).

[23] Christopher L. Barrett, Harry B. Hunt III, Madhav V. Marathe, S. S. Ravi, Daniel J. Rosenkrantz, and Richard Edwin Stearns. "Complexity of Reachability Problems for Finite Discrete Sequential Dynamical Systems." In: *J. Computer and System Sciences* 72 (2006), pp. 1317–1345 (pages 16, 51, 52, 54).

[24] Christopher L Barrett, Keith R Bisset, Stephen G Eubank, Xizhou Feng, and Madhav V Marathe. "EpiSimdemics: an efficient algorithm for simulating the spread of infectious disease over large realistic social networks." In: *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*. IEEE Press. 2008, p. 37 (page 75).

[25] Christopher L. Barrett, Harry B. Hunt III, Madhav V. Marathe, S. S. Ravi, Daniel J. Rosenkrantz, Richard Edwin Stearns, and Mayur Thakur. "Predecessor Existence Problems for Finite Discrete Sequential Dynamical Systems." In: *Theoretical Computer Science* 386 (1–2 2006), pp. 3–37 (page 16).

[26] Russell R Barton, Barry L Nelson, and Wei Xie. "Quantifying input uncertainty via simulation confidence intervals." In: *INFORMS Journal on Computing* 26.1 (2013), pp. 74–87 (page 13).

[27] Ola Ghazi Batarseh and Yan Wang. "Reliable simulation with input uncertainties using an interval-based approach." In: *Simulation Conference, 2008. WSC 2008. Winter*. IEEE. 2008, pp. 344–352 (page 13).

[28] Nils Bengtsson, Guodong Shao, Björn Johansson, Y Tina Lee, Swee Leong, Anders Skoogh, and Charles Mclean. "Input data management methodology for discrete event simulation." In: *Proceedings of the 2009 winter simulation conference (WSC)*. IEEE. 2009, pp. 1335–1344 (page 32).

[29] Alan A Berryman. "Biological control, thresholds, and pest outbreaks." In: *Environmental Entomology* 11.3 (1982), pp. 544–549 (page 48).

[30] Keith R. Bisset, Jose Cadena, Maleq Khan, Christopher J. Kuhlman, Bryan L. Lewis, and Pyrros Telionis. "An Integrated Agent-Based Approach for Modeling Disease Spread in Large Populations to Support Health Informatics." In: *IEEE International Conference on Biomedical and Health Informatics*. Las Vegas, NV, 2016 (page 74).

[31] Keith R Bisset, Jiangzhuo Chen, Xizhou Feng, VS Kumar, and Madhav V Marathe. "EpiFast: a fast algorithm for large scale realistic epidemic simulations on distributed memory systems." In: *Proceedings of the 23rd international conference on Supercomputing*. ACM. 2009, pp. 430–439 (pages 11, 21, 46, 75).

[32] Keith R Bisset, Jiangzhuo Chen, Xizhou Feng, Yifei Ma, and Madhav V Marathe. "Indemics: an interactive data intensive framework for high performance epidemic simulation." In: *Proceedings of the 24th ACM International Conference on Supercomputing*. ACM. 2010, pp. 233–242 (page 75).

[33] Keith R Bisset, Suruchi Deodhar, Hemanth Makkapati, Madhav V Marathe, Paula Stretz, and Christopher L Barrett. "Simfrastructure: A Flexible and Adaptable Middleware Platform for Modeling and Analysis of Socially Coupled Systems." In: *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*. IEEE. 2013, pp. 506–513 (page 24).

[34] Barry W Boehm, Ray Madachy, Bert Steece, et al. *Software cost estimation with Cocomo II with Cdrom*. Prentice Hall PTR, 2000 (page 42).

[35] Emanuele Borgonovo. "A new uncertainty importance measure." In: *Reliability Engineering & System Safety* 92.6 (2007), pp. 771–784 (page 80).

[36] George EP Box and Donald W Behnken. "Some new three level designs for the study of quantitative variables." In: *Technometrics* 2.4 (1960), pp. 455–475 (page 36).

[37] Leo Breiman. "Random forests." In: *Machine learning* 45.1 (2001), pp. 5–32 (page 38).

