

Student Work

12-1-2006

Dynamics of Random Boolean Networks Governed by a Generalization of Rule 22 Of Elementary Cellular Automata.

Gary L. Beck

Follow this and additional works at: <https://digitalcommons.unomaha.edu/studentwork>
Please take our feedback survey at: https://unomaha.az1.qualtrics.com/jfe/form/SV_8cchtFmpDyGfBLE

Recommended Citation

Beck, Gary L., "Dynamics of Random Boolean Networks Governed by a Generalization of Rule 22 Of Elementary Cellular Automata." (2006). *Student Work*. 3549.
<https://digitalcommons.unomaha.edu/studentwork/3549>

This Thesis is brought to you for free and open access by DigitalCommons@UNO. It has been accepted for inclusion in Student Work by an authorized administrator of DigitalCommons@UNO. For more information, please contact unodigitalcommons@unomaha.edu.

Dynamics of Random Boolean Networks Governed by a Generalization of Rule 22 Of Elementary Cellular Automata

A Thesis

presented to the

Department of Mathematics

and the

Faculty of the Graduate College

in partial fulfillment

of the requirements for the degree

Master of Arts

University of Nebraska at Omaha

By

Gary L. Beck

December 2006

UMI Number: EP74747

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI EP74747

Published by ProQuest LLC (2015). Copyright in the Dissertation held by the Author.

Microform Edition © ProQuest LLC.

All rights reserved. This work is protected against unauthorized copying under Title 17, United States Code



ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

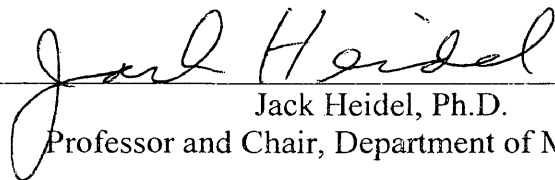
Thesis Acceptance

Acceptance for the faculty of the Graduate College,
University of Nebraska, in partial fulfillment of the
Requirements for the degree Master of Arts in Mathematics,
University of Nebraska at Omaha

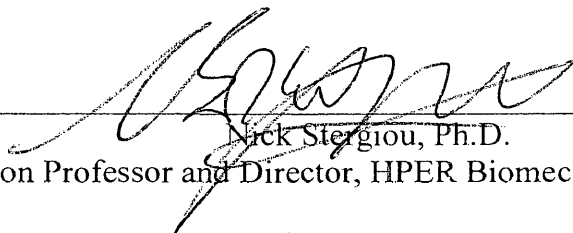
Committee



Dora Matache, Ph.D.
Graduate Thesis Advisor
Assistant Professor, Department of Mathematics



Jack Heidel, Ph.D.
Professor and Chair, Department of Mathematics



Nick Stergiou, Ph.D.
Isaacson Professor and Director, HPER Biomechanics Laboratory

Graduate Committee Chair:



Steve From, Ph.D.
Professor, Department of Mathematics

Date:

November 21, 2006

Table of Contents

Table of Contents

Abstract

Acknowledgements

1	Introduction	1
1.1	Cellular Automata	2
1.2	Elementary Cellular Automata	3
1.2.1	Elementary Cellular Automata Classes	4
1.3	Random Boolean Networks	5
1.4	Transition Functions	6
1.5	Overview of the Relevant Literature and Motivation for the Current Work	9
1.5.1	Relevant Work in the Literature	9
1.5.2	Current Work Inspiring this Research	13
2	The Boolean Model	17
2.1	Iterations of the System and the Model	25
2.1.1	Fixed Connectivity	26
2.1.2	Variable Connectivity	30
3	Analysis of the Dynamics	36
3.1	Statistical Analysis	36
3.1.1	Fixed Connectivity	37
3.1.2	Variable Connectivity	39
3.2	Bifurcation Diagrams and Lyapunov Exponents	45
3.3	Fixed Points and Delay Plots	49
3.3.1	Focus on Fixed Points	49

3.3.2	Delay Plots and Attractors	53
4	Conclusions	59
5	Directions for Further Investigation	61
	Bibliography	64
	Appendices	72
A	Fixed Connectivity Matlab Code	73
B	Variable Connectivity Matlab Code	78
C	Additional Codes for Model Simulations	84
C.1	Randomly Assigned Parents	84
C.2	k Nearest Neighbors Parents	86
C.3	Probability Model Code	86
C.4	Matlab Code for Randomly Selecting a Single Rule	88
D	Lyapunov Exponent and Bifurcation Diagram Matlab Code	90
D.1	Lyapunov Exponents and Bifurcation Diagrams	90
D.2	Derivative Function Code for Lyapunov Exponents	96
E	Fixed Points	99
F	Delay Plots	103

Abstract

**Dynamics of Random Boolean Networks Governed by
a Generalization of Rule 22 of Elementary Cellular Automata**

Gary L. Beck, M.A.

University of Nebraska, 2006

Dora Matache, Ph.D., Thesis Advisor

This study considers a simple Boolean network with N nodes, each node's state at time t being determined by a certain number of k parent nodes. The network is analyzed when the connectivity k is fixed or variable. This is an extension of a model studied by Andrecut [4] who considered the dynamics of two stochastic coupled maps. Making use of the Boolean rule that is a generalization of Rule 22 of elementary cellular automata, a generalization of the formula for providing the probability of finding a node in state 1 at a time t is determined and used to generate consecutive states of the network for both the real system and the model. We show that the model is a good match for the real system for some parameter combinations. However, our results indicate that in some cases the model may not be a good fit,

thus contradicting results of [4]. For valid parameter combinations, we also study the dynamics of the network through Lyapunov exponents, bifurcation diagrams, fixed point analysis and delay plots. We conclude that for fixed connectivity the model is a more reliable match for the real system than in the case of variable connectivity, and that the system may exhibit stability or chaos depending on the underlying parameters. In general high connectivity is associated with a convergence to zero of the probability of finding a node in state 1 at time t .

Acknowledgements

I would like to thank Dr. Dora Matache for being such an incredible educator. Her mentorship during this process has been appreciated more than I could ever express here.

Dr. Fredrick McCurdy, my friend and inspiration, is an example of the educator's educator. His thirst for knowledge truly inspired me to pursue this trek.

I would also like to thank Dr. Steve From for being a great sounding board for me during my years in graduate school. As a non-traditional student, it was helpful to have him as an advisor to keep me going even when I was ready to give up.

I would also like to thank Drs. Steve From, Jack Heidel and Nick Stergiou for serving on my thesis committee, giving up a great deal of time to review this work and providing feedback to make this a very complete product.

Finally, my partner, Will Dallaghan, deserves the biggest thanks of all for having to deal with me since I began this journey in 1999.

Chapter 1

Introduction

As advances in technology aid in unraveling the complexities of life, more studies are being conducted pertaining to questions surrounding genetics, evolution, biological processes, and computational systems. The premise that a system can be analyzed based on its behavior, not on isolated situations, provides the theoretical construct behind the study of the dynamics of complex networks. Identifying and understanding the dynamics of how these networks interact lead to developing mathematical models to reflect reality. The complexities of some of these systems present difficulties in developing optimal solutions.

This section introduces the general concepts behind elementary cellular automata and the various rules governing them. We will also describe how random Boolean networks are used to model certain systems. This information will help understand the process by which our work was derived.

1.1 Cellular Automata

Computers not only enable mathematicians and scientists to solve some of these complex problems of network dynamics, they also raise additional questions, such as the concept of artificial self-reproduction [3]. John von Neumann is credited as being the first to demonstrate automata, or cells, capable of self-replication given the proper logic and initial conditions [48]. Cellular automata also provide simple models for a variety of physical, chemical and biological systems.

Cellular automata consist of a network of elements existing in some geometric lattice structure. These structures may be in one or multiple dimensions. They are mathematical representations of physical systems with a discrete number of states per cell. Cellular automata operate through application of transition functions, or updating rules, associated with each cell of the automata. As a result, the network exhibits evolution as these transition functions are applied simultaneously. The goal of the study of these cellular automata is to elucidate general features of behavior and perhaps devise generalized rules of behavior.

As Stephen Wolfram notes, any system with multiple identical discrete elements undergoing deterministic local interactions may also be modeled as cellular automaton [51]. Examples of natural phenomenon that may be modeled as cellular automata include snowflake growth, biological functions of simple organism transformations into complicated forms, and some mathematical systems found in number theory.

1.2 Elementary Cellular Automata

The simplest class of cellular automata may be further explained using one-dimensional lattices with two states per cell. This class also takes into consideration the nearest neighbor rule for the dynamics of the system and closed networks with periodic boundary conditions. This class is known as elementary cellular automata (ECA). One-dimensional cellular automata consists of a line of cells, each cell having a neighbor to the left and to the right. The cells exist in two states. In binary terms, the values of these cells is either 0 or 1, which is equivalent to Boolean values of FALSE or TRUE, respectively. For the figures of these network strings, the 0 and 1 states are represented by white and black cells, respectively (See Figure 1.1).

As a one-dimensional network evolves, the state can be visualized in two dimensions. As successive application of the logic guiding the transition, known as transition functions, additional rows are added for each complete mapping. Because the evolution of the system is actually recursive mapping of the transition function, each row represents a time step; e.g., the system at time step t_1 evolves to the system state at time step t_2 [3].



Figure 1.1: Elementary cellular automata input structure for a possible transition function. White indicates the cell is OFF; black cells are ON.

Such recursive mapping results in dynamical system that may be qualitatively studied. The most basic qualitative features of dynamical systems are the attractors that are found over a period of time, whether it be over a discrete period of time or to infinity. The initial conditions that evolve to a given attractor are referred to as basins of attraction; the understanding of the numbers of attractors and how the basins partition the state space is essential for comprehending the function of the system under investigation [6].

1.2.1 Elementary Cellular Automata Classes

Based on the extensive work Stephen Wolfram has conducted with ECA, standard classifications based on the behaviors of the rule sets are used to group the various ECA rules [50]. It is possible for elementary cellular automata to produce all types of behaviors in any given system. Initial conditions required to produce these behaviors need not be complicated. Wolfram's four classes of behavior are - I, II, III, and IV.

For Class I, system states rapidly evolve to a steady state. Rules in this class yield trivial behavior because their evolution is very predictable. Class II demonstrates simple behavior. This class of systems evolve into non-uniform states or evolve by alternating among a limited number of states periodically. Since cellular automata are deterministic systems, it is almost certain that any cellular automata evolution

will eventually become periodic. Being sensitive to initial conditions and slight perturbations is a hallmark of Class III because there appear to be no attractors and the states are random [29]. In the case of Rule 30, it is inherently chaotic with the exception of all but a trivial set of initial conditions. Class IV is a mix of Class II and III, demonstrating simple and random behaviors, comparable to a transition from solid to liquid. It is a complex transition from order to chaos.

1.3 Random Boolean Networks

For a more general system than cellular automata, a random Boolean network can function as a model for a large class of biological networks [53], neural networks [1] and genetic systems ([45], [46]), as well as chemical processes [23]. Boolean network models have been used extensively in modeling networks in which the node activity can be described by two states, ON and OFF, "active and nonactive", "responsive and non-responsive", "up-regulated and down-regulated", and in which each node is updated based on logical relationships with other nodes, called parents. In fact, Stuart Kauffman originally developed random Boolean networks as a model for genetic regulatory networks [26]. They have been referred to as $N - K$ models or Kauffman networks.

The dynamics of random Boolean networks may be ordered, chaotic, or complex [26] depending on values of nodes (N) and connections (k). Diverse properties of ECA

and random Boolean networks have been used to model phenomena in physics [41], biology ([21], [27]), cognition [30], artificial life [50], artificial neural networks [28], and music [9]. In general, random Boolean networks are good models for complex systems ([14], [53]).

The similarity between random Boolean networks and cellular automata consists of a set of nodes operated on by other nodes (called parents). Cellular automata consists of an infinite number of nodes in a spatial structure ([3], [52]); Boolean networks are finite but not arranged in any spatial manner ([23], [26], [33], [34]). Network connections may be nonlocal wherein the input and logic can be different for each network element. These elements change their values from input according to given Boolean rules. Therefore, the possible alternative network constructs can take into account all possible connection and rule schemes, making random Boolean networks generic. Therefore, if the specific structure or function of a system is complex or unknown, general properties found in a particular random Boolean network model may be applied to understand its processes [14].

1.4 Transition Functions

The operation of the cellular automata and Boolean networks is controlled by conditional logic that maps a neighborhood of cells into a single state, known as transition functions. When cellular automata networks have complete connectivity, cells evolve

based on the states of all the cells in the network. However, elementary cellular automata have a neighborhood with radius of one cell, $r = 1$. Given this type of connectivity, a cell will transition to a new state at time $t + 1$ based on its state and those of its nearest neighbor at time t .

For automata with k possible states and a neighborhood of r cells, there will be k^{2r+1} combinations of cell states which form the transition function input. Consider elementary cellular automata with two possible states, 0 or 1, with a neighborhood of three cells. The number of combinations of cell states is $2^3 = 8$ (See Figure 1.1).

Because each of these neighborhoods can evolve to one of two states, there are $k^{k^{2r+1}}$ possible sets of transition functions. For elementary cellular automata, this means there are $2^{2^3} = 256$ possible sets of rules. Specifically, for each combination of cell sequences, a decimal value equivalent is assigned [3]. Each of these decimal equivalents represents a single bit in an eight-bit byte. With the values of cells equalling 0 or 1, evolution of the cell states can be simplified into a rule number that defines the logic of the transition function. Figure 1.2 demonstrates the coding process Anderson defines for Rule 22. Transition functions impact the evolution of

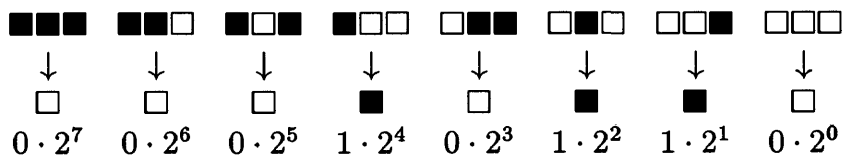


Figure 1.2: Demonstration of a transition function and corresponding bit value. The decimal representation of this binary value is 22.

cellular automata by the sum of the weights of the neighborhood. A totalistic rule for a $(k, 1)$ automaton would take the form $f(a, b, c) = r(a + b + c) \bmod k$, where r is the function of integers modulo k [37]. For ECA under a totalistic rule, the next state for the element depends only on the sum of total number of bits from its assigned neighborhood at time $t : s(t)$. The neighborhood has $k_j + 1$ elements, the current element, and k_j parents. Therefore, s takes values from 0 to $k_j + 1$ and the totalistic rule is described by a Boolean function

$$r_{k_j}(s) : \{0, 1, \dots, k_j + 1\} \rightarrow \{0, 1\}$$

According to Wolfram [50], transition functions impact the evolution of cellular automata by a proportionate number of elements in the neighborhood.

This understanding of ECA evolution provides the basic foundation for how cellular automata may be used to model real networks. With the aid of computers, it is much easier to conduct research in this field. What follows is a discussion of research that has been published as well as work that is ongoing to provide a better understanding of the application of developments in ECA and random Boolean networks.

1.5 Overview of the Relevant Literature and Motivation for the Current Work

As was mentioned in the Introduction, the investigation of dynamical systems and their application in various systems is being extensively studied. Dynamical networks in biology are found everywhere from the brain to ecology [53]. The mapping of the human genome sparked even more interest in the function and behavior of genetic regulatory networks ([11], [21], [45], [46]), neural networks [1], artificial neural networks [28], chemical [23], organizational [47], and physical networks ([31], [32]). In fact, scientific discoveries generate new questions about the accuracy of mathematical models and the function of the processes they model. Therefore, the need for theoretical approximations and concepts to understand the avalanche of data is urgent.

1.5.1 Relevant Work in the Literature

In the study of Boolean networks, much work has been published on the subject. From the initial work done by Kauffman [25], use of random Boolean networks to model network dynamics has been studied to great extent. The following discussion details some of the current applications in a variety of fields. This is a select review of the literature to demonstrate what has been researched to date.

For example, in the field of neurobiology and psychiatry, the study of measures made on patterns of chaotic motion on topological manifolds has provided an objective means of researching topics such as personality and character. Using techniques derived from the analysis of differential equations and their maps in phase space, elusive personality behaviors such as obsessive-compulsive disorders have been studied. In one study [30], use of computer testing demonstrated a divergence of Lyapunov measures which differentiated the study participants into obsessive-compulsive and borderline character disorders. The model was meant to demonstrate the mechanisms of cause and effect. Contrary to the memory capacity of the brain, expanding dynamical systems lose memory and settle into basins of attraction. Therefore, attempts at using this type of research as a predictor of personality behaviors may be premature. However, further research is being done in the field by Mandell and others.