[38] T Butler, C Dawson, and T Wildey. "Propagation of uncertainties using improved surrogate models." In: *SIAM/ASA Journal on Uncertainty Quantification* 1.1 (2013), pp. 164–191 (page 14).

[39] AC Calder, B Fryxell, T Plewa, R Rosner, LJ Dursi, VG Weirs, T Dupont, HF Robey, JO Kane, BA Remington, et al. "On validating an astrophysical simulation code." In: *The Astrophysical Journal Supplement Series* 143.1 (2002), p. 201 (page 1).

[40] Mateus R Campos, Antonio Biondi, Abhijin Adiga, Raul NC Guedes, and Nicolas Desneux. "From the Western Palaearctic region to beyond: *Tuta absoluta* ten years after invading Europe." In: *Journal of Pest Science* (2017) (page 86).

[41] Damon Centola and Michael Macy. "Complex contagions and the weakness of long ties1." In: *American Journal of Sociology* 113.3 (2007), pp. 702–734 (page 48).

[42] Parimal Pal Chaudhuri. *Additive cellular automata: theory and applications*. Vol. 1. John Wiley & Sons, 1997 (page 15).

[43] Xi Chen and Qiang Zhou. "Sequential experimental designs for stochastic kriging." In: *Proceedings of the 2014 Winter Simulation Conference*. IEEE Press. 2014, pp. 3821–3832 (page 14).

[44] Hugh Chipman, Pritam Ranjan, and Weiwei Wang. "Sequential design for computer experiments with a flexible Bayesian additive model." In: *Canadian Journal of Statistics* 40.4 (2012), pp. 663–678 (page 14).

[45] Committee on Mathematical Foundations of Verification, Validation and Uncertainty Quantification, Board on Mathematical Sciences and Their Applications, and Division on Engineering and Physical Sciences. *Assessing the Reliability of Complex Models: Mathematical and Statistical Foundations of Verification, Validation, and Uncertainty Quantification*. Washington, D.C.: The National Academies Press, 2012 (pages 1, 3, 73).

[46] Maria I Davidich and Stefan Bornholdt. "Boolean network model predicts cell cycle sequence of fission yeast." In: *PloS one* 3.2 (2008), e1672 (page 47).

[47] Xinwei Deng, V Roshan Joseph, Agus Sudjianto, and CF Jeff Wu. "Active learning through sequential design, with applications to detection of money laundering." In: *Journal of the American Statistical Association* 104.487 (2009) (page 14).

[48] Bernard Derrida and Yves Pomeau. "Random networks of automata: a simple annealed approximation." In: *EPL (Europhysics Letters)* 1.2 (1986), p. 45 (page 58).

[49] Nicolas Desneux, Eric Wajnberg, Kris AG Wyckhuys, Giovanni Burgio, Salvatore Arpaia, Consuelo A Narváez-Vasquez, Joel González-Cabrera, Diana Catalán Ruescas, Elisabeth Tabone, Jacques Frandon, et al. "Biological invasion of European tomato crops by *Tuta absoluta*, ecology, geographic expansion and prospects for biological control." In: *Journal of Pest Science* 83.3 (2010), pp. 197–215 (page 86).

[50] Reinhard Diestel. *Graph Theory*. 2nd ed. Springer, 2000 (page 54).

[51] Regan Early, Bethany A Bradley, Jeffrey S Dukes, Joshua J Lawler, Julian D Olden, Dana M Blumenthal, Patrick Gonzalez, Edwin D Grosholz, Ines Ibañez, Luke P Miller, et al. "Global threats from invasive alien species in the twenty-first century and national response capacities." In: *Nature Communications* 7 (2016) (page 85).

[52] Michael Scott Eldred, Clayton Garrett Webster, and Paul Constantine. "Evaluation of non-intrusive approaches for Wiener-Askey generalized polynomial chaos." In: *Proceedings of the 10th AIAA Non-Deterministic Approaches Conference, number AIAA-2008-1892, Schaumburg, IL*. Vol. 117. 2008, p. 189 (page 13).

[53] S. Eubank, H. Guclu, V. S. A. Kumar, M. V. Madhav, A. Srinivasan, Z. Toroczkai, and N. Wang. "Modelling disease outbreaks in realistic urban social networks." In: *Nature* 429.6988 (2004), pp. 180–184 (pages 45, 46).

[54] Roland Ewald, Jan Himmelspach, Matthias Jeschke, Stefan Leye, and Adelinde M Uhrmacher. "Flexible experimentation in the modeling and simulation framework JAMES IIâĂŤimplications for computational systems biology." In: *Briefings in bioinformatics* 11.3 (2010), pp. 290–300 (page 17).

[55] Kai-Tai Fang, Runze Li, and Agus Sudjianto. *Design and modeling for computer experiments*. CRC Press, 2005 (pages 2, 14).

[56] Alexander Felfernig. "Standardized configuration knowledge representations as technological foundation for mass customization." In: *IEEE Transactions on Engineering Management* 54.1 (2007), pp. 41–56 (page 25).

[57] Malcolm R Forster. "Key concepts in model selection: Performance and generalizability." In: *Journal of mathematical psychology* 44.1 (2000), pp. 205–231 (page 37).

[58] Christoph Fretter, Agnes Szejka, and Barbara Drossel. "Perturbation propagation in random and evolved Boolean networks." In: *New Journal of Physics* 11.3 (2009), p. 033005 (pages 2, 16, 58).

[59] *GaLib*. *Available at* `http://cinet.vbi.vt.edu/granite/granite.html`. Last accessed: June 2017 (page 16).

[60] A. Ganesh, L. Massoulié, and D. Towsley. "The effect of network topology on the spread of epidemics." In: *IEEE Infocom*. Miami, FL, 2005 (page 2).

[61] Frederick Geiger, Michael Klybor, and William Rosenfeld. *Commercial data registry system*. US Patent App. 09/753,068. 2000 (page 17).

[62] E. Goles and J. Olivos. "Comportement périodique des fonctions à seuil binaires et applications." In: *Discrete Applied Mathematics* 3.2 (1981), pp. 93–105 (pages 51, 52).

[63] Eric Goles and Servet Martinez. *Neural and Automata Networks : Dynamical Behavior and Applications*. Vol. 58. Mathematics and its Applications. Kluwer Academic, 1990 (pages 45, 52, 53).

[64] Eric Goles and Servet Martínez. *Neural and automata networks: dynamical behavior and applications*. Vol. 58. Springer Science & Business Media, 2013 (page 15).

[65] Eric Goles and Pedro Montealegre. "Computational complexity of threshold automata networks under different updating schemes." In: *Theoretical Computer Science* 559.0 (2014). Non-uniform Cellular Automata, pp. 3 –19 (page 53).

[66] Eric Goles-Chacc, Franҫoise Fogelman-Soulie, and Didier Pellegrin. "Decreasing Energy Functions as a Tool for Studying Threshold Networks." In: *Discrete Applied Mathematics* 12 (1985), pp. 261–277 (page 51).

[67] PC Gontijo, MC Picanço, EJG Pereira, JC Martins, M Chediak, and RNC Guedes. "Spatial and temporal variation in the control failure likelihood of the tomato leaf miner, *Tuta absoluta*." In: *Annals of Applied Biology* 162.1 (2013), pp. 50–59 (page 86).

[68] Jim Gray, David Liu, Maria Nieto-Santisteban, Alex Szalay, David DeWitt, and Gerd Heber. "Scientific data management in the coming decade." In: *ACM SIGMOD Record* 34.4 (2005), pp. 34–41 (page 17).

[69] F Grousset, M Suffert, and F Petter. "EPPO Study on pest risks associated with the import of tomato fruit." In: *EPPO Bulletin* 45.1 (2015), pp. 153–156 (page 86).

[70] RNC Guedes and HAA Siqueira. "The tomato borer *Tuta absoluta*: insecticide resistance and control failure." In: *Plant Sciences Reviews 2012* (2013), p. 245 (page 86).