Pure simulations are not representative of real systems. The presence of "noise" in a system may result in changes in systemic behaviors. Multicellular structures require extreme error tolerance to evolve over time. Studying the topology and robustness in biological and genetic regulatory networks has demonstrated that effects of fluctuations on the system are largely dependent on reliability measures ([7], [19], [21], [27], [42]). Incorporating redundancy into the role of robustness [17] aids in slowing the evolution of dynamical systems, which is a more accurate reflection of genetic regulatory networks. In this approach a model was developed to generate a

copy of certain elements in the system. The redundancy of links was investigated and shown to have no affect on network stability. However, introducing redundant nodes allowed the model to evolve gradually over time, providing a "smoothness" to the map.

Because of their convenient and easily understood structure, Boolean networks are used extensively to model complex networks. These have been studied as discrete time and continuous time networks. Use of discrete time random Boolean networks to model genomic networks is important, but also has limitations due to convergence to basins of attraction. Introduction of continuous time delays for system updates is necessary for more accurate models [40], which is a network of state variables wherein the relationships among state variables are dictated by Boolean relations with delays. However, as with the initial work by Kauffman [25] on genetic regulatory networks, synchronous updates have been shown to be unrealistic and thus asynchronous updating schemes continue to be studied to more accurately reflect reality.

Along with genetic and biological regulatory networks, circadian and other biological rhythms are important components of intracellular functions. Because random Boolean networks have been associated with cyclical behavior they have been associated with molecular clocks. In one study [43], random asynchronous updates of the network are implemented by updating a node at random using a uniform probability

with replacement and repeating the operation over time. In this study, the asynchronous random Boolean networks were capable of producing rhythm and found a circular functional structure producing travelling waves. Even in the face of perturbations and errors, the system fell back to the same attractor as long as the functional topology held. This work is now being explored in the neuronal dynamics related to changes in mental processes [41].

Because biological, social, chemical and other physical networks are so complex, understanding these complex systems is crucial. Although random Boolean networks are good models to use, there is work being done with cellular automata to further define complexity. One study by Sanchez and Lopez-Ruiz [44] compared two one-dimensional cellular automata evolving under Rules 22, 30, 90 and 110 by plotting two rules and the difference of each plot, which exhibit a synchronization transition zone. This process allowed the authors to identify a complexity measurement, which for these rules, indicated the highest complexity is reached on the borders of the synchronization region at a consistent probability.

All of these studies have assumed a particular ECA rule or random Boolean network model to examine system dynamics. In generating a time series of binary strings, Voorhees used an induction algorithm to make unbiased best guess estimates of the cellular automata rules. For induction in this study, the emphasis was to extract

order from a noisy data stream by focusing on the randomness of the data. The general conclusions indicated that it may not be possible to identify particular cellular automata rules in a noisy deterministic process [49].

These and many other studies have specifically targeted random Boolean networks or ECA rules to garner understanding of real networks. What we reported is a fraction of the work that has been published. The next section details specific work in dynamics of ECA and random Boolean networks.

1.5.2 Current Work Inspiring this Research

The previously mentioned work that has been investigated with cellular automata and random Boolean networks is a small sample of what has been done and so much more needs to be done. Utilizing classes of cellular automata outlined by Wolfram [50], various authors have developed Boolean models for real networks. For example, Andrecut and Ali [5] introduced a mean field model for the case of a synchronous Boolean network with fixed number of parents governed by a generalization of the ECA Rule 126. The mean field approach assumes that the parent nodes act independently and that the network has a relatively large number of nodes.

It is known that certain elementary cellular automata rules can generate complex patterns in the evolution of cellular automata [50]. For example Rules 22 and 126 are Class III elementary cellular automata rules in Wolfram's characterization. As

observed by Matache and Heidel [34], both Rule 126 and Rule 22 have a natural and simple interpretation in terms of the growth of cell colonies. For Rule 126, complete crowding of live, ON, cells causes death, OFF, in the next generation. Complete isolation of a potential cell prevents birth in the next generation. A similar interpretation holds for Rule 22; however, it is not quite as absolute. We attach a graph of pattern formation plots for both ECA Rules 126 and 22, to identify the differences in the long term behavior (Figure 1.3). We start with a network of 200 nodes in which only one node is ON (black) at time $t = 1$ and in which the evolution from one time step to another is given by one of the two rules. The time evolves downward and we iterate the network 400 time steps. We can see the similarities as well as the differences in the graph below. Therefore, it is of interest to focus attention to generalizations of these two rules to networks that have a fixed but arbitrary number of nodes. Rule 126 has been extensively studied in ([4], [12], [34], [35], [36]) for networks with fixed or variable number of parents, synchronous or asynchronous. However, an extension of Rule 22 has not been investigated in this context so far. In this study we propose a generalized ECA Rule 22 and identify the dynamics of a network governed by variants of this generalization for some of the admissible parameter combinations.

Continuing his previous work, Andrecut [4] recently published a paper investigating mean field approximations for Boolean networks governed by the class of totalistic

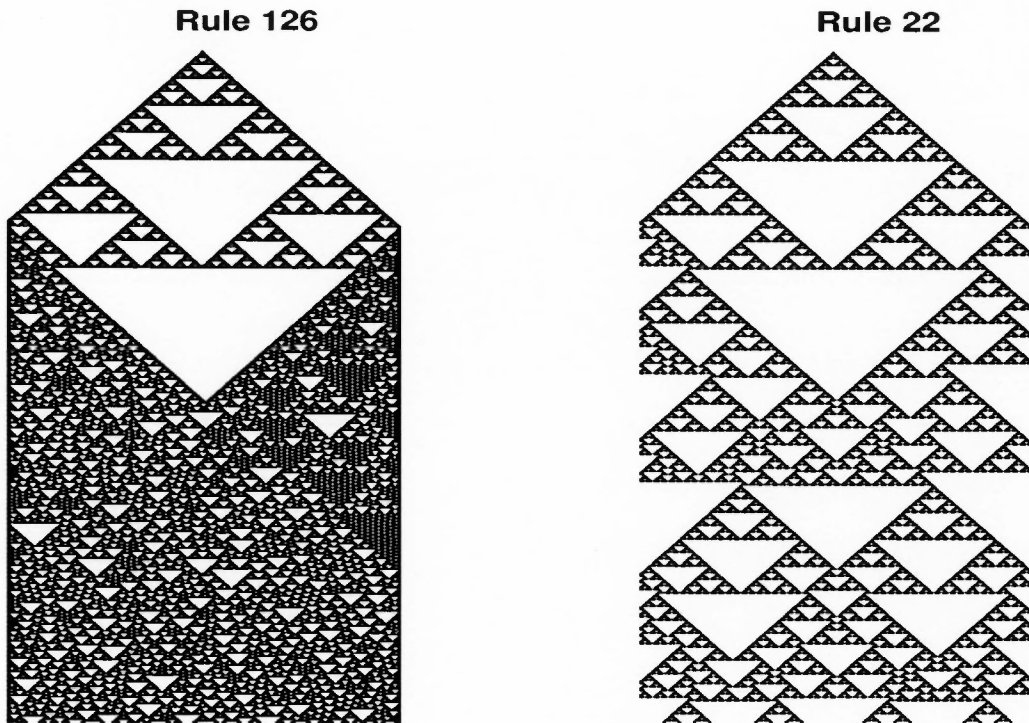


Figure 1.3: Pattern formation plots for the ECA Rules 126 and 22. The time evolves downward and nodes that are ON are plotted with black. In each graph the network has 200 nodes and is iterated 400 time steps.

ECA rules. This work also demonstrated that the dynamics of two stochastic coupled random Boolean networks can be described by two deterministic coupled maps. In Andrecut's study, excellent agreement of the model and the real system has been demonstrated. This was done by generating iterations of the model and the system that agree almost perfectly. Unfortunately only initial iterations have been investigated in [4]. To study the dynamics of such networks using a mathematical model, one needs to consider the long-run behavior. Therefore, making sure that higher order iterates of the model match the corresponding iterations of the real system is

mandatory.

In this study we will show that from this perspective the model presented in [4] is not generally applicable for synchronous updates. We will indicate certain Boolean rules and parameter values that will generate a mismatch of higher order iterates despite the perfect match of the first iteration, thus compromising the possibility of using the claimed mathematical model as a tool for a study of the dynamics of the corresponding real network. At the same time we will indicate the ranges of parameters for which the model is suitable for modeling the real system and perform a statistical analysis to understand what the significant parameters are that generate the mismatches. We will also note that even though the model is generated under the assumption of random neighborhoods for each node, it can still approximate alternate cellular automata under certain circumstances; that is when the neighborhoods of the nodes are chosen locally. For the cases when the model and the reality indicate a good match we will analyze the dynamics of the system using tools from the theory of dynamical systems and chaos. We will explore bifurcation diagrams, Lyapunov exponents, fixed points, and delay plots.

Before the analyses can be addressed, we will first describe the model for ECA Rule 22. In the next section we will fully explain the model development and provide a proof of our equation.

Chapter 2

The Boolean Model

Consider a network with N nodes. Each node c_n , $n = 1, 2, \dots, N$ can take on only two values 0 or 1. At each time point t the system can be in one of 2^N possible states. The connectivity of the nodes is fixed throughout the evolution of the system, but the nodes are allowed to have different numbers of parent nodes. Thus the nodes can be grouped into classes of size M_j containing all the nodes with a fixed connectivity k_j , $j = 1, 2, \dots, J$. It follows that $\sum_{j=1}^J M_j = N$. The parents of a node are chosen randomly from the remaining $N - 1$ nodes and do not change thereafter. More precisely, if a node has k parents, then a set of k nodes is chosen from the remaining $N - 1$ nodes with probability $\frac{1}{\binom{N-1}{k}}$. The nodes can be updated synchronously or asynchronously. At each time point t , a proportion $\alpha(t) \in [0, 1]$ of nodes is updated. The case $\alpha(t) = 1$ corresponds to a synchronous network. This will be our main focus in this study. Future work will focus on asynchronous networks.

We assume that all the nodes in each class of nodes with a fixed number of parents

k_j update according to the same randomly chosen rule at a given time point t . This implies that all the nodes in a certain class have the same collection of possible predictors.

A totalistic cellular automata rule depends only on the number of 1s in the neighborhood of a node. Given a node c_n with k_j parents, a totalistic rule for that node can be expressed as a pair of Boolean functions

$$(r_0^{k_j}, r_1^{k_j}) : \{0, 1, 2, \dots, k_j\}^2 \rightarrow \{0, 1\}$$

such that

$$c_n(t+1) = \begin{cases} r_0^{k_j}(s) & \text{if } c_n(t) = 0 \\ r_1^{k_j}(s) & \text{if } c_n(t) = 1 \end{cases}$$

where s is the sum of the values of the parents of node c_n at time t .

For example, ECA rule 22 specifies that a node becomes 1 at time $t+1$ if and only if the sum of the values of the three nodes at time t is equal to 1. That is exactly one node is 1 at time t and the others are 0. This translates into the following values for the functions $(r_0^{k_j}, r_1^{k_j})$:

$$(r_0^2, r_1^2) : \{0, 1, 2\}^2 \rightarrow \{0, 1\}$$

with

$$\begin{cases} r_0^2(0) = 0, & r_0^2(1) = 1, & r_0^2(2) = 0 \\ r_1^2(0) = 1, & r_1^2(1) = 0, & r_1^2(2) = 0 \end{cases}$$

Observe that this is the same as saying that if $s + c_n(t) = 1$ then $c_n(t+1) = 1$, otherwise it is 0.

Similarly, ECA rule 126 specifies that a node becomes 0 at time $t + 1$ if and only if the sum of the values of the three nodes at time t is either 0 or 3. This translates into the following values for the functions $(r_0^{k_j}, r_1^{k_j})$:

$$(r_0^2, r_1^2) : \{0, 1, 2\}^2 \rightarrow \{0, 1\}$$

with

$$\begin{cases} r_0^2(0) = 0, & r_0^2(1) = 1, & r_0^2(2) = 1 \\ r_1^2(0) = 1, & r_1^2(1) = 1, & r_1^2(2) = 0 \end{cases}$$

Observe that this is the same as saying that if $s + c_n(t) = 0$ or 3 then $c_n(t + 1) = 0$, otherwise it is 1.

A general model for the probability $p(t + 1)$ of a node being in state 1 at time $t + 1$, given $p(t)$ has been obtained by Andrecut [4]. This model is applicable to the class of legalistic rules and in a more general setting for the class of so-called probabilistic Boolean networks in which a node can be updated according to more than one Boolean rule. The model can be expressed as follows.

$$(1) \quad p(t + 1) = \sum_{j=1}^J c_j g_{k_j}(t)$$

where

$$g_{k_j}(t) = [1 - \alpha(t)]p(t) + \alpha(t) \sum_{s=0}^{k_j} \gamma(p(t), r_0^{k_j}(s), r_1^{k_j}(s)) P_{s, k_j}(p(t)),$$

$$\gamma(p(t), r_0^{k_j}(s), r_1^{k_j}(s)) = (1 - p(t))r_0^{k_j}(s) + p(t)r_1^{k_j}(s),$$

and

$$P_{s, k_j}(p(t)) = \binom{k_j}{s} p(t)^s (1 - p(t))^{k_j - s}$$

Here $k_j, j = 1, 2, \dots, J$ are the possible values for the number of parents, $c_j, j = 1, 2, \dots, J$ are the proportions of nodes with k_j parents (and therefore $\sum_{j=1}^J c_j = 1$), and $\alpha(t)$ is the proportion of nodes to be updated at time t .

The sum s will be used in this work to express the sum of the values of nodes in the neighborhood of a selected node c_n .

The model above is obtained as follows. Consider the number of nodes with connectivity k_j that change their state using the totalistic rule $r_{k_j}(s)$, then we have

$$(2) \quad N_{0 \rightarrow 1}^{k_j}(t) = c_j N_0(t) [1 - p(t)] \sum_{s=0}^{k_j} r_0^{k_j}(s) P_{s, k_j}(p(t))$$

$$(3) \quad N_{1 \rightarrow 0}^{k_j}(t) = c_j N_1(t) p(t) [1 - \sum_{s=0}^{k_j} r_1^{k_j}(s+1) P_{s, k_j}(p(t))]$$

$$(4) \quad N_{0 \rightarrow 0}^{k_j}(t) = c_j N_0(t) [1 - p(t)] [1 - \sum_{s=0}^{k_j} r_0^{k_j}(s) P_{s, k_j}(p(t))]$$

$$(5) \quad N_{1 \rightarrow 1}^{k_j}(t) = c_j N_1(t) p(t) \sum_{s=0}^{k_j} r_1^{k_j}(s+1) P_{s, k_j}(p(t))$$

Thus, $N_{0 \rightarrow 1}^{k_j}$ is the number of nodes with connectivity k_j changing their state from 0 to 1 at time t . Similarly $N_{1 \rightarrow 0}^{k_j}$ is the number of nodes changing their state from 1 to 0. The same principle applies for $N_{0 \rightarrow 0}^{k_j}$ and $N_{1 \rightarrow 1}^{k_j}$. We assume the parent nodes act independently. To confirm this, it is important to check that these quantities equal N . Observe that $N_{0 \rightarrow 0}^{k_j} + N_{0 \rightarrow 1}^{k_j} = N_0^{k_j}(t), j = 1, 2, \dots, J$ and $N_{1 \rightarrow 0}^{k_j} + N_{1 \rightarrow 1}^{k_j} =$

$N_1^{k_j}(t), j = 1, 2, \dots, J$. Therefore, we have

$$(6) \quad \sum_{j=1}^J \left[N_{0 \rightarrow 1}^{k_j}(t) + N_{1 \rightarrow 0}^{k_j}(t) + N_{0 \rightarrow 0}^{k_j}(t) + N_{1 \rightarrow 1}^{k_j}(t) \right] = N$$

Consequently, the formula for probability that a node is in state 1 at time $t + 1$ is

$$\begin{aligned} p(t+1) &= \frac{1}{N} \sum_{j=1}^J [N_{0 \rightarrow 1}^{k_j}(t) + N_{1 \rightarrow 1}^{k_j}(t)] \\ &= c_{k_j}([1 - \alpha(t)]p(t) + \alpha(t) \sum_{s=0}^{k_j} P_{s,k_j}(p(t))[(1 - p(t))r_0^{k_j}(s) + p(t)r_1^{k_j}(s+1)]) \end{aligned}$$

which is exactly the model (1).

The elementary cellular automata Rule 22 maps a node $c_n(t)$ to state 1 at time $t + 1$ if and only if exactly one node has value 1 in its neighborhood then the next time step has value 1. In other words, if exactly one-third of the nodes as presented in Figure 1 are in state 1 in the neighborhood of the selected node. Note that the neighborhood includes the node itself. We propose to extend this rule to a network of size N by using the following general rule:

$$(7) \quad c_n(t+1) = \begin{cases} 1 & \text{if } d_1 \leq \frac{s}{k+1} \leq d_2 \\ 0 & \text{if otherwise} \end{cases}$$

where s is the total number of 1s in the neighborhood of node c_n , and k is the connectivity of the node. Here $0 \leq d_1 \leq d_2 \leq 1$ are fixed parameters. The rule basically states that the node is turned ON if and only if the proportion of 1s in its neighborhood is within given bounds. Otherwise the node is turned OFF. We will

make use of this rule as well as a variant of it in which the value of the node c_n is not included in the sum s and thus the fraction $\frac{s}{k+1}$ becomes $\frac{s}{k}$ in rule (7).