[71] Elizabeth Halloran, Neil Ferguson, Stephen Eubank, Ira Longini, Derek Cummings, Bryan Lewis, Shufu Xu, Christophe Fraser, Anil Vullikanti, and Timothy Germann. "Modeling targeted layered containment of an influenza pandemic in the United States." In: *Proceedings of the National Academy of Sciences* 105.12 (2008), pp. 4639–4644 (pages 74–76).

[72] Kingsley E Haynes, A Stewart Fotheringham, et al. *Gravity and spatial interaction models*. Vol. 2. Sage Beverly Hills, CA, 1984 (page 85).

[73] A Samad Hedayat, Neil James Alexander Sloane, and John Stufken. *Orthogonal arrays: theory and applications*. Springer, 1999 (page 36).

[74] Jon C Helton, Jay Dean Johnson, Cedric J Sallaberry, and Curt B Storlie. "Survey of sampling-based methods for uncertainty and sensitivity analysis." In: *Reliability Engineering & System Safety* 91.10 (2006), pp. 1175–1209 (page 14).

[75] Shane G Henderson. "Input modeling: input model uncertainty: why do we care and what should we do about it?" In: *Proceedings of the 35th conference on Winter simulation: driving innovation*. Winter Simulation Conference. 2003, pp. 90–100 (page 13).

[76] David Heymann, Thomson Prentice, and Lina Tucker Reinders. *The world health report 2007: a safer future: global public health security in the 21st century*. World Health Organization, 2007 (page 74).

[77] Dave Higdon, James Gattiker, Brian Williams, and Maria Rightley. "Computer model calibration using high-dimensional output." In: *Journal of the American Statistical Association* (2012) (pages 14, 15, 38).

[78] Toshimitsu Homma and Andrea Saltelli. "Importance measures in global sensitivity analysis of nonlinear models." In: *Reliability Engineering & System Safety* 52.1 (1996), pp. 1–17 (page 14).

[79] K. Hou, H. Wang, and W. c. Feng. "Delivering Parallel Programmability to the Masses via the Intel MIC Ecosystem: A Case Study." In: *2014 43rd International Conference on Parallel Processing Workshops*. 2014, pp. 273–282 (page 6).

[80] Kaixi Hou, Hao Wang, and Wu-chun Feng. "Aspas: A framework for automatic simdization of parallel sorting on x86-based many-core processors." In: *Proceedings of the 29th ACM on International Conference on Supercomputing*. ACM. 2015, pp. 383–392 (page 15).

[81] Philip E Hulme. "Trade, transport and trouble: managing invasive species pathways in an era of globalization." In: *Journal of Applied Ecology* 46.1 (2009), pp. 10–18 (page 85).

[82] Chii-Ruey Hwang. "Simulated annealing: theory and applications." In: *Acta Applicandae Mathematicae* 12.1 (1988), pp. 108–111 (page 38).

[83] Andrew Ilachinski. *Cellular automata: a discrete universe*. World Scientific, 2001 (page 15).

[84] Ronald L Iman and Jon C Helton. "An investigation of uncertainty and sensitivity analysis techniques for computer models." In: *Risk analysis* 8.1 (1988), pp. 71–90 (page 14).

[85] A. Jarrah, B. Raposa, and R. Laubenbacher. "Nested Canalyzing, Unate Cascade, and Polynomial Functions." In: *Physica D* 233 (2 2007), pp. 167–174 (page 45).

[86] Abdul S. Jarrah and Reinhard Laubenbacher. "On the Algebraic Geometry of Polynomial Dynamical Systems." In: *Emerging Applications of Algebraic Geometry*. Vol. 149. The IMA Volumes in Mathematics and its Applications. Springer New York, 2009, pp. 109–123 (page 45).

[87] Pablo Kaluza, Andrea Kölzsch, Michael T Gastner, and Bernd Blasius. "The complex network of global cargo ship movements." In: *Journal of the Royal Society Interface* 7.48 (2010), pp. 1093–1103 (page 87).

[88] O Karadjova, Z Ilieva, V Krumov, E Petrova, V Ventsislavov, et al. "*Tuta absoluta* (Meyrick)(Lepidoptera: Gelechiidae): Potential for entry, establishment and spread in Bulgaria." In: *Bulgarian Journal of Agricultural Science* 19.3 (2013), pp. 563–571 (page 86).