Observe that if the network has 3 nodes and $d_1 = d_2 = \frac{1}{3}$ the rule is exactly the elementary cellular automata Rule 22. Hung et al. [24] have introduced a different generalization of Rule 22, namely a node is turned ON if and only if a single node is ON in the neighborhood of the node under consideration. They use this rule in the context of synchronization of stochastically coupled random Boolean networks. Another direct generalization of Rule 22 would be to use the same proportion for a network with N nodes. The interpretation implies if in a node's neighborhood at most one-third of the nodes are in state 1 then the node becomes 1 at the next time point. Another possible generalization is to use the rule that when the proportion of nodes is exactly one-third then the node is turned ON, otherwise it is OFF. We note that the generalization provided in (7) allows us to extend the study to a wider variety of totalistic rules than the extensions of Rule 22 specified above.

Let us start with rule (7) and provide the exact model for the probability of a node being in state 1. The notation $[x]$ is used to indicate that x is an integer.

Proposition: Under the assumption of a Boolean network evolving according to rule (7) and with the characteristics specified in model (1), the probability $p(t+1)$

of a node being in state 1 at time $t + 1$ given $p(t)$ can be written as

$$p(t + 1) = [1 - \alpha(t)]p(t) + \alpha(t) \sum_{j=1}^J c_j g_j(t)$$

where

$$(8) \quad \begin{aligned} g_{k_j}(t) &= \sum_{s=[d_1^j(k_j+1)]+1}^{[d_2^j(k_j+1)]-1} \binom{k_j}{s} p(t)^s (1-p(t))^{k_j-s} \\ &+ \binom{k_j}{[d_1^j(k_j+1)]} p(t)^{[d_1^j(k_j+1)]+1} (1-p(t))^{k_j-[d_1^j(k_j+1)]} \\ &+ \binom{k_j}{[d_2^j(k_j+1)]} p(t)^{[d_2^j(k_j+1)]} (1-p(t))^{k_j-[d_2^j(k_j+1)]+1} \end{aligned}$$

The values $0 \leq d_1^j \leq d_2^j \leq 1, j = 1, 2, \dots, J$ are the fixed proportions in rule (7) and they can vary for nodes with different number of parents. However, all the nodes with a given number of parents evolve according to the same rule.

Proof. Observe that under the rule (7) we have the following formula for the updating rules $r_0^{k_j}(s)$ used in model (1).

$$r_0^{k_j}(s) = \begin{cases} 1 & \text{if } d_1^j \leq \frac{s}{k_j+1} \leq d_2^j \\ 0 & \text{otherwise} \end{cases}$$

Given that s is an integer, we can rewrite this in terms of the actual values of s as shown below.

$$r_0^{k_j}(s) = \begin{cases} 1 & \text{if } [d_1^j(k_j+1)] + 1 \leq s \leq [d_2^j(k_j+1)] \\ 0 & \text{otherwise} \end{cases}$$

Similarly one can see that

$$r_1^{k_j}(s) = \begin{cases} 1 & \text{if } [d_1^j(k_j+1)] \leq s \leq [d_2^j(k_j+1)] - 1 \\ 0 & \text{otherwise} \end{cases}$$

Thus, the function $g_{k_j}(t)$ in model (1) becomes

$$\begin{aligned}
g_{k_j}(t) &= (1 - p(t)) \sum_{s=0}^{k_j} r_0^{k_j}(s) P_{s,k_j} + p(t) \sum_{s=0}^{k_j} r_1^{k_j}(s) P_{s,k_j} \\
&= (1 - p(t)) \sum_{[d_1^j(k_j+1)]+1}^{[d_2^j(k_j+1)]} P_{s,k_j} + p(t) \sum_{[d_1^j(k_j+1)]}^{[d_2^j(k_j+1)]-1} P_{s,k_j} \\
&= \sum_{[d_1^j(k_j+1)]+1}^{[d_2^j(k_j+1)]-1} P_{s,k_j} + p(t) P_{[d_1^j(k_j+1)],k_j} + (1 - p(t)) P_{[d_2^j(k_j+1)],k_j}
\end{aligned}$$

which yields the final formula. ■

For this study, we focus on synchronous updates of the system. As such, since $\alpha(t)$ is the proportion of nodes to be updated at time t , for synchronous updates the proportion will be $\alpha(t) = 1$. Therefore,

$$p(t+1) = [1 - \alpha(t)]p(t) + \alpha(t) \sum_{j=1}^J c_j g_j(t)$$

simplifies to model (1) as

$$p(t+1) = \sum_{j=1}^J c_j g_j(t)$$

In the next section we use Matlab to generate consecutive states of the model and system in order to identify how well the model matches the reality. We will examine multiple iterations using a variety of parameters.

2.1 Iterations of the System and the Model

It is useful to provide simulations of the model to determine if the system matches the model over time. All the simulations in this work are performed using Matlab and represent synchronous networks (See Appendices A-C for Matlab programs). A match between model, graphically depicted as a continuous line, and system, graphically depicted as individual plots, are considered true if the system closely follows the behavior of the model, even if it is only loosely mimicking the model behavior. The analysis of asynchronous networks will make the object of future research.

In this section we fix the network size to $N = 128$ for simplicity. However, to further validate findings for some of the iterations, 256 nodes were used to determine a match for the model and Boolean system. Those we report on matching appear to be more of a fit when we use $N = 256$, but for those we report on that do not match the model and system continued to demonstrate no match. For each parameter combination we perform various iterations from 1 up to 256 iterations of both the model and the system to surpass transient behavior and we plot the results on the same graph for comparison to identify behavioral trends. We perform such comparisons for the case in which the connectivity k is fixed for all nodes, as well as for the case when two connectivity values k_1 and k_2 are allowed in the network. Moreover, we consider the case of random selection of parent nodes, as well as the case of generalized ECA where the parents of node c_n are $\{c_{n-\frac{k}{2}}, c_{n-\frac{k}{2}+1}, \dots, c_{n-1}, c_{n+1}, c_{n+\frac{k}{2}}\}$

for k even, where N is the network size. Although in the construction of the model it is assumed that the parents are chosen randomly, we will see that in some cases the model is a good approximation for the non-random parent selection specified above.

In each simulation we fix the parameters d_1 and d_2 as well as the proportions c_1 and c_2 when applicable. Values for d_1 and d_2 are chosen to observe behavior when distances $d_2 - d_1$ are small, medium and large. We also consider the cases when $d_1 = 0$ or $d_2 = 1$. Observe that when both these conditions hold, that is, $d_1 = 0$ or $d_2 = 1$, then $c_n(t + 1) = 0$ or $c_n(t + 1) = 1$ and we recover the ECA Rule 126.

2.1.1 Fixed Connectivity

For the case of a fixed connectivity k , the simulations are conducted to observe behavior for small, medium, and large values of the difference $d_2 - d_1$. Values of parameters are chosen as follows: $d_1 = 0.1, 0.2, 0.3, 0.4, 0.5$ and $d_2 = 0.4, 0.6, 0.8, 1.0$. It should be noted that initial iterations using intervals of 0.05 for distances d_1 and d_2 yielded little difference and were thus simplified to intervals of 0.1 for all simulations. Connectivity values include $k = 1, 2, 4, 8, 16, 32, 48$ for simplicity. Although an extensive number of simulations has been performed, in what follows we provide typical graphs and a summary of the findings regarding the (approximate) ranges of parameters that yield a good match versus a mismatch of the model and the system.

Figure 2.1 is representative of trends in the model and Boolean system iterations

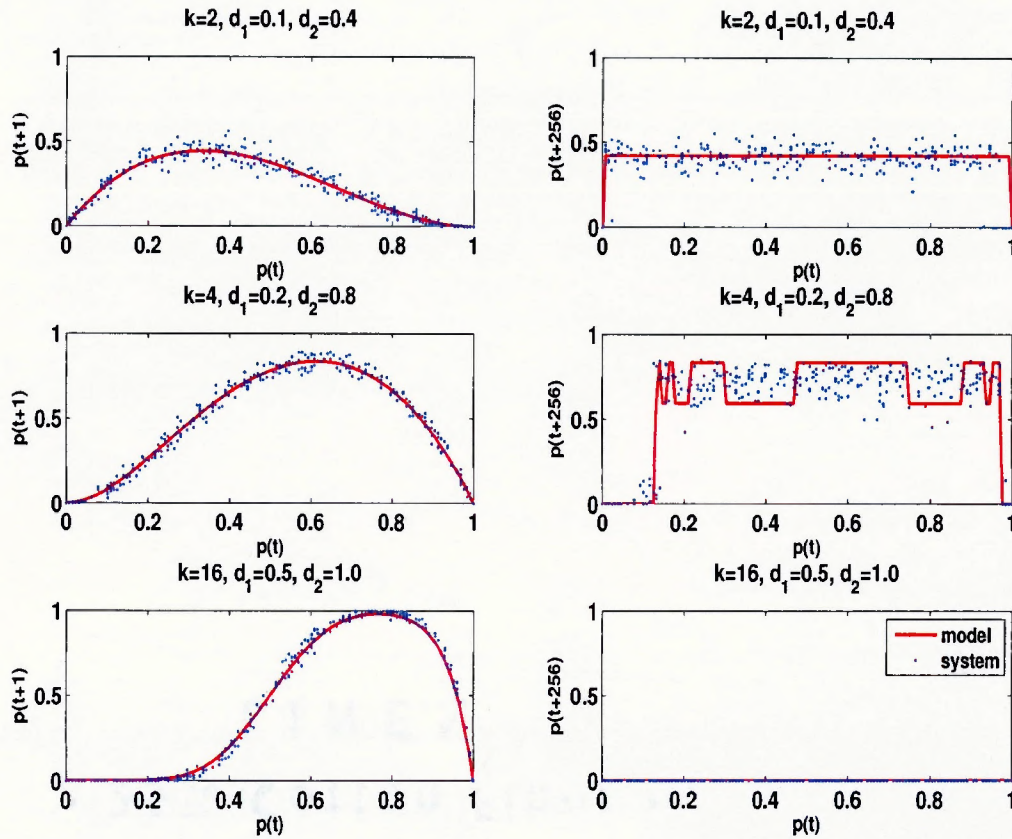


Figure 2.1: Fixed Connectivity with k Randomly Assigned Parents

when $N = 128$ and randomly assigned parents. As it is shown in the left column, after one iteration Figure 2.1 there is an excellent match between the system (points) and the model (continuous) for all three cases considered in this figure. After 256 iterations, the model and system match well although the match is not necessarily as good as in the case of the first iteration. We observe that for $k = 2$ the system and the model reach a horizontal plateau at $p \simeq 0.45$. For $k = 4$, a more dynamic behavior is observed after 256 iterations and the system continues to match the model behavior,

Table 2.1: Model and System Trends for Fixed Connectivity: k Randomly Assigned Parent

k	$d_2 - d_1$	Model/System Description
$1 \leq k \leq 4$	0.1 - 0.7	horizontal line
$4 \leq k \leq 8$	0.2 - 0.9	dynamic model
$k > 8$	any distance	convergence to 0

albeit it loosely. Finally, when $k = 16$ we observe both model and system converge to 0 over time. Table 2.1 presents a summary of the trends observed in Figure 2.1 observed in the various simulations. These are indicative of those plots that yielded a good match between the model and the system and do not include information about systems that do not match the model after 256 iterations. For $k \leq 4$ and $0.1 \leq d_2 - d_1 \leq 0.7$, those matching after 256 iterations tend to be stable. For any distance $d_2 - d_1 \geq 0.2$ the model exhibits more dynamic behavior. When $k > 8$ and $4 \leq k \leq 8$, the model and system converge to 0 regardless of the value of $d_2 - d_1$. The next set of plots in Figure 2.2 represents trends similar to those of the model and Boolean system for random parent assignment; however, these iterations use the k nearest nodes for parent nodes when $N = 128$. As it is shown, the first iteration yields excellent match in all cases. After 256 iterations, the model and system exhibit stability as evidenced by the horizontal line. The case $k = 8$ demonstrates a more chaotic behavior after 256 iterations. The system behavior tracks the model closely enough; it should be noted this particular parameter was tested at $N = 256$ wherein

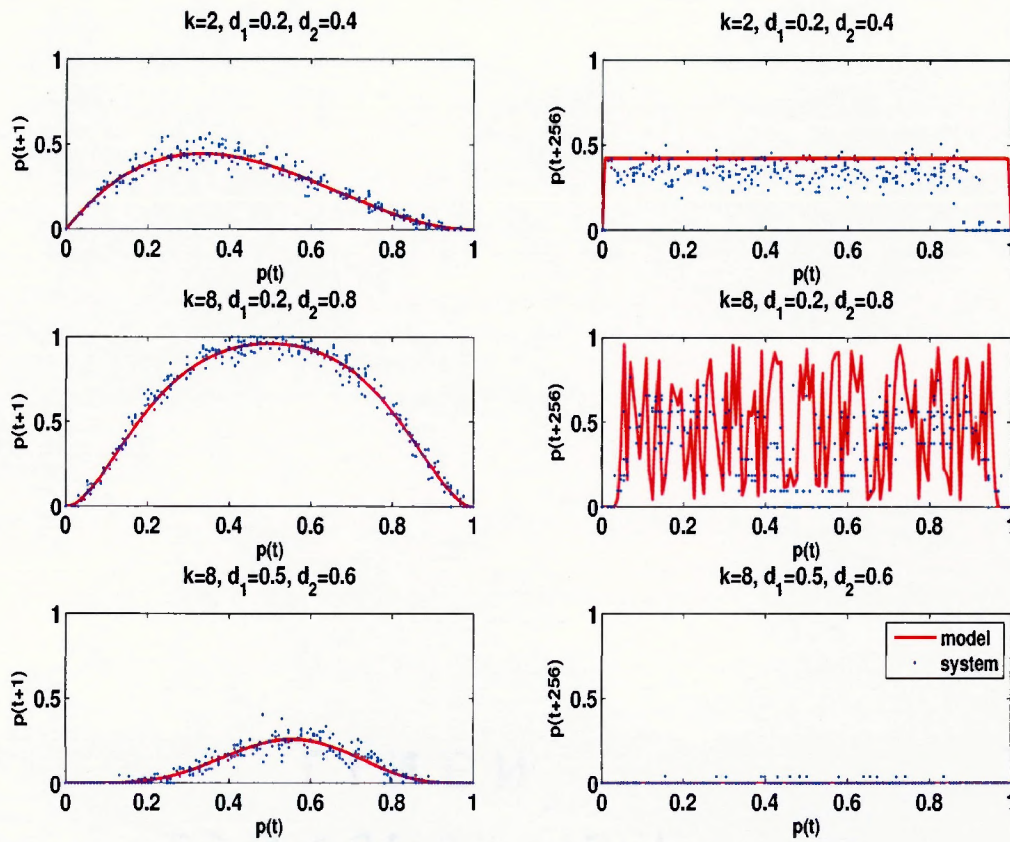


Figure 2.2: Fixed Connectivity with k Nearest Neighbor Parents

the higher concentration of nodes provided a better match to between the model and system. Finally, the case $k = 16$ converges to 0 for both model and system over time. Table 2.2 presents a summary of parameters and brief description for k nearest neighbors. Not all of the plots provided a match or a close approximate match after 256 iterations for both parent types. In order to demonstrate these mismatches, Figure 2.3 is included to display these dynamics. For k randomly assigned parents the first iteration provides a good match. For k nearest neighbors, the first iteration

Table 2.2: Model and System Trends for Fixed Connectivity: k Nearest Neighbor Parent

k parents	$d_2 - d_1$	Model/System Description
$k \leq 2$	≤ 0.5	convergence to 0
$2 \leq k \leq 4$	any distance	horizontal line
$4 \leq k \leq 8$	0.3 - 0.8	dynamic model
$k \geq 8$	any distance	convergence to 0

demonstrates matching behavior by the system. However, after 256 iterations the models are dynamic but the system converges to 0 for both parent types.