[89] Ulas Karaoz, T. M. Murali, Stan Letovsky, Yu Zheng, Chunming Ding, Charles R. Cantor, and Simon Kasif. "Whole-genome annotation by using evidence integration in functional-linkage networks." In: *Proceedings of the National Academy of Sciences* 101.9 (2004), pp. 2888–2893 (page 47).

[90] Jarkko Kari. "Reversibility of 2D CA." In: *Physica D* 45–46 (1990), pp. 379–385 (page 16).

[91] S. A. Kauffman. "Metabolic stability and epigenesis in randomly constructed genetic nets." In: *J. Theor. Biol.* 22.3 (1969), pp. 437–467 (page 47).

[92] Stuart A. Kauffman. "Metabolic stability and epigenesis in randomly constructed genetic nets." In: *Journal of Theoretical Biology* 22.3 (1969), pp. 437–467 (pages 15, 45).

[93] Stuart A. Kauffman. *The origins of order: Self organization and selection in evolution*. Oxford University Press, USA, 1993 (page 45).

[94] David Kempe, Jon Kleinberg, and Éva Tardos. "Maximizing the spread of influence through a social network." In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2003, pp. 137–146 (page 48).

[95] Marc C Kennedy and Anthony O'Hagan. "Bayesian calibration of computer models." In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63.3 (2001), pp. 425–464 (page 15).

[96] Pavel Kindlmann. *Himalayan biodiversity in the changing world*. Springer Science & Business Media, 2011 (page 86).

[97] Jack PC Kleijnen. "An overview of the design and analysis of simulation experiments for sensitivity analysis." In: *European Journal of Operational Research* 164.2 (2005), pp. 287–300 (page 14).

[98] Konstantin Klemm and Stefan Bornholdt. "Stable and unstable attractors in Boolean networks." In: *Physical Review E* 72.5 (2005), p. 055101 (page 16).

[99] Chris Kuhlman, Henning S. Mortveit, David Murrugarra, and V. S. Anil Kumar. "Bifurcations in Boolean Networks." In: *Automata 2011 – 17th International Workshop on Cellular Automata and Discrete Complex Systems, 21–23 November, 2011, Santiago, Chile*. 2012, pp. 29–46 (pages 48, 51, 52).

[100] Chris J Kuhlman and Henning S Mortveit. "Attractor stability in nonuniform Boolean networks." In: *Theoretical Computer Science* 559 (2014), pp. 20–33 (pages 2, 16, 58, 68).

[101] Setsuya Kurahashi. "A Health Policy Simulation Model of Ebola Haemorrhagic Fever and Zika Fever." In: *Agent and Multi-Agent Systems: Technology and Applications*. Springer, 2016, pp. 319–329 (page 74).

[102] L. Layne, E.S. Dimitrova, and M. Macauley. "Nested canalyzing depth and network stability." In: *Bull. Math. Biol.* In press (2012) (pages 58, 60).

[103] Olivier P Le Maître and Omar M Knio. *Introduction: Uncertainty Quantification and Propagation*. Springer, 2010 (page 2).

[104] Thomas Liggett. *Interacting particle systems*. Vol. 276. Springer Science & Business Media, 2012 (page 15).

[105] Andrey Y Lokhov, Marc Mézard, Hiroki Ohta, and Lenka Zdeborová. "Inferring the origin of an epidemic with a dynamic message-passing algorithm." In: *Physical Review E* 90.1 (2014), p. 012801 (pages 88, 89).

[106] Jamie X Luo and Matthew S Turner. "Evolving sensitivity balances boolean networks." In: *PLoS One* 7.5 (2012), e36010 (page 58).

[107] Matthew Macauley and Henning S. Mortveit. "Cycle Equivalence of Graph Dynamical Systems." In: *Nonlinearity* 22 (2009), pp. 421–436 (page 70).

[108] Matthew Macauley and Henning S Mortveit. "Update sequence stability in graph dynamical systems." In: *arXiv preprint arXiv:0909.1723* (2009) (page 16).

[109] Roger Duncan Magarey, DM Borchert, JS Engle, M Colunga-Garcia, Frank H Koch, and D Yemshanov. "Risk maps for targeting exotic plant pest detection programs in the United States." In: *EPPO Bulletin* 41.1 (2011), pp. 46–56 (page 85).

[110] David Thomas Magill. "Optimal adaptive estimation of sampled stochastic processes." In: *Automatic Control, IEEE Transactions on* 10.4 (1965), pp. 434–439 (page 14).

[111] Stefano Marelli and Bruno Sudret. *UQLab: a framework for uncertainty quantification in MATLAB*. ETH-Zürich, 2014 (page 17).

[112] Michael D McKay, Richard J Beckman, and William J Conover. "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code." In: *Technometrics* 21.2 (1979), pp. 239–245 (pages 14, 36).

[113] Melanie Mitchell. *An introduction to genetic algorithms*. MIT press, 1998 (page 38).

[114] Parviz Moin and Krishnan Mahesh. "Direct numerical simulation: a tool in turbulence research." In: *Annual review of fluid mechanics* 30.1 (1998), pp. 539–578 (page 1).

[115] Douglas C Montgomery, Douglas C Montgomery, and Douglas C Montgomery. *Design and analysis of experiments*. Vol. 7. Wiley New York, 1984 (pages 36, 38).

[116] Max D Morris. "Factorial sampling plans for preliminary computational experiments." In: *Technometrics* 33.2 (1991), pp. 161–174 (page 38).

[117] Henning Mortveit and Christian Reidys. *An introduction to sequential dynamical systems*. Springer, 2007 (page 15).

[118] Henning S Mortveit. "Limit cycle structure for block-sequential threshold systems." In: *Cellular Automata*. Springer, 2012, pp. 672–678 (pages 15, 51, 52, 54).

[119] James M Murphy, David MH Sexton, David N Barnett, Gareth S Jones, Mark J Webb, Matthew Collins, and David A Stainforth. "Quantification of modelling uncertainties in a large ensemble of climate change simulations." In: *Nature* 430.7001 (2004), pp. 768–772 (page 14).

[120] *NetworkX*. *Available at* `https://networkx.github.io`. Last accessed: June 2017 (page 16).

[121] John F Hernandez Nopsa, Gregory J Daglish, David W Hagstrum, John F Leslie, Thomas W Phillips, Caterina Scoglio, Sara Thomas-Sharma, Gimme H Walter, and Karen A Garrett. "Ecological networks in stored grain: Key postharvest nodes for emerging pests, pathogens, and mycotoxins." In: *BioScience* (2015), biv122 (page 85).

[122] NPPO. Tuta absoluta *Povolny (Gelechiidae) - tomato leaf miner - in tomato packaging facility in The Netherlands, National Plant Protection Organization (NPPO), Wageningen*. 2009 (page 86).

[123] Jeremy E Oakley and Anthony O'Hagan. "Probabilistic sensitivity analysis of complex models: a Bayesian approach." In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 66.3 (2004), pp. 751–769 (page 38).

[124] H. Owhadi, C. Scovel, T. J. Sullivan, M. McKerns, and M. Ortiz. "Optimal Uncertainty Quantification." In: *SIAM Review* 55.2 (2013), pp. 271–345 (page 13).

[125] Sevcan Oztemiz. "*Tuta absoluta* povolny (Lepidoptera: Gelechiidae), the exotic pest in Turkey." In: *Romanian Journal of Biology* (2014) (page 86).

[126] Nidhi Parikh, Samarth Swarup, Paula E Stretz, Caitlin M Rivers, Bryan L Lewis, Madhav V Marathe, Stephen G Eubank, Christopher L Barrett, Kristian Lum, and Youngyun Chungbaek. "Modeling human behavior in the aftermath of a hypothetical improvised nuclear detonation." In: *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems. 2013, pp. 949–956 (page 7).

[127] Matthew Parno, Andrew Davis, and Patrick Conrad. *MIT Uncertainty Quantification (MUQ) library, 2014* (page 17).