2.1.2 Variable Connectivity

For the case of variable connectivity k , we allow two possible values for the number of parents k_1 and k_2 . The simulations were conducted to cover small, medium, and large values of the differences $d_2^1 - d_1^1$ and $d_2^2 - d_1^2$ corresponding to the two Boolean rules used. The actual parameters ranged as follows: $d_1^1 = 0 - 0.6$, $d_2^1 = 0.1 - 0.9$, $d_1^2 = 0.3 - 0.9$, and $d_2^2 = 0.4 - 1.0$. Although an extensive number of simulations has been performed, in what follows we provide typical graphs and a summary of the findings regarding the (approximate) ranges of parameters that yield a good match versus a mismatch of the model and the system. Figure 2.4 is representative of trends in the model and Boolean system iterations when $N = 128$ with randomly assigned parents. As it is shown, the first column is for single iterations to show the match between the model and system. The second column represents trends in the model

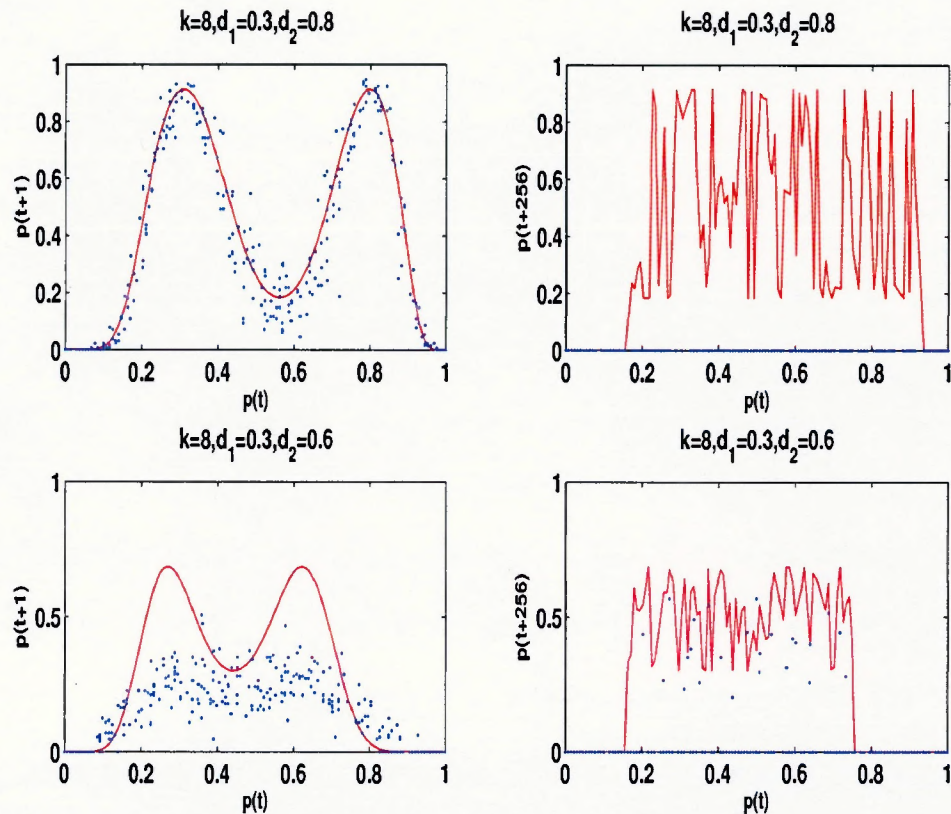


Figure 2.3: Mismatches for Fixed Connectivity. k randomly assigned parents for 1 and 256 iterations (top); k nearest neighbors for 1 and 256 iterations (bottom). Observe that after 256 iterations the system is mainly at 0 while the model is dynamic.

and system after 256 iterations. The first two lines demonstrate stability in the model and good match with the system after 256 iterations. For the second line, the model and system are mostly at 0 with the exception of the activity where $p(t) \simeq 0.65 - 0.95$. The dynamics of the model in third line are loosely matched by the system. Finally, the last line demonstrates the model and system converging to 0 after 256 iterations.

Table 2.3 details the differences and randomly assigned parents based on generalized ECA rules are identified. As you can see, the delineation of trends is not as

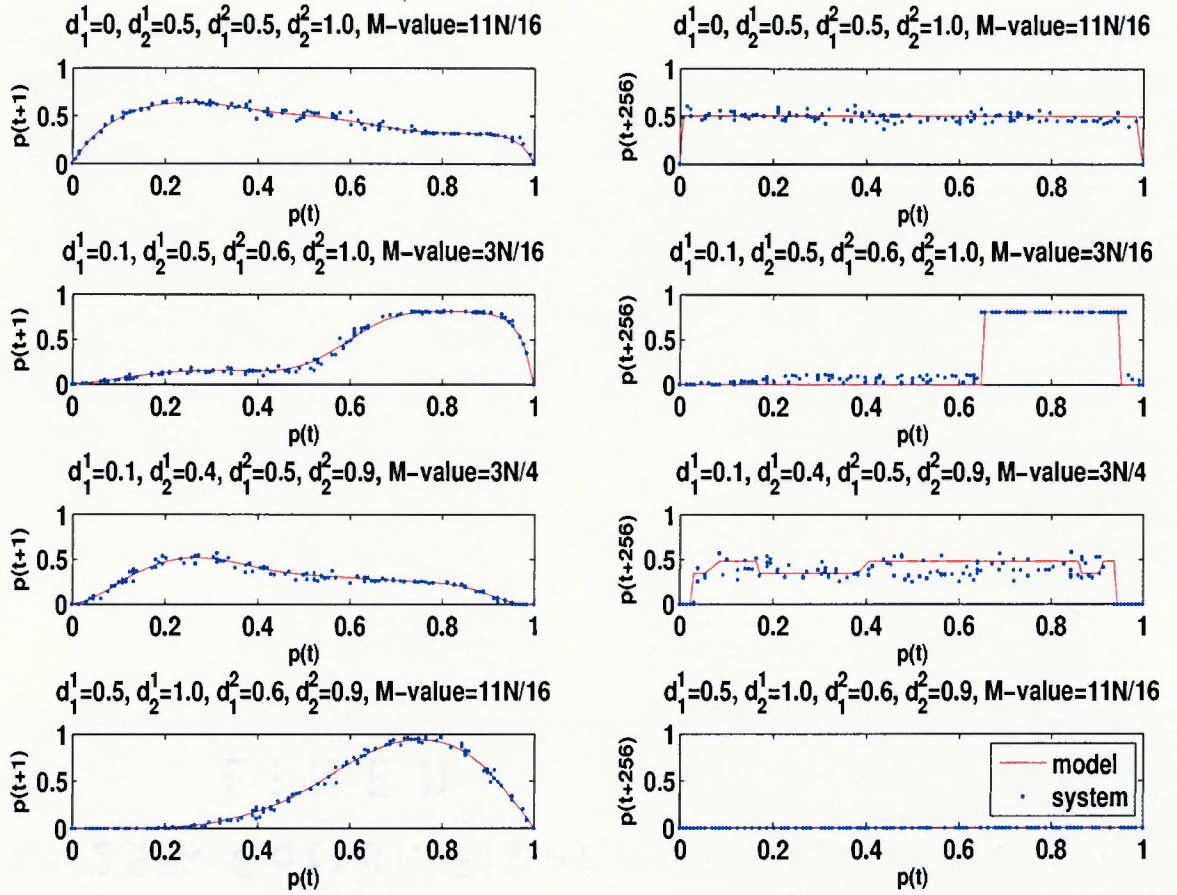


Figure 2.4: Variable Connectivity for k Randomly Assigned Parents

obviously defined as it is for fixed connectivity. The values for d_2^1 , d_2^2 , d_1^1 , and d_2^2 are symmetric, overlapping, small, and large for all of these differences. The M-values in this table and Table 2.4 represent values for k_1 , that is M_1 . The matches and mismatches were present for the combinations of parameters tested. Figure 2.5 is representative of trends in the model and Boolean system for random parent assignment for k nearest nodes for parent nodes. Unlike the dynamics for randomly

Table 2.3: Model and System Trends for Variable Connectivity: k Randomly Assigned Parent

$d_2^1 - d_1^1$	$d_2^2 - d_1^2$	M-values	Model/System Description
0.1 – 0.5	0.1 – 0.6	all values	primarily at 0 with plateau region
0.1 – 0.5	0.1 – 0.6	all values	horizontal line
0.1 – 0.4	0.1 – 0.6	$> \frac{N}{4}$	dynamic model
0.1 – 0.5	0.1 – 0.6	all values	convergence to 0

assigned parents, the matches for k nearest neighbor are more sparse. In the first line, after 256 iterations a stable region is present; the model and system are at 0 for small and large values of $p(t)$. The second line demonstrates a dynamic model and loosely matching system. This is again checked with $N = 256$ which demonstrates a better match between model and system with increased nodes. As with the other iterations, the third line displays when this model and system go to 0. Table 2.4 presents a summary of parameters and brief description for k nearest neighbors. As mentioned for randomly assigned parents for variable connectivity, the table displays the relative uncertainty with which these trends occurred. As with fixed connectivity, not all of

Table 2.4: Model and System Trends for Variable Connectivity: k Nearest Neighbor

$d_2^1 - d_1^1$	$d_2^2 - d_1^2$	M-values	Model/System Description
0.1 – 0.5	0.1 – 0.6	all values	convergence to 0
0.2 – 0.4	0.1 – 0.6	all values	primarily at 0 with plateau region
0.3 – 0.4	0.1 – 0.6	$> \frac{N}{4}$	dynamic model

the plots provided a match or a close approximate match after 256 iterations. In order to demonstrate these mismatches, Figure 2.6 is included to display these dynamics.

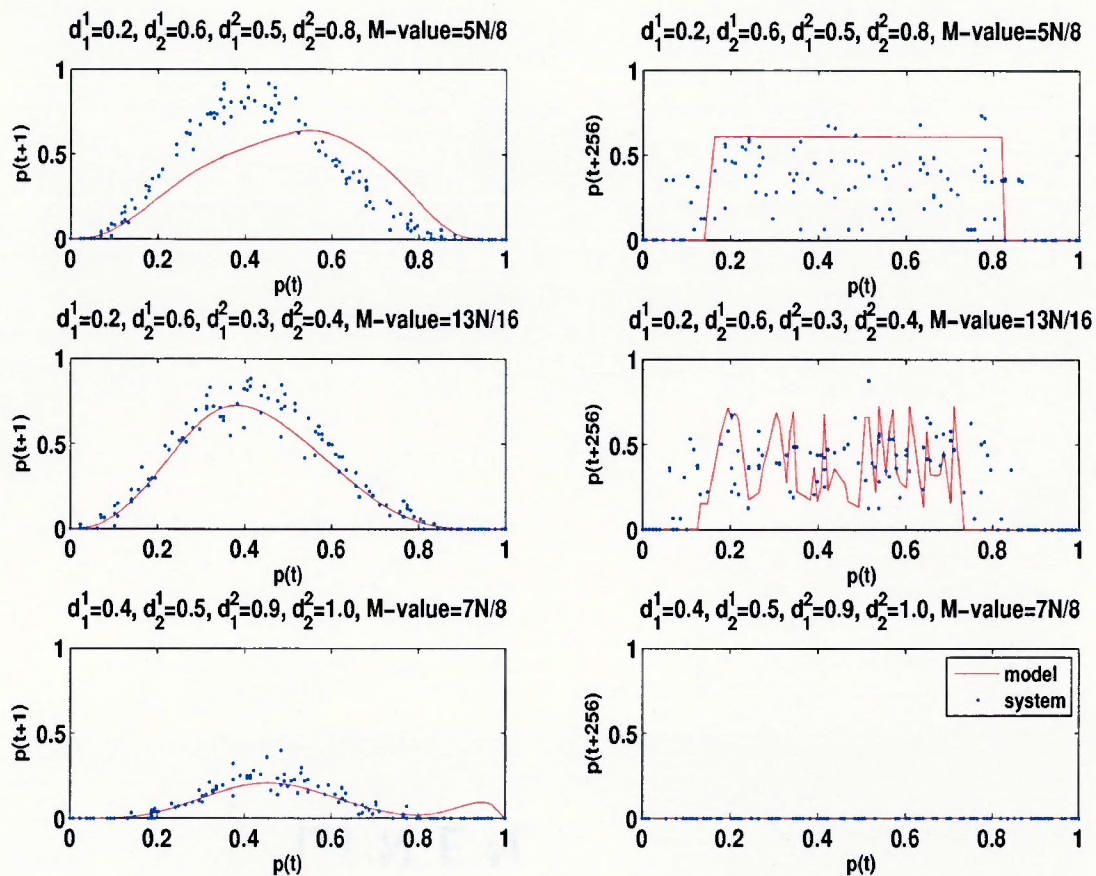


Figure 2.5: Variable Connectivity for k Nearest Neighbors Parents

For k randomly assigned parents in the first line, the first iteration is a match; however, for k nearest neighbors on the second line even the first iteration is not a good match for these parameters. After 256 iterations the system for k randomly assigned parents goes to the origin but the model is dynamic. Where the system behavior is determined by k nearest neighbors, the system is somewhat dynamic but does not mimic the model behavior closely enough to say it matches, particularly since it goes

to the origin once $p(t) > 0.6$.

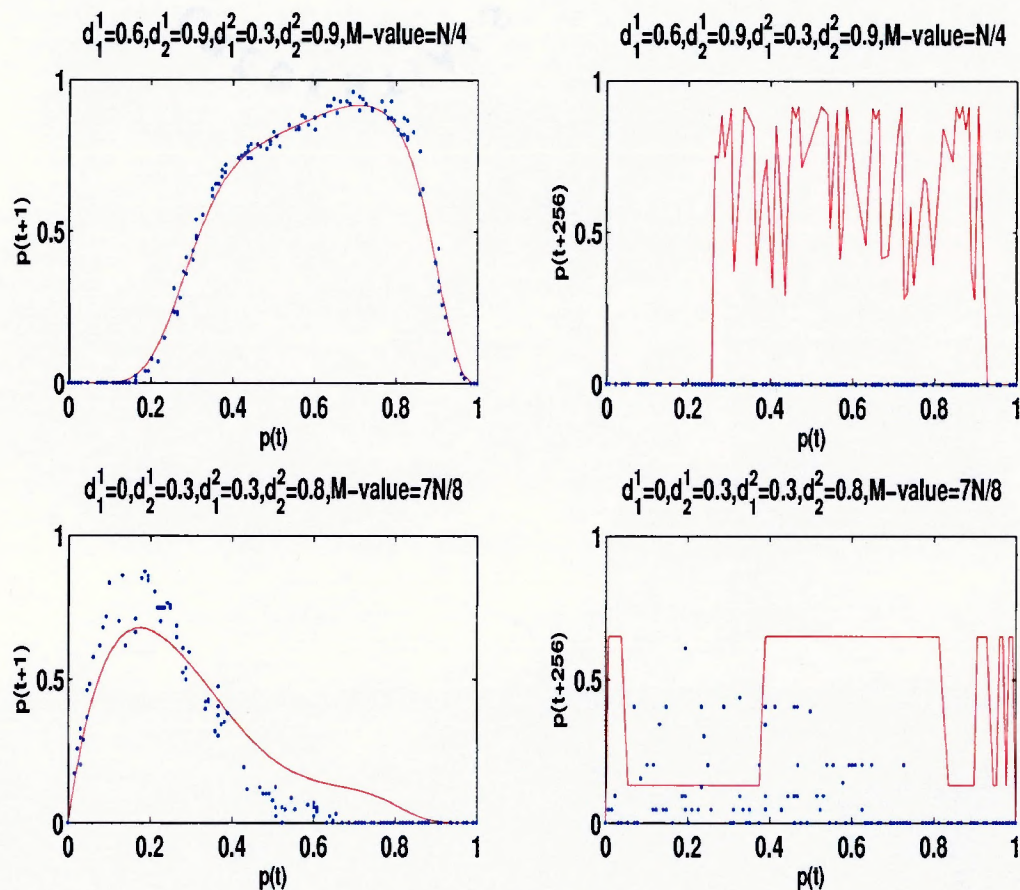


Figure 2.6: Mismatches for Variable Connectivity. k randomly assigned parents for 1 and 256 iterations (top). Observe that after 256 iterations the system is mainly at 0 while the model is dynamic. k nearest neighbors for 1 and 256 iterations (bottom). Observe that after 256 iterations the system is mainly at 0 with some activity while the model is dynamic.

Chapter 3

Analysis of the Dynamics

3.1 Statistical Analysis

To begin the analysis of the dynamics of the model with the Boolean system, statistical analyses are conducted to determine what parameters influence the Boolean system behavior, if any at all. We use contingency tables to record and analyze the relationship between two or more categorical variables. We include a quick review of the test. We let θ_{ij} be the probability of the j th (column) outcome for the i th (row) population of an $r \times c$ table, the null hypothesis test implies

$$\theta_{1j} = \theta_{2j} = \dots = \theta_{rj}$$

for every $j = 1, 2, \dots, c$. The alternative hypothesis is $\theta_{1j}, \theta_{2j}, \dots, \theta_{rj}$ are not equal for at least one value of j .

Let f_{ij} denote the observed frequency for the cell in the i th row and the j th column. The rows total $f_{i.}$ and the columns total $f_{.j}$. The sum of all cells is denoted

f . With this notation, the estimated probabilities of $\theta_{i.}$ and $\theta_{.j}$ are

$$\hat{\theta}_{i.} = \frac{f_{i.}}{f}$$

and

$$\hat{\theta}_{.j} = \frac{f_{.j}}{f}$$

By the null hypothesis for independence,

$$e_{ij} = \hat{\theta}_{i.} \hat{\theta}_{.j} f = \frac{f_{i.} f_{.j}}{f} f = \frac{f_{i.} f_{.j}}{f}$$

With e_{ij} calculated, we can then determine the Chi-square value by the following equation

$$\chi^2 = \sum_{i=1}^r \sum_{j=1}^c \frac{(f_{ij} - e_{ij})^2}{e_{ij}}$$

If this value is greater than $\chi_{\alpha, (r-1), (c-1)}^2$ then the null hypothesis must be rejected [39]. It should be noted that in each of our contingency tables, Chi-square testing is used for simplicity even though some sample sizes may be better served using a test for smaller populations.

3.1.1 Fixed Connectivity

For the purposes of this study, rows consist of whether the system and model match after 256 iterations and the columns represent the parameter whose relation to the result we are testing. For example, Table 3.1 specifically displays the data when $d_2 - d_1 = 0.2$ for k randomly assigned parents, the incidences of matches and mismatches

Table 3.1: χ^2 Test for Fixed Distance with Randomly Assigned Parents

k	1	2	4	8	16	24	32	48	Total
Model/System Not Match	0	0	1	1	0	0	0	0	2
Model/System Match	2	2	1	1	2	2	2	2	14

For fixed distance $d_2 - d_1 = 0.2$ with randomly assigned parents, $\chi^2 = 6.86$ exceeds the test statistic of $\chi_{0.95,7}^2 = 2.167$

are counted for different values of k . Since the Chi-square result exceeds the test statistic, we reject the null hypothesis that k is independent of match or mismatch results. Using this technique for different groups by fixing connectivity k or distances, multiple tests are conducted. In Tables 3.2 and 3.3, we fix k and calculate the number of matches and mismatches for each distance $d_2 - d_1$. For the case of fixed connectivity, contingency tables are generated for both randomly assigned values of k as well as k nearest neighbors for fixed k iterations. For k randomly assigned parents, Table

Table 3.2: χ^2 Test for k Randomly Assigned Parents

$d_2 - d_1$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	Total
Model/System Not Match	1	2	5	3	4	3	3	1	1	23
Model/System Match	12	14	19	13	17	13	10	4	7	109

3.2 demonstrates $\chi_{0.95,1,8}^2 = 1.84$ which is less than the test statistic of 2.733, thus we cannot reject the null hypothesis of independence of k with model/system matches. This indicates the relationship between system and model matches is not dependent on values of $d_2 - d_1$. Table 3.3 details the results of k -nearest neighbors, which also

shows we cannot reject the null hypothesis because $\chi_{0.95,1,8}^2 = 2.66 \leq 2.733$. Therefore, for fixed connectivity of k randomly assigned parents and k -nearest neighbors, the model and system dynamics are not dependent on distance. When additional Chi-

Table 3.3: χ^2 Test for k -Nearest Neighbors

$d_2 - d_1$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	Total
Model/System Not Match	1	3	5	5	5	4	3	1	2	29
Model/System Match	12	13	19	11	16	12	10	4	6	103

square tests are conducted fixing the value of $d_2 - d_1$ and testing for independence of connectivity k , for all distances for both k randomly assigned parents and k -nearest neighbors, we reject the null hypothesis that connectivity has no relationship to the dynamics of the model. The Chi-square test statistic $\chi_{0.95,1,7}^2 = 2.17$ is exceeded for all distances 0.1–0.9. Therefore, for fixed connectivity and either parent type, we can conclude model and system matches or mismatches are dictated by the concentration of parent nodes. Thus values of k have more impact than values of $d_2 - d_1$.

3.1.2 Variable Connectivity

Statistical analyses for variable connectivity are more involved, requiring a variety of contingency tables to test the results. Individual parameters are fixed; i.e., M-values of k_1 , k_1 distance, k_2 distance. Tests are run to determine what relationship each has on the dynamics of the model and system. We also fix two parameters to test the

Table 3.4: χ^2 Test of $d_2^1 - d_1^1$ for Randomly Assigned Parents when M-values Fixed

Match	0.1	0.2	0.3	0.4	0.5	Total	Mismatch	0.1	0.2	0.3	0.4	0.5	Total
$\frac{N}{8}$	4	3	13	7	3	30	$\frac{N}{8}$	0	1	1	2	0	4
$\frac{3N}{16}$	3	3	13	7	3	29	$\frac{3N}{16}$	1	1	1	2	0	5
$\frac{N}{4}$	4	3	10	7	3	27	$\frac{N}{4}$	0	1	4	2	0	7
$\frac{5N}{16}$	4	4	7	4	3	22	$\frac{5N}{16}$	0	0	7	5	0	12
$\frac{3N}{8}$	4	4	8	5	3	24	$\frac{3N}{8}$	0	0	6	4	0	10
$\frac{7N}{16}$	4	4	8	6	3	25	$\frac{7N}{16}$	0	0	6	3	0	9
$\frac{N}{2}$	4	4	7	6	3	24	$\frac{N}{2}$	0	0	7	3	0	10
$\frac{9N}{16}$	4	4	7	6	3	24	$\frac{9N}{16}$	0	0	7	3	0	10
$\frac{5N}{8}$	4	4	7	5	3	23	$\frac{5N}{8}$	0	0	7	4	0	11
$\frac{11N}{16}$	4	3	8	5	3	23	$\frac{11N}{16}$	0	1	6	4	0	11
$\frac{3N}{4}$	4	3	6	5	3	21	$\frac{3N}{4}$	0	1	8	4	0	13
$\frac{13N}{16}$	3	4	5	6	3	21	$\frac{13N}{16}$	1	0	9	3	0	13
$\frac{7N}{8}$	3	2	5	6	3	19	$\frac{7N}{8}$	1	2	9	3	0	15

relationship of the third to the match or mismatch of the system to the model. In what follows is a discussion of the findings for k randomly assigned parents followed by findings for k -nearest neighbors. When we fix the M-values and test the null hypothesis that model/system matches are independent of distances $d_2^1 - d_1^1$ and $d_2^2 - d_1^2$ for k randomly assigned parents, the null hypothesis is rejected for all k_1 distances because Chi-square results for all parameters tested exceeded $\chi_{0.95,1,4}^2 = 0.711$ (Table 3.4). When $\frac{5N}{8} \leq k_1 \leq \frac{11N}{16}$ for distance $d_2^2 - d_1^2$ (Table 3.5), we are unable to reject the null hypothesis that matches in the system are independent of distance $d_2^2 - d_1^2$. Fixing distances $d_2^1 - d_1^1$ and $d_2^2 - d_1^2$ to test the independence of M-values, we reject the null hypothesis $0.3 \leq d_2^2 - d_1^2 \leq 0.5$. Additional testing to provide more specific

Table 3.5: χ^2 Test of $d_2^2 - d_1^2$ for Randomly Assigned Parents when M-values Fixed

Match	0.1	0.2	0.3	0.4	0.5	0.6	Total
$\frac{N}{8}$	6	7	7	6	3	3	32
$\frac{3N}{16}$	5	7	7	5	3	4	31
$\frac{N}{4}$	5	6	6	6	4	2	29
$\frac{5N}{16}$	5	5	5	5	3	1	24
$\frac{3N}{8}$	5	5	6	5	4	1	26
$\frac{7N}{16}$	5	5	7	5	4	1	27
$\frac{N}{2}$	5	5	6	5	3	2	26
$\frac{9N}{16}$	4	5	6	5	3	2	26
$\frac{5N^*}{8}$	4	5	6	5	3	2	25
$\frac{11N^{**}}{16}$	4	4	6	5	3	3	25
$\frac{3N}{4}$	4	4	6	4	3	2	23
$\frac{13N}{16}$	3	5	5	6	3	1	23
$\frac{7N}{8}$	4	4	5	4	3	1	21
Mismatch	0.1	0.2	0.3	0.4	0.5	0.6	Total
$\frac{N}{8}$	0	0	1	1	1	1	4
$\frac{3N}{16}$	1	0	1	2	1	0	5
$\frac{N}{4}$	1	1	2	1	0	2	7
$\frac{5N}{16}$	1	2	3	2	1	3	12
$\frac{3N}{8}$	1	2	2	2	0	3	10
$\frac{7N}{16}$	1	2	1	2	0	3	9
$\frac{N}{2}$	1	2	2	2	1	2	10
$\frac{9N}{16}$	2	2	2	1	1	2	10
$\frac{5N^*}{8}$	2	2	2	2	1	2	11
$\frac{11N^{**}}{16}$	2	3	2	2	1	1	11
$\frac{3N}{4}$	2	3	2	3	1	2	13
$\frac{13N}{16}$	3	2	3	1	1	3	13
$\frac{7N}{8}$	2	3	3	3	1	3	15

* $\chi^2 = 0.935$ and ** $\chi^2 = 0.767$ are less than $\chi^2 = 1.145$ so we cannot reject the null hypothesis.

Table 3.6: χ^2 Test of $d_2^1 - d_1^1$ for k -Nearest Neighbor when M-values Fixed

Match	0.1	0.2	0.3	0.4	0.5	Total	Mismatch	0.1	0.2	0.3	0.4	0.5	Total
$\frac{N}{8}$	4	3	12	3	2	24	$\frac{N}{8}$	0	1	2	6	1	10
$\frac{3N}{16}$	3	2	11	1	2	19	$\frac{3N}{16}$	1	2	3	8	1	15
$\frac{N}{4}$	2	3	7	2	2	16	$\frac{N}{4}$	2	1	7	7	1	18
$\frac{5N}{16}$	2	2	5	2	2	13	$\frac{5N}{16}$	2	2	9	7	1	21
$\frac{3N}{8}$	3	2	6	2	2	15	$\frac{3N}{8}$	1	2	8	7	1	19
$\frac{7N}{16}$	3	2	6	2	2	15	$\frac{7N}{16}$	1	2	8	7	1	19
$\frac{N}{2}$	3	3	1	3	2	12	$\frac{N}{2}$	1	1	13	6	1	22
$\frac{9N}{16}$	3	3	2	4	2	14	$\frac{9N}{16}$	1	1	12	5	1	20
$\frac{5N}{8}$	3	2	3	4	2	14	$\frac{5N}{8}$	1	2	11	5	1	20
$\frac{11N}{16}$	3	2	3	4	2	14	$\frac{11N}{16}$	1	2	11	5	1	20
$\frac{3N}{4}$	3	2	2	4	1	12	$\frac{3N}{4}$	1	2	12	5	2	22
$\frac{13N}{16}$	3	2	3	4	1	13	$\frac{13N}{16}$	1	2	11	5	2	21
$\frac{7N}{8}$	3	1	4	5	1	14	$\frac{7N}{8}$	1	3	10	4	2	20

insight for these results involved fixing two parameters; i.e., M-value and distance $d_2^1 - d_1^1$, M-value and distance $d_2^2 - d_1^2$, distances $d_2^1 - d_1^1$ and $d_2^2 - d_1^2$. We randomly selected $d_2^1 - d_1^1$ and $d_2^2 - d_1^2$ equal 0.3 or 0.4 for additional testing. The results show that when $\frac{N}{2} \leq M \leq \frac{3N}{4}$ and $d_2^1 - d_1^1 = 0.3$, the distances $d_2^2 - d_1^2$ have no relation to the system and model matches. Yet when M-values and distances $d_2^2 - d_1^2$ are fixed, the $d_2^1 - d_1^1$ parameter is influential on the real system matching. Therefore, our studies show when $\frac{N}{2} \leq M \leq \frac{3N}{4}$, $d_2^1 - d_1^1 = 0.3$, and $0.3 \leq d_2^2 - d_1^2 \leq 0.5$ no single parameter has greater influence on the model behavior.

The analyses for k -nearest neighbors indicate that for all distances and M-values the dynamics of the system and model are dependent on all parameters tested. Fixing

the M-values against distances $d_2^1 - d_1^1$ and $d_2^2 - d_1^2$ show the null hypothesis is rejected for all values because Chi-square results exceed $\chi^2 = 0.711$ (Table 3.6) and $\chi^2 = 1.145$ (Table 3.7). When $d_2^1 - d_1^1$ distances equalled 0.2 and 0.5, the Chi-square test indicates the M-values for these two values do not have a relationship to the dynamics of the model and system. Concurrently, when $d_2^2 - d_1^2$ equals 0.1 and 0.4, the Chi-square test indicates independence for the M-values as well. However, given the samples sizes and the fact the Chi-square results were so close to the test statistic, we can infer that if the sample size from which to choose would have been larger the null hypothesis would be rejected for all four values.

Because of the results, we fix two values to further identify if particular parameters truly do not have a relationship to the dynamics of the model and system. We fix M-values and k_1 distance, M-values and k_2 distances, and k_1 and k_2 distances. These results confirm our hypothesis. Matches and mismatches between model and system after 256 iterations may occur at any point regardless of parent type. This implies this particular model is unpredictable and may make for a less than optimal model for use in testing real systems.

Based on our analyses, when connectivity is fixed, values of k have more impact on the model and system matching. This is true for k randomly assigned parents and for k -nearest neighbors. In contrast, when connectivity is variable, model and system matches cannot be definitively attributed to distances $d_2^1 - d_1^1$ and $d_2^2 - d_1^2$ or

Table 3.7: χ^2 Test of $d_2^2 - d_1^2$ for k -Nearest Neighbor when M-Values Fixed

Match	0.1	0.2	0.3	0.4	0.5	0.6	Total
$\frac{N}{8}$	5	7	5	4	2	3	26
$\frac{3N}{16}$	4	7	4	2	2	2	21
$\frac{N}{4}$	3	6	3	2	3	1	188
$\frac{5N}{16}$	3	4	3	2	2	1	15
$\frac{3N}{8}$	3	4	5	3	1	1	179
$\frac{7N}{16}$	3	4	6	2	1	1	17
$\frac{N}{2}$	3	3	4	3	0	1	142
$\frac{9N}{16}$	3	3	5	3	0	2	16
$\frac{5N}{8}$	3	3	4	4	0	2	16
$\frac{11N}{16}$	3	3	4	4	0	2	16
$\frac{3N}{4}$	3	4	3	2	0	2	14
$\frac{13N}{16}$	3	4	3	3	0	2	15
$\frac{7N}{8}$	3	5	3	3	1	1	16
Mismatch	0.1	0.2	0.3	0.4	0.5	0.6	Total
$\frac{N}{8}$	1	0	3	3	2	1	10
$\frac{3N}{16}$	1	0	3	3	2	1	10
$\frac{N}{4}$	3	1	5	5	1	3	18
$\frac{5N}{16}$	3	3	5	5	2	3	21
$\frac{3N}{8}$	3	3	3	4	3	3	19
$\frac{7N}{16}$	3	3	2	5	3	3	19
$\frac{N}{2}$	3	4	4	4	4	3	22
$\frac{9N}{16}$	3	4	3	4	4	2	20
$\frac{5N}{8}$	3	4	4	3	4	2	20
$\frac{11N}{16}$	3	4	4	3	4	2	20
$\frac{3N}{4}$	3	3	5	5	4	2	22
$\frac{13N}{16}$	3	3	5	4	4	2	21
$\frac{7N}{8}$	3	2	5	4	3	3	20

M-values.

3.2 Bifurcation Diagrams and Lyapunov Exponents

In this section we present bifurcation diagrams and corresponding Lyapunov exponent computations for some of the parameter values that yield a good match of the model and the system. The goal is to provide a basic understanding of the dynamics of the system.

Bifurcation diagrams are graphical tools that allow us to understand if the system exhibits periodic points in the long run or chaos. To generate a bifurcation diagram, we first fix all the parameters except k which is allowed to vary. For each value of k the model is iterated a number of times to pass the transient phase and then the resulting values of $p(t)$ are plotted on the vertical line passing through k for many initial conditions. For example, if the resulting plot is one single point, then the system exhibits one fixed point which attracts all the orbits, and therefore it is stable. If the resulting plot for a specific value of k is two points, then the system has a stable period-2 orbit. The passage from one fixed point to a period-2 orbit indicates a period-doubling bifurcation. On the other hand, if for a fixed value of k the resulting plot is a collection of points with no visible order, then there is a possibility of higher order periodic orbits or chaos.

In addition, to supplement the results from the bifurcation diagrams, one needs to

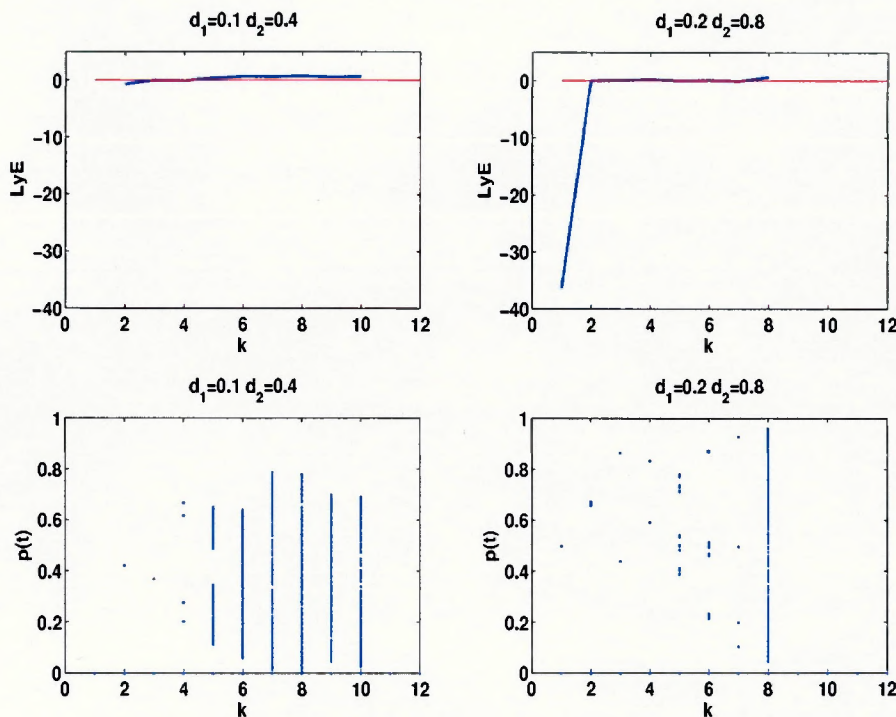


Figure 3.1: Lyapunov Exponents (LyE) and bifurcation diagrams for selected parameters with $N = 128$. Observe in the first row for small values of k LyE is negative, indicating order in the system. As k increases, LyE is positive and thus the system exhibits sensitivity of the orbits to initial values. The bifurcation diagrams in the second row indicate that several values of k yield a wide range of possible values. Combined with the positive Lyapunov Exponents this is suggestive of chaos.

also analyze the sensitivity of the orbits to initial values. This is done mainly through the computation of the Lyapunov exponents which provide an approximation for the average multiplicative rate of divergence or convergence of orbits starting at nearby points. The Lyapunov exponent is defined to be

$$(9) \quad L(x_0; f) = \lim_{n \rightarrow \infty} \left[\prod_{j=0}^{n-1} |f'(x_j)| \right]^{\frac{1}{n}}$$

where $f(x)$ is the map describing the evolution of the system from one time point to

another, and $\{x_0, x_1, \dots, x_n\}$ represents the trajectory of x_0 under the map f . Because we are calculating Lyapunov exponents for the function defining the system, time delay estimates and embedding dimensions are not estimated. When the Lyapunov exponent of an orbit is negative, the implication is that a stable orbit is attained. Conversely, when the Lyapunov exponent is positive, the implication is that chaos is present, provided the orbit is not asymptotically periodic [2]. Any orbit that is attracted to a sink is asymptotically periodic. It should be noted that Lyapunov exponents are undefined for some orbits, particularly an orbit containing a point x_i with $f'(x_i) = 0$ causes it to be undefined.

Observe that in the case of the model (1), to compute the Lyapunov exponents we first need the derivative of the function

$$(10) \quad f(p) = \sum_{j=1}^J c_j \left[(1-\alpha)p + \alpha \sum_{s=0}^{k_j} [(1-p)r_0^{k_j}(s) + pr_1^{k_j}(s)] \binom{k_j}{s} p^s (1-p)^{k_j-s} \right] =$$

$$= (1-\alpha)p + \alpha \sum_{j=1}^J c_j \sum_{s=0}^{k_j} [(1-p)r_0^{k_j}(s) + pr_1^{k_j}(s)] \binom{k_j}{s} p^s (1-p)^{k_j-s}.$$

Then

$$(11) \quad f'(p) = (1-\alpha) + \alpha \sum_{j=1}^J c_j \left\{ [-r_0^{k_j}(0) + r_1^{k_j}(0)](1-p)^{k_j} - [(1-p)r_0^{k_j}(0) + pr_1^{k_j}(0)]k_j(1-p)^{k_j-1} + \right.$$

$$+ \sum_{s=1}^{k_j-1} \left[[-r_0^{k_j}(s) + r_1^{k_j}(s)] \binom{k_j}{s} p^s (1-p)^{k_j-s} + \right.$$

$$\left. \left. + [(1-p)r_0^{k_j}(s) + pr_1^{k_j}(s)] \binom{k_j}{s} [sp^{s-1}(1-p)^{k_j-s} - p^s(k_j-s)(1-p)^{k_j-s-1}] \right] \right\} +$$

$$+[-r_0^{k_j}(k_j) + r_1^{k_j}(k_j)]p^{k_j} + [(1-p)r_0^{k_j}(k_j) + pr_1^{k_j}(k_j)]k_j p^{k_j-1} \Big\}.$$

We use the formula (10) in the computation of the Lyapunov exponents (9).

In the simulations we fix the number of nodes to $N = 128$, and we iterate the system 1000 time points before plotting the bifurcation diagrams. Matlab is used to generate the simulations (See Appendix D for Matlab program). We vary the connectivity values k between 1 and 12 for both the case of a fixed connectivity for all nodes, and the two-dimensional case of two different allowable connectivity values (recall that for larger values of k both the system and the model converge to the origin). Given that the values of k are integer, the diagrams show separated vertical lines, and the diagrams cannot be refined. We use the initial point $p_0 = 0.2$ for the Lyapunov exponent computations, but any other choice would yield a similar result. Observe that again $\alpha = 1$, so the system is synchronous.

In Figure 3.1 we present the evolution for a network whose parameters provide a match after 256 iterations for fixed connectivity (As seen in Figure 2.1). When $d_1 = 0.1$ and $d_2 = 0.4$ we can see as the value of k increases ($k \geq 4$), the Lyapunov exponents become positive indicating sensitivity to initial values for these parameters. When $d_1 = 0.2$ and $d_2 = 0.8$ and $k > 8$, the dynamics of the model become chaotic. In the second line, as k increases the points indicate more chaotic behavior in these bifurcation diagrams.

We note that similar graphs are obtained for other parameter combinations.

3.3 Fixed Points and Delay Plots

3.3.1 Focus on Fixed Points

It is useful to understand how the fixed points of the map behave. To do this we need to solve the equation $f(p) = p$ where $f(p)$ is given in formula (10). Observe that this equation can be written as follows:

$$(12) \quad p = \sum_{j=1}^J c_j \sum_{s=0}^{k_j} [(1-p)r_0^{k_j}(s) + pr_1^{k_j}(s)] \binom{k_j}{s} p^s (1-p)^{k_j-s}.$$

We can only deal with this equation numerically. Observe that $p = 0$ is automatically a fixed point since by the way the Boolean rule was defined we have $r_0^{k_j}(0) = 0$ because if all the nodes are zero then the output is a zero. We will analyze this case in more detail. For other fixed points we will use a procedure that will be described later.

By replacing $p = 0$ in formula (11) with $k_j > 1$, we obtain

$$f'(p) = (1 - \alpha) + \alpha \sum_{j=1}^J c_j \left\{ [-r_0^{k_j}(0) + r_1^{k_j}(0)] - k_j r_0^{k_j}(0) + k_j r_0^{k_j}(1) \right\}$$

Since $r_0^{k_j}(0) = 0$ and when $\alpha = 1$ (that is the system is synchronous), this then simplifies to

$$f'(0) = \sum_{j=1}^J c_j [r_1^{k_j}(0) + k_j r_0^{k_j}(1)]$$

Since the functions $r_0^{k_j}$ and $r_1^{k_j}$ can take on only values 0 and 1, there is a total of four possible combinations for any given k_j . For simplicity we will discuss in detail the special case when the rules are the same for all connectivity values. That is the values $r_0^{k_j}$ and $r_1^{k_j}$ are either 0 or 1 for all k_j .

- **Case 1:** $r_1^{k_j}(0) = r_0^{k_j}(1) = 0$ for all k_j .

Then observe that $|f'(0)| = 0 < 1$; therefore the origin is stable. Observe that if the values of k_j are large enough and the values of the corresponding d_1^j in the generalization of Rule 22 are also large enough, then this case is satisfied automatically. This is in agreement with the previous observations that in many cases, the probability $p(t)$ converges to 0.

- **Case 2:** $r_1^{k_j}(0) = 0$ and $r_0^{k_j}(1) = 1$ for all k_j .

Then $|f'(0)| = \sum_{j=1}^J c_j k_j$. For example, for single connectivity k , the stability condition $|f'(0)| < 1$ is equivalent to $|k| < 1$, which is not possible since $k > 1$. In this case, 0 is unstable. For two values of connectivity $k_1 > 1$ and $k_2 > 1$, observe that

$$c_1 k_1 + c_2 k_2 \leq 1 \Leftrightarrow c_1 k_1 + (1 - c_1) k_2 = k_2 + c_1 (k_1 - k_2) \leq 1.$$

If $k_1 \geq k_2$ then clearly this quantity is at least $k_2 > 1$ and thus the origin is unstable. If $k_1 < k_2$ then the condition becomes

$$c_1 \geq \frac{k_2 - 1}{k_2 - k_1}.$$

Since $c_1 < 1$ we have that $k_2 - 1 \leq k_2 - k_1 \rightarrow k_1 \leq 1$ which is not possible.

Thus $c_1 k_1 + c_2 k_2 > 1$ and therefore the origin is unstable.

In general, observe that

$$\begin{aligned} \sum_{j=1}^J c_j k_j &= c_1 k_1 + c_2 k_2 + \cdots + c_{J-1} k_{J-1} + (1 - c_1 - c_2 - \cdots - c_{J-1}) k_J = \\ &= c_1(k_1 - k_J) + c_2(k_2 - k_J) + \cdots + c_{J-1}(k_{J-1} - k_J) + k_J. \end{aligned}$$

We can always order the connectivity values such that $k_1 \geq k_2 \geq \cdots \geq k_J$.

Thus all the terms $k_j - k_J$ are nonnegative in the expression above and thus the entire expression is at least $k_J > 1$. Thus the origin is unstable.

- **Case 3:** $r_1^{k_j}(0) = 1$ and $r_0^{k_j}(1) = 0$ for all k_j .

Then $f'(0) = \sum_{j=1}^J c_j = 1$ which indicates $f'(0) = 1$. Thus we cannot make decisions about the stability of 0.

- **Case 4:** $r_1^{k_j}(0) = r_0^{k_j}(1) = 1$ for all k_j .

Then $f'(0) = \sum_{j=1}^J c_j(1 + k_j) = \sum_{j=1}^J c_j + \sum_{j=1}^J c_j k_j = 1 + \sum_{j=1}^J c_j k_j > 1$ which means that the origin is unstable.

In conclusion, for the special case when the Boolean rules are chosen such that the values $r_1^{k_j}(0)$ and $r_0^{k_j}(1)$ are the same for all connectivity values k_j , the origin could be stable or unstable per the cases above, and there are situations in which we cannot make a decision based on the value of $f'(0)$.

In order to find the fixed points according to the formula (12) we have set up an initial Matlab program to solve the equation numerically. However, due to certain

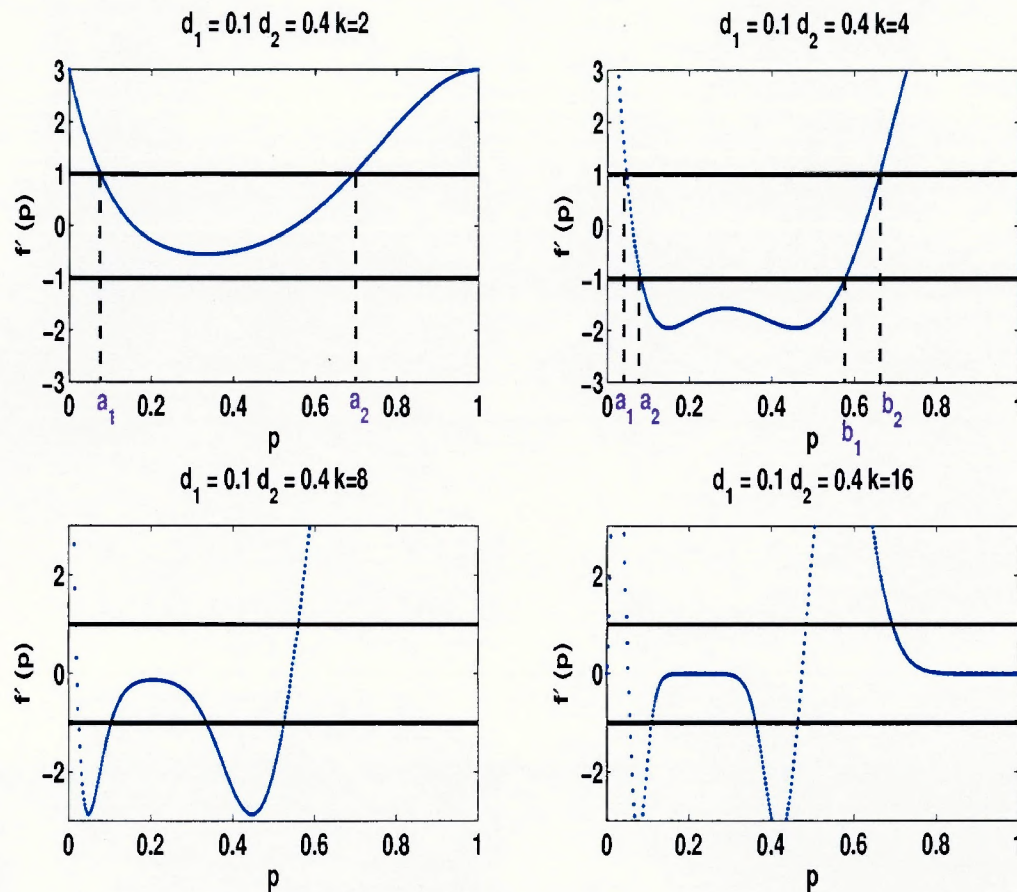


Figure 3.2: Fixed points $d_1 = 0.1$ and $d_2 = 0.4$ for different k

difficulties with the symbolic toolbox of Matlab, we will have to involve a different approach to understand the stability of the fixed points without actually solving the equation numerically. To this aim, in formula (11) for $f'(p)$ we replace p by its expansion (12) and then plot the resulting function of p . Values of $f'(p)$ within the range of $(-1, 1)$ generate stability; i.e., the fixed points falling within the range yielding $f'(p) \in (-1, 1)$ are stable. This does not indicate we actually have fixed points in this range, but if they exist then they are stable attractors. Those outside

the range $(-1, 1)$ generate instability.

To demonstrate this, we plot fixed points for distance parameters d_1 and d_2 identical to those used to model fixed connectivity with various values of k using Matlab (See Appendix E for Matlab program). In Figure 3.2 when $k = 2$, the fixed points are in the approximate interval (a_1, a_2) . The fixed points in this range are stable. Notice the fixed points in the approximate ranges of $(0, a_1)$ and $(a_2, 1)$ fall outside the range of $(-1, 1)$ and are thus unstable. Conversely, when $k = 4$ there is a much smaller range of stable points. Notice the approximate ranges (a_1, a_2) and (b_1, b_2) . These fixed points are stable. Fixed points falling in the approximate ranges of $(0, a_1)$, (a_2, b_1) , and $(b_2, 1)$ are unstable. Given these examples, the reader can thus ascertain the stable and unstable regions on the remaining plots in Figure 3.2 as well as those in Figures 3.3 and 3.4. Notice when $k = 16$ in Figure (11), the origin is stable. Additionally, in Figure 3.3, when $k \geq 4$, the origin is stable for these parameters; the origin is stable for all k parameters in Figure 3.4.

3.3.2 Delay Plots and Attractors

For some of the parameter combinations that yield a match of the model and the system, three dimensional delay plots were generated by plotting $p(t)$ versus $p(t - 1)$ and $p(t - 2)$ for a set of 1000 consecutive time points, and for various values of the parameters as specified in the graphs using Matlab (See Appendix F for Matlab

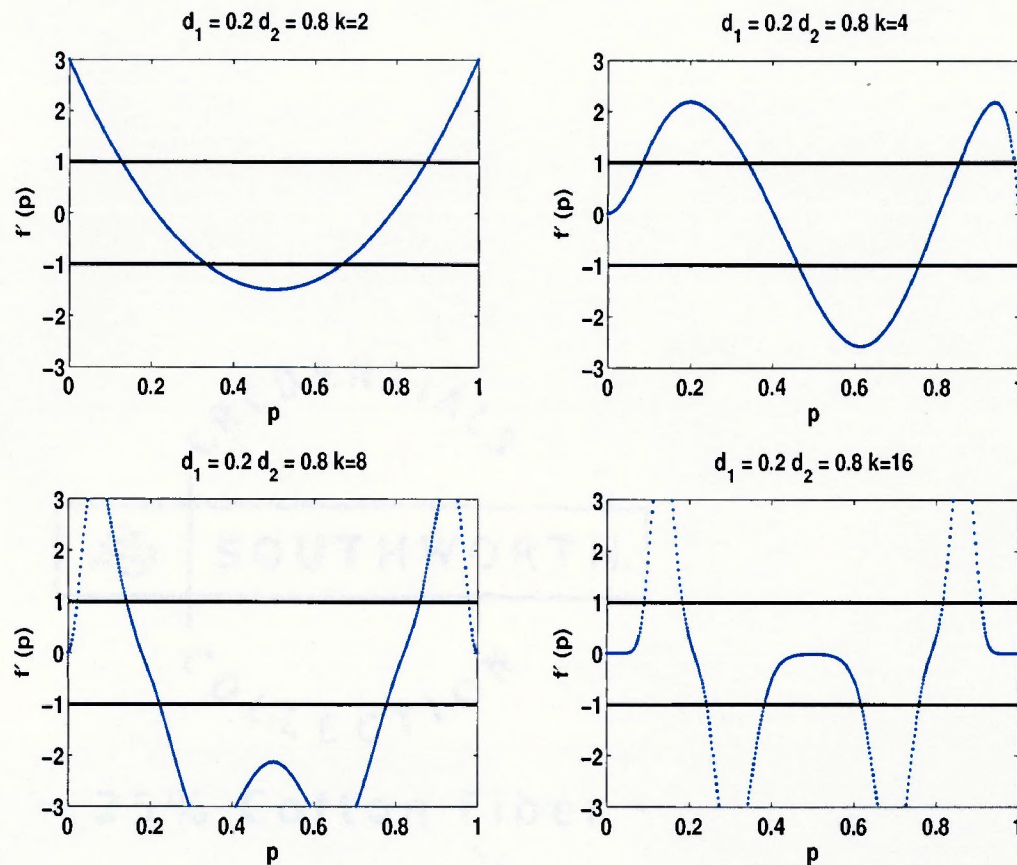


Figure 3.3: Fixed points $d_1 = 0.2$ and $d_2 = 0.8$ for different k

program). The usage of three dimensions allows the untangling of the two dimensional bifurcation diagrams for a better view. The system is iterated 1000 times prior to plotting to surpass the transient period. We involve a coloring procedure that allows us to determine the attractors and identify stable points over time (See Figure 3.5). The colors indicate if there are ordered (same color in a continuous area) or chaotic attractors (the multiple colors do not follow any pattern) over time. For example, in Figure 3.6 when $d_1 = 0.1$ and $d_2 = 0.4$, we observe attractors of period one when $k = 2$

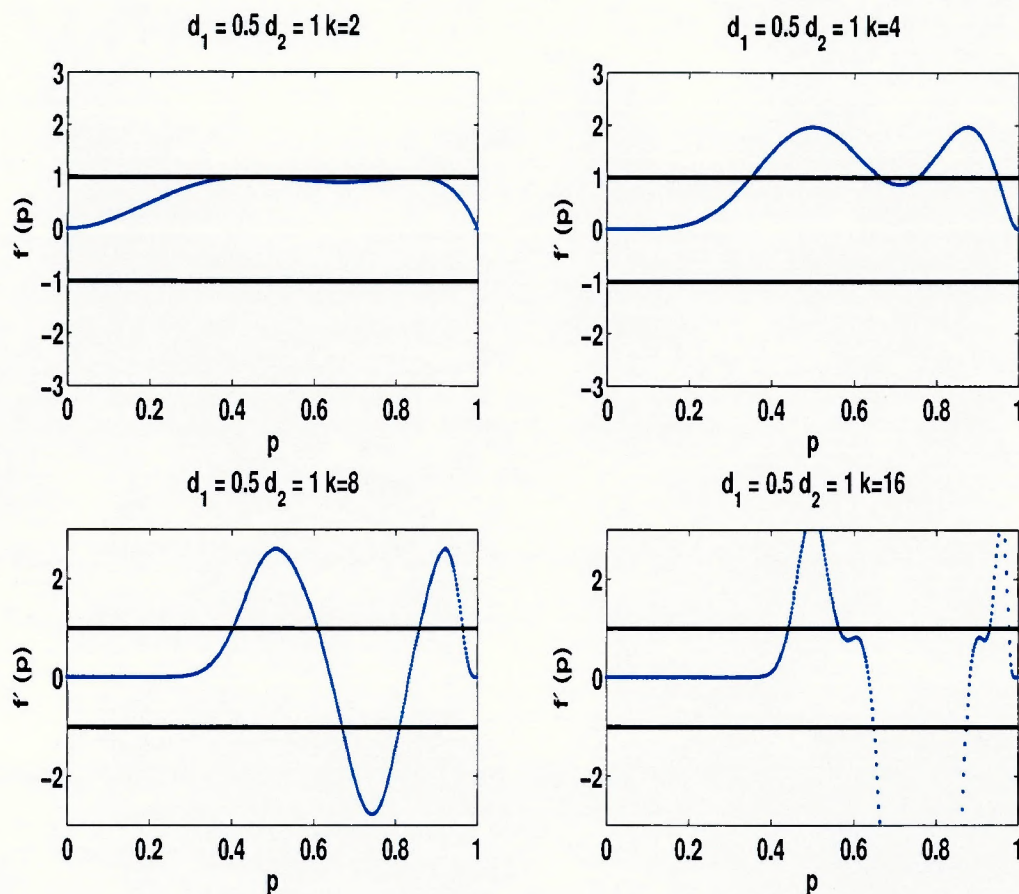


Figure 3.4: Fixed points $d_1 = 0.5$ and $d_2 = 1.0$ for different k

and $k = 3$. When $k = 4$, we observe period four attractors, which is demonstrated by the bifurcation diagram in Figure 3.1. However, for $5 \leq k \leq 10$, there is less ordering of the shades over time which suggests the points change in an irregular, chaotic fashion over time. From the fixed points plots in Figure 3.2, the delay plots also confirm findings when $k > 10$ the model and real Boolean system are stable at the origin. When analyzing delay plots for $d_1 = 0.2$ and $d_2 = 0.8$ (Figure 3.7) and many values of k , we observe period two, four and greater attractors. It is only when

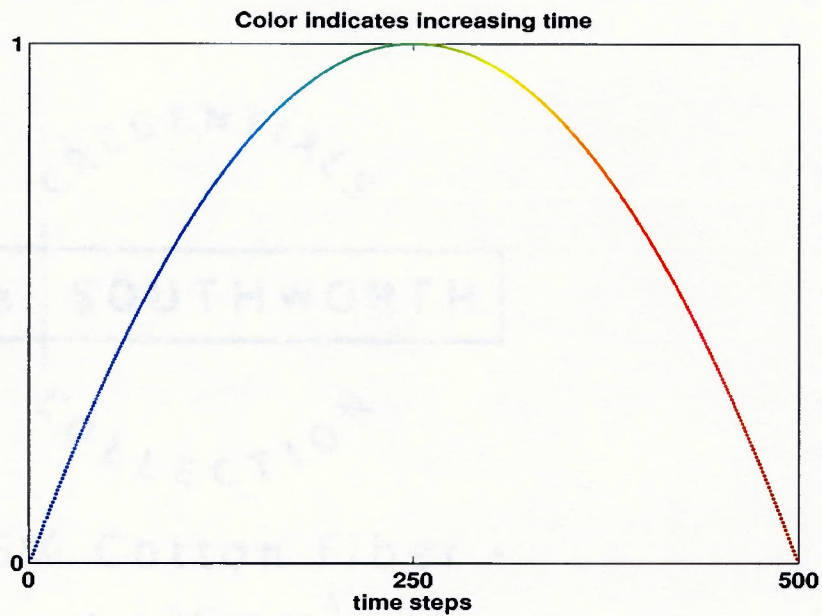


Figure 3.5: Color Code for Delay Plots

$k = 8$ that there is demonstrated chaotic behavior. This correlates to the bifurcation diagram in Figure 3.1. The stable point plots in Figure 3.3 also show there are very few possible stable points for $k = 8$. As is exhibited by the bifurcation and stable point plots, the delay plots also confirm when $k > 8$ the model and real Boolean system are stable at the origin.

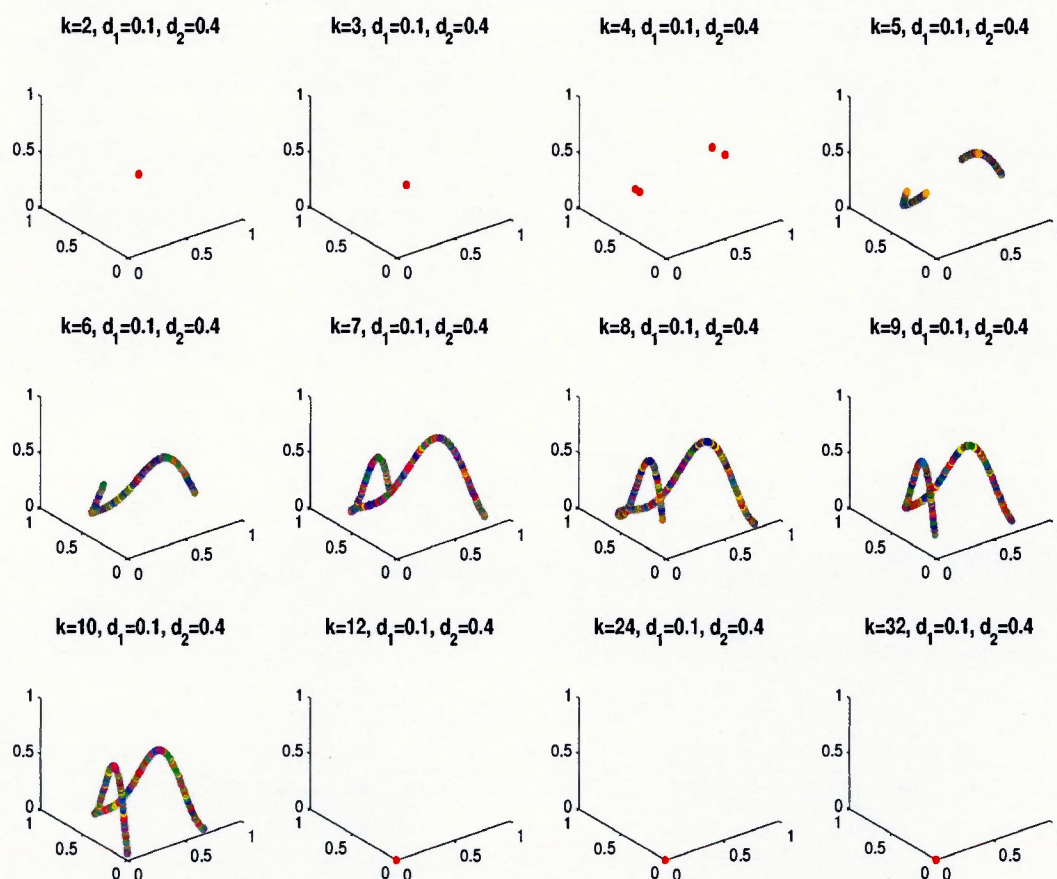


Figure 3.6: Three-Dimensional Delay Plots for a series of 1000 consecutive time points for $d_1 = 0.1$, $d_2 = 0.4$. The graphs are in the unit cube $[0; 1] \times [0; 1] \times [0; 1]$. The solid shades induced by a coloring procedure suggest ordered attractors for some values of $k = 2, 3, 4, 12, 24, 32$ and varied colors suggest chaotic attractors for $k = 5, 6, 7, 8, 9, 10$. This agrees with the bifurcation diagram in Figure 3.1.

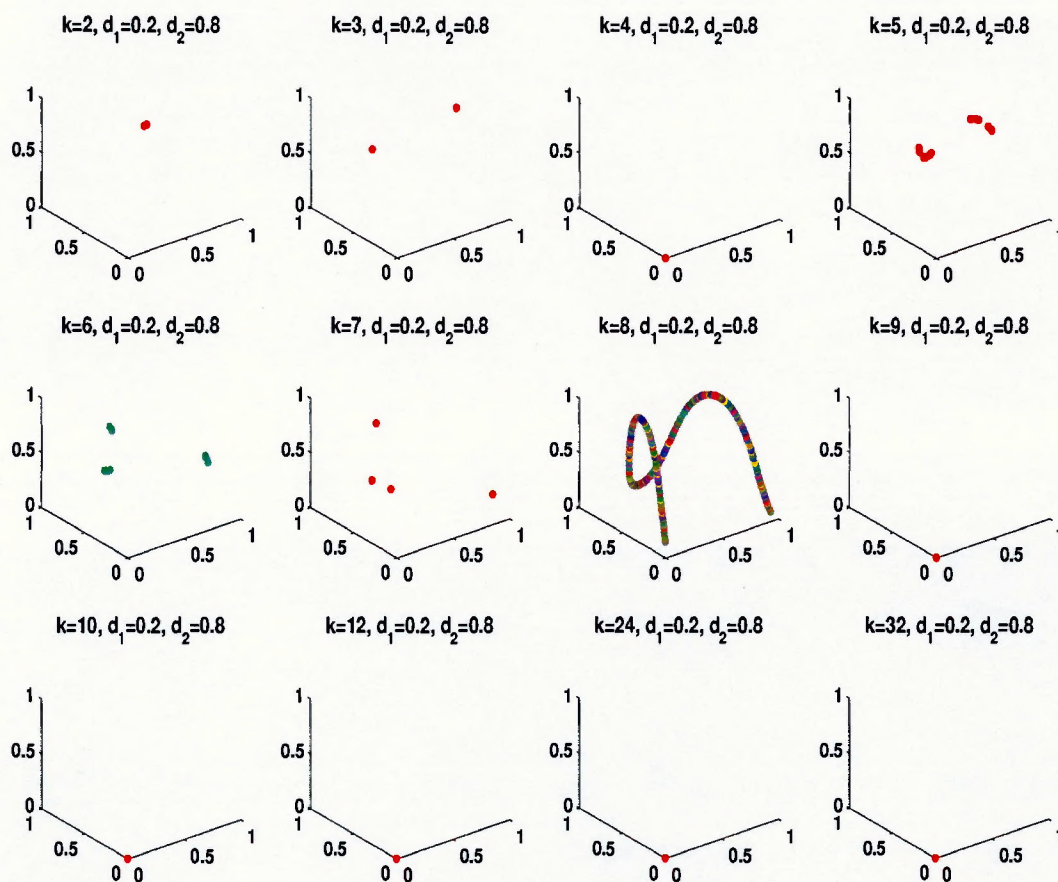


Figure 3.7: Three-Dimensional Delay Plots for a series of 1000 consecutive time points for $d_1 = 0.2$, $d_2 = 0.8$. The graphs are in the unit cube $[0; 1] \times [0; 1] \times [0; 1]$. The solid shades induced by a coloring procedure suggest ordered attractors for some values of $k = 2, 3, 4, 5, 6, 7, 9, 10, 12, 24, 32$ and varied colors suggest chaotic attractors for $k = 8$. This agrees with the bifurcation diagram in Figure 3.1.

Chapter 4

Conclusions

The original motivation for the probability density rule (7) is that it generalizes the elementary cellular automata Rule 22. This rule says it is turned ON if and only if its precursors have a single on node. If a more generalized one-dimensional cellular automata is set up in the same manner, whether k (constant) is fixed or k nearest neighbors are ON or OFF, then its easy to show that its density function either approaches a limiting value or a periodic orbit. For example, this behavior is exhibited in Figures 3.6 and 3.7 for random Boolean networks. We show that if a network is operating under a generalized Rule 22 and considering the number of nodes per precursor, then the rest of the network demonstrates a simplified behavior.

This study generalizes a cellular automata model proposed by Andrecut [4] for a Boolean network with a unique Boolean rule for all nodes. By using the mean field approximation, Andrecut demonstrates excellent agreement between the map model dynamics and the real Boolean network dynamics by considering only the first

iteration. This work allows for fixed and variable number of parents for each node as well as analysis of the dynamics over time. An algorithm for simulation of the model is introduced and simulation results show that for fixed connectivity with randomly assigned parents the model fits the real Boolean system with more frequency than for k nearest neighbors.

However, when the synchronous updating schemes are run for variable k with the different parent types, the model did not fit the real Boolean system with as much success as previously reported by Andrecut. The reason for this is that the model used in this paper is obtained through a mean-field approach in which parent nodes are assumed to act independently. In reality there are correlations between the parents and therefore there can be a build up of correlations over time. This aspect is not captured in the model. Our results are further confirmed by the results of the Lyapunov exponents, bifurcation diagrams, and delay plots.

Chapter 5

Directions for Further Investigation

The next phase of this research will be analyzing asynchronous updates of the ECA Rule 22, since this is of importance in modeling systems composed of multiple interacting components. Along with this, generalizing this model to allow for multiple Boolean rules to be used in consecutive states of the system would be of interest. This would provide a much more realistic model for biological cellular networks whose update schemes are dependent upon various protein interactions ([23], [27]). The approach to this would be to change the unique Boolean rule to be used based on other cellular automata rules [50] which could lead to interesting new models and dynamic behaviors.

Other work is being conducted by introducing "noise" in the system to determine the stability of the system to perturbations ([19], [30]). In scale free networks perturbing a very highly connected node is expected to have a much greater impact

than perturbing a node with low connectivity. The work by Goodrich et al. [19] has shown that introducing noise into a system using ECA Rule 126 actually demonstrates a stabilizing effect on the dynamics of the system. This is done by introducing a perturbation into the system by changing the value of a node. Additional nodes may be altered to see if the system stabilizes or becomes chaotic. Thus, it would be of interest to further this work with ECA Rule 22 for both synchronous and asynchronous update schemes. A natural step would be to consider one-step or multiple-step correlations between the nodes. This would provide a more realistic approach especially for parameter values that did not yield a good match of the model and the real network in this study.

Also, taking into account the topology of the network as opposed to randomly selecting the parent nodes is another avenue of study. As observed by Marr and Hütt ([31], [32]), it is known that the variation of topology of the network from random to regular or scale-free network has a clear impact on pattern formation in binary cellular automata. Embedding the topology in the network model could lead to interesting results regarding the effect of topology on the dynamics of the system.

In the case of statistical analysis, it may be possible to obtain more accurate significance using logistic regression. This technique involves binary dependent variables wherein no assumption about the distribution of independent variables is made. The

goal of logistic regression is to correctly predict the outcome for individual cases using the most parsimonious model. This may work if it is assumed that distance or connectivity is assumed independent ([8], [10]). It may also be helpful to run trend analyses to potentially predict when the model and system will match.

There are many possibilities for further study of this model and its behavior under different conditions.

Bibliography

- [1] Aldana M., Cluzel P., *A Natural Class of Robust Networks*, PNAS 100 (2003), p. 8710-8714.
- [2] Alligood K.T., Sauer T.D., Yorke J.A., *Chaos: An Introduction to Dynamical Systems*, Springer-Verlag (1996).
- [3] Anderson R.J., *Characterization of Performance, Robustness, and Behavior Relationships in a Directly Connected Material Handling System*, Doctor of Philosophy Dissertation, <http://scholar.lib.vt.edu/theses/available/etd-04182006-130843/unrestricted/Intro-to-Ch6-M2.pdf>, 2006.
- [4] Andrecut M., *Mean Field Dynamics of Random Boolean Networks*, J. Stat. Mech., P02003 (2005).
- [5] Andrecut M., Ali M.K., *Chaos in a Simple Boolean Network*, Internatl. J. Mod. Phys., 15 (2001), p. 17-23.

- [6] Bagley R.J., Glass L., *Counting and Classifying Attractors in High Dimensional Dynamical Systems* J. Theor. Biol., 183 (1996), p. 269-284.
- [7] Bornholdt S., Rohlf T., *Topological Evolution fo Dynamical Networks: Global Criticality from Local Dynamics*, Phys. Rvw. Letters, 84 (2000), p. 6114-6117.
- [8] Brannic M.T., *Logistic Regression*, <http://luna.cas.usf.edu/mbrannic/files/regression/Logistic.html>.
- [9] Burratson D., *Variety, Pattern and Isomorphism*, Proceedings of the Third Iteration Conference, Monash University, 2005.
- [10] Connor E.F., *Logistic Regression*, <http://online.sfsu.edu/efc/classes/biol710/logistic/logisticreg.htm>.
- [11] Coutinho R., Fernandez B., Lima R., Meyroneine A., *Discrete Time Piecewise Affine Models of Genetic Regulatory Networks*, J. Math. Biol. 52 (2006), p. 524-570.
- [12] Deng X., Geng H., Matache M.T., *Dynamics of Asynchronous Random Boolean Networks with Asynchrony Generated by Stochastic Processes*, submitted, http://www.unomaha.edu/~dmatache/Papers/ARBN_stochproc.pdf.
- [13] Dougherty E.R., Kim S., Chen Y., *Coefficient of Determination in Nonlinear Signaling Processing*, Signal Processing, Vol. 80, 10 (2000), p. 2219-2235.

- [14] Gershenson C., *Classification of Random Boolean Networks*, In: Standish R.K., Bedeau M.A., Abbas H.A. (eds.), *Artificial Life VIII, The 8th International Conference on the Simulation and Synthesis of Living Systems* (2002), p. 1-8.
- [15] Gershenson C., *Introduction to Random Boolean Networks*, In: Bedeau M., Husbands P., Hutton T., Kumar S., Suzuki H. (eds.) *Workshop and Tutorial Proceedings, Ninth International Conference on the Simulation and Synthesis of Living Systems (ALife IX)* (2004) p. 160-173.
- [16] Gershenson C., Broekaert J., Aerts D., *Contextual Random Boolean Networks*, In: Banzhaf W., Christaller T., Dittrich P., Kim J.T., Ziegler J. (eds.), *Advances in Artificial Life, 7th European Conference, ECAL 2003, Dortmund, Germany* (2003), p. 615-624.
- [17] Gershenson C., Kauffman S.A., Shmulevich I., *The Role of Redundancy in the Robustness of Random Boolean Networks*, In: Rocha L.M., Yaeger L.S., Bedeau M.A., Floreano D., Goldstone R.L., Vespignani A. (eds.), *Artificial Life X, Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems* (2006), p. 35-42.
- [18] Goodman R.A., *Random Boolean Networks with the Number of Parents Generated by Certain Probability Distributions*, Presented

at the 2006 University of Nebraska at Omaha Mathematics Symposiums, <http://www.unomaha.edu/wwwmath/minigrants/2005-2006/RayGoodmanReport.pdf>, 2006.

- [19] Goodrich C.S., Matache M.T., *The Stabilizing Effect of Noise on the Dynamics of a Boolean Network*, Submitted.
- [20] Gutowitz H.A., *A Heirarchical Classification of Cellular Automata*, Physica D. 45 (1990), p. 136-156.
- [21] Hallinan J., Wiles J., *Evolving Genetic Regulatory Networks Using an Artificial Genome*, In: Chen Y.P.P. (ed.), Second Asia-Pacific Bioinformatics Conference (APBC2004), Volume 29 of CRPIT, Dunedin, New Zealand (2004), p. 291-296.
- [22] Harvey I., Bossomaier T., *Time Out of Joint: Attactors in Asynchronous Random Boolean Networks*, Proceedings of the Fourth European Conference on Artificial Life (ECAL97), MIT Press, 1997, p. 65-75.
- [23] Heidel J., Maloney J., Farrow C., Rogers J.A., *Finding Cycles in Synchronous Boolean Networks with Applications to Biochemical Systems*, Intl. J. Bifurcation Chaos Appl. Sci. Eng. 13 (2003) 535-552.
- [24] Hung Y-C., Ho M-C., Lih J-S., Jiang I-M., *Chaos Synchronization of Two Stochastically Coupled Random Boolean Networks*, Physics Letters A, 356, 2006, p. 35-43.

- [25] Kauffman S., *Complexity and Genetic Networks*, Exystence Project News (2003), <http://sandi.soc.surrey.ac.uk/>.
- [26] Kauffman S.A., *Self-Organization and Adaptation in Complex Systems*, In: *The Origins of Order: Self-Organization and Selection in Evolution*, Oxford University Press, New York (1993), p. 173-235.
- [27] Klemm K., Bornholdt S., *Topology of Biological Networks and Reliability of Information Processing*, PNAS, 102 (2005), p. 18414-18419.
- [28] Kurz M.J., Stergiou N., *An Artificial Neural Network that Utilizes Hip Joint Actuations to Control Bifurcations and Chaos in a Passive Dynamic Bipedal Walking Model*, Biol. Cybern. 93 (2005), p. 213-221.
- [29] Lindgren K., *Cellular Automata*, Lecture notes published on <http://frit.fy.chalmers/se/cs/cas/courses/infortheory/pdfs/ITCS-06.pdf> (2003).
- [30] Mandell A.J., Selz K.A., *Nonlinear Dynamical Patterns as Personality Theory for Neurobiology and Psychiatry*, Psych., 58 (1995), p. 371-390.
- [31] Marr C., Hütt M-T., *Topology Regulates Pattern Formation Capacity of Binary Cellular Automata on Graphs*, Physica A, 354 (2005), p. 641662.
- [32] Marr C., Hütt M-T., *Similar Impact of Topological and Dynamic Noise on Complex Patterns*, http://arxiv.org/PS_cache/cond-mat/pdf/0509/0509199.pdf.

- [33] Matache M.T., Heidel J., *Probabilistic Boolean Networks Under Legalistic ECA Rules*, in preparation.
- [34] Matache M.T., Heidel J., *Random Boolean Network Model Exhibiting Deterministic Chaos*, Phys. Rev. E 69, 056214, 2004, 10 pages.
- [35] Matache M.T., Heidel J., *Asynchronous Random Boolean Network Model Based on Elementary Cellular Automata Rule 126*, Phys. Rev. E 71, 026232 (2005), 13 pages.
- [36] Matache M.T., *Asynchronous Random Boolean Network Model with Variable Number of Parents based on Elementary Cellular Automata Rule 126*, IJMPB 20 (2006) p. 897-923.
- [37] McIntosh H.V., *Linear Cellular Automata*, <http://delta.cs.cinvestav.mx/~mcintosh/oldweb/lcau/lcau.html>, 1990.
- [38] Mesot B., Teuscher C., *Deducing Local Rules for Solving Global Tasks with Random Boolean Networks*, Physica D. 211 (2005), p. 88-106.
- [39] Miller I., Miller M., *John E. Freund's Mathematical Statistics, 6th Edition*, Prentice-Hall, New Jersey (1999), p. 438-441.
- [40] Öktem H., Pearson R., Yli-Harja O., Nicorici D., Egiazarian K., Astola J. *A Computational Model for Simulating Continuous Time Boolean Networks*,

Proceedings from the Workshop on Genomic Signal Processing and Statistics,
<http://www.gensips.gatech.edu/proceedings/Contributed/CP2-11.pdf>, 2002.

- [41] Pezard L., Nandrino J.L., *Dynamic Paradigm in Psychopathology: "Chaos Theory", from Physics to Psychiatry*, *Encephale*, 27 (2001), p. 260-268.
- [42] Rohlf T., Bornholdt S., *Self-Organized Pattern Formation and Noise-Induced Control Based on Particle Computations*, *J. Stat. Mech.* (2005) L12001.
- [43] Rohlfshagen P., DiPaolo E.A., *The Circular Topology of Rhythm in Asynchronous Random Boolean Networks*, *Biosystems* 73 (2004), p. 141-152.
- [44] Sanchez J.R., Lopez-Ruiz R., *Detecting Synchronization in Spatially Extended Discrete Systems by Complexity Measurements*, *Discrete Dynamics Nat. Soc.* 3 (2005) 337-342.
- [45] Shmulevich I., Dougherty E.R., Kim S., Zhang W., *Probabilistic Boolean Networks: A Rule-Based Uncertainty Model for Gene Regulatory Networks*, *Bioinformatics*, Vol. 18, 2 (2002), p. 261-274.
- [46] Shmulevich I., Dougherty E.R., Zhang W., *From Boolean to Probabilistic Boolean Networks as Models for Genetic Regulatory Networks*, *Proceedings of the IEEE*, Vol. 90, 11 (2002), p. 1778-1792.

- [47] Thompson M.J., *Use of Cellular Automata Models to Examine Complexity of Organizational Behaviors*, Masters Thesis, University of Weston, Sidney, 2005.
- [48] von Neumann J., *The Theory of Self-Reproducing Automata*. University of Illinois Press, 1966.
- [49] Voorhees B., *Emergency and Induction of Cellular Automata Rules via Probabilistic Reinforcement Paradigms*, Complexity 11 (2006) p. 44-57.
- [50] Wolfram S., *A New Kind of Science*, Wolfram Media, Champaign, 2002.
- [51] Wolfram S., *Cellular Automata as Simple Self-Organizing Systems*, [http: www.stephenwolfram.com/publications/articles/ca/82-cellular/1/text.html](http://www.stephenwolfram.com/publications/articles/ca/82-cellular/1/text.html), 1982.
- [52] Wolfram S., *Computation Theory of Cellular Automata*, Commun. Math. Phys. 96 (1984) p. 15-57.
- [53] Wuensche A., *Basins of Attraction in Network Dynamics: A Conceptual Framework for Biomolecular Networks*, In Schlosser G., Wagner G.P. (eds), Chicago University Press, Chicago, 2004, pp. 288-314.

Appendices

Appendix A

Fixed Connectivity Matlab Code

The following code is used to generate iterations of the model and system for fixed connectivity.

```
clear;clf;

N = 128; kparam = [8 10 16 22 32];

d1 = [0.2 0.3 0.4];

d2 = 1*ones(1,length(d1));

IT = 5;

fontsize=14;

P1,1 = [1];

M = N;

numpointsonxaxis=N;
```

```

initialproportions = linspace(0,1,numpointsonxaxis);

initial values for iterations xt = N;

numplot=1; for kindex = 1:length(kparam)

k = kparam(kindex);

KK = num2str(k);

clear pa

pa = parentsrandom(N,k,M); or pa = parentsCA(N,k,M);

for dindex = 1:length(d1) d1param = d1(dindex);

d2param = d2(dindex);

D2 = num2str(d2param);

flag = 0;

if d1param == 0

if d2param ~ = 1

R1,1=[0 ones(1,floor((k+1)*d2param)) zeros(1, k - floor((k+1)*d2param))];

R1,2=[ones(1,floor((k+1)*d2param)) zeros(1,k-floor((k+1)*d2param)) 0];

else

R1,1=[0 ones(1,k(i))];

R1,2=[ones(1,k(i)) 0];

end

flag = 1;

```

```

end

if d2param == 1

R1,1=[zeros(1,floor((k+1)*d1param)+1) ones(1, k - floor((k+1)*d1param))];
R1,2=[zeros(1,floor((k+1)*d1param)) ones(1,k - floor((k+1)*d1param)) 0];

flag = 1;

end

if flag == 0;

R1,1=[0 zeros(1,floor((k+1)*d1param)) ones(1,floor((k+1)*d2param)-
floor((k+1)*d1param)) zeros(1, k-floor((k+1)*d2param))];
R1,2=[zeros(1,floor((k+1)*d1param)) ones(1,floor((k+1)*d2param) -
floor((k+1)*d1param)) zeros(1,k-floor((k+1)*d2param)) 0];

end

clear v prob

steps = IT+1;

v = cell(steps,N);

prob = cell(1,steps);

L=1;

for i = 1:length(initialproportions)

for m=1:N

u = rand;

```

```
if u <= initialproportions(i)
v1,L(m) = 0;
else
v1,L(m) = 1;
end
end
prob1,1(L) = sum(v1,L);
for step = 1:IT+1
vstep+1,L = oneiteration(vstep,L,xt,k,P,R,pa);
prob1,step+1(L) = sum(vstep+1,L);
end
L=L+1;
end
p = sort(prob1,1/N);
p1 = pbnmodel(N,p,xt,k,M,P,R);
saveiterate(1,1:length(p1)) = p1;
for it=2:IT+1
p1 = pbnmodel(N,p1,xt,k,M,P,R);
end
saveiterate = p1;
```

```
subplot(length(kparam),length(d1),numplot);  
plot(p,saveiterate,'r','LineWidth',1);  
hold;  
plot(prob1,1/N,prob1,IT+2/N,'.','MarkerSize',5);  
axis([0 1 0 1]);  
axis off;  
hold off;  
numplot = numplot+1;  
end  
end
```

Appendix B

Variable Connectivity Matlab

Code

The following code is used to generate iterations of the model and system for variable connectivity.

```
clear;clf;
```

```
N = 128;
```

```
k = [10 32];
```

```
Mvalues = N/8:N/16:7*N/8;
```

```
D1,1 = [0];
```

```
D1,2 = [0.5];
```

```
D2,1 = [0.5];
```

```
D2,2 = [1];  
rule1bound = [1 1];  
P1,1=[1];  
P2,1=[1];  
IT = 1;  
numpointsonxaxis = N;  
initialproportions = linspace(0,1,numpointsonxaxis);  
xt = N;  
[pline, pcol] = size(P);  
iterate = num2str(IT);  
fsize=14;  
numplot=1;  
subplotline = ceil(length(Mvalues)/2);  
subplotcol = 2;  
for mindex = 1:length(Mvalues)  
M = [Mvalues(mindex