[128] Romualdo Pastor-Satorras, Claudio Castellano, Piet Van Mieghem, and Alessandro Vespignani. "Epidemic processes in complex networks." In: *Reviews of modern physics* 87.3 (2015), p. 925 (pages 85, 88).

[129] Judea Pearl. *Reverend Bayes on inference engines: A distributed hierarchical approach*. Cognitive Systems Laboratory, School of Engineering and Applied Science, University of California, Los Angeles, 1982 (page 88).

[130] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. "Scikit-learn: Machine learning in Python." In: *The Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (page 17).

[131] L Felipe Perrone, Christopher S Main, and Bryan C Ward. "SAFE: simulation automation framework for experiments." In: *Proceedings of the Winter Simulation Conference*. Winter Simulation Conference. 2012, p. 249 (page 17).

[132] Robin L Plackett and J Peter Burman. "The design of optimum multifactorial experiments." In: *Biometrika* 33.4 (1946), pp. 305–325 (page 36).

[133] *pyDOE*. *Available at* `https://pythonhosted.org/pyDOE`. Last accessed: June 2017 (pages 17, 36).

[134] Carl Edward Rasmussen. "Gaussian processes for machine learning." In: (2006) (pages 37, 38).

[135] Andre S Ribeiro and Stuart A Kauffman. "Noisy attractors and ergodic sets in models of gene regulatory networks." In: *Journal of theoretical biology* 247.4 (2007), pp. 743–755 (pages 2, 15, 58).

[136] Caitlin Rivers. "Ebola: models do more than forecast." In: *Nature* 515.7528 (2014), pp. 492–492 (page 7).

[137] Christopher J Roy and William L Oberkampf. "A comprehensive framework for verification, validation, and uncertainty quantification in scientific computing." In: *Computer Methods in Applied Mechanics and Engineering* 200.25 (2011), pp. 2131–2144 (page 13).

[138] Andrea Saltelli, Karen Chan, E Marian Scott, et al. *Sensitivity analysis*. Vol. 1. Wiley New York, 2000 (pages 2, 14).

[139] Andrea Saltelli, Stefano Tarantola, and KP-S Chan. "A quantitative model-independent method for global sensitivity analysis of model output." In: *Technometrics* 41.1 (1999), pp. 39–56 (page 38).

[140] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008 (page 14).

[141] Thomas J Santner, Brian J Williams, and William Notz. *The design and analysis of computer experiments*. Springer, 2003 (pages 2, 14, 36, 38).

[142] Robert G Sargent. "Verification and validation of simulation models." In: *Proceedings of the 37th conference on Winter simulation*. winter simulation conference. 2005, pp. 130–143 (page 33).

[143] Devavrat Shah and Tauhid Zaman. "Rumors in a network: Who's the culprit?" In: *IEEE Transactions on information theory* 57.8 (2011), pp. 5163–5181 (page 89).

[144] I. Shmulevich, E. R. Dougherty, and W. Zhang. "From Boolean to probabilistic Boolean networks as models of genetic regulatory networks." In: *Proceedings of the IEEE* 90.11 (2002), pp. 1778–1792 (page 45).

[145] I. Shmulevich and S. A. Kauffman. "Activities and Sensitivities in Boolean Network Models." In: *Physical Review Letters* 93.4 (2004), 048701:1–4 (pages 11, 57, 58, 60, 62, 67, 72).

[146] I. Shmulevich, E. R. Dougherty, S. Kim, and W. Zhang. "Probabilistic Boolean Networks: A Rule-based Uncertainty Model for Gene Regulatory Networks." In: *Bioinformatics* 18.2 (2002), pp. 261–274 (page 45).

[147] Ilya Shmulevich, Edward R Dougherty, and Wei Zhang. "From Boolean to probabilistic Boolean networks as models of genetic regulatory networks." In: *Proceedings of the IEEE* 90.11 (2002), pp. 1778–1792 (pages 15, 16).

[148] Ilya Shmulevich and Stuart A Kauffman. "Activities and sensitivities in Boolean network models." In: *Physical review letters* 93.4 (2004), p. 048701 (pages 2, 16).

[149] Ilya Shmulevich, Edward R Dougherty, Seungchan Kim, and Wei Zhang. "Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks." In: *Bioinformatics* 18.2 (2002), pp. 261–274 (page 16).

[150] T. Smith, N. Maire, A. Ross, M. Penny, N. Chitnis, A. Schapira, A. Studer, B. Genton, C. Lengeler, F. Tediosi, D. de Savigny, and M. Tanner. "Towards a comprehensive simulation model of malaria epidemiology and control." In: *Parasitology* 135.13 (2008), pp. 1507–1516 (page 48).

[151] Alex J Smola and Bernhard Schölkopf. "A tutorial on support vector regression." In: *Statistics and computing* 14.3 (2004), pp. 199–222 (page 38).

[152] Ilya M Sobol. "Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates." In: *Mathematics and computers in simulation* 55.1 (2001), pp. 271–280 (pages 38, 80).

[153] Il'ya Meerovich Sobol'. "On sensitivity estimation for nonlinear mathematical models." In: *Matematicheskoe Modelirovanie* 2.1 (1990), pp. 112–118 (page 14).

[154] Richard Soley et al. "Model driven architecture." In: *OMG white paper* 308.308 (2000), p. 5 (page 27).

[155] Robert W Sutherst. "Climate change and invasive species: a conceptual framework." In: *Invasive species in a changing world* (2000), pp. 211–240 (page 85).

[156] K. Sutner. "On the computational complexity of finite cellular automata." In: *Journal of Computer and System Sciences* 50.1 (1995), pp. 87–97 (page 16).

[157] Sweta Sutrave, Caterina Scoglio, Scott A Isard, JM Shawn Hutchinson, and Karen A Garrett. "Identifying highly connected counties compensates for resource limitations when evaluating national spread of an invasive pathogen." In: *PLoS One* 7.6 (2012), e37793 (page 85).

[158] Boxin Tang. "Orthogonal array-based Latin hypercubes." In: *Journal of the American Statistical Association* 88.424 (1993), pp. 1392–1397 (pages 14, 36).

[159] Claudia Taylor, Achla Marathe, and Richard Beckman. "Same influenza vaccination strategies but different outcomes across US cities?" In: *International Journal of Infectious Diseases* 14.9 (2010), e792–e795 (pages 28, 81).

[160] Charles Tong. "PSUADE userâĂŹs manual." In: *Lawrence Livermore National Laboratory (LLNL), Livermore, CA* 109 (2005) (pages 6, 9, 16, 17).

[161] Timothy G Trucano, Laura Painton Swiler, Takera Igusa, William L Oberkampf, and Martin Pilch. "Calibration, validation, and sensitivity analysis: What's what." In: *Reliability Engineering & System Safety* 91.10 (2006), pp. 1331–1357 (page 15).

[162] John von Neumann. *Theory of Self-Reproducing Automata*. Edited and completed by Arthur W. Burks. University of Illinois Press, 1966 (page 45).

[163] Chen Wang, Qingyun Duan, Charles H Tong, Zhenhua Di, and Wei Gong. "A GUI platform for uncertainty quantification of complex dynamical models." In: *Environmental Modelling & Software* 76 (2016), pp. 1–12 (pages 6, 9, 17).

[164] Brian J Williams, Thomas J Santner, and William I Notz. "Sequential design of computer experiments to minimize integrated response functions." In: *Statistica Sinica* (2000), pp. 1133–1152 (page 14).

[165] Stephen Wolfram et al. *Theory and applications of cellular automata*. Vol. 1. World scientific Singapore, 1986 (page 15).

[166] Sichao Wu, Abhijin Adiga, and Henning S Mortveit. "Limit cycle structure for dynamic bi-threshold systems." In: *Theoretical Computer Science* 559 (2014), pp. 34–41 (pages 46, 48, 51, 52).

[167] Jae-Seung Yeom, Abhinav Bhatele, Keith Bisset, Eric Bohm, Abhishek Gupta, Laxmikant V Kale, Madhav Marathe, Dimitrios S Nikolopoulos, Martin Schulz, and Lukasz Wesolowski. "Overcoming the scalability challenges of epidemic simulations on blue waters." In: *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*. IEEE. 2014, pp. 755–764 (pages 2, 15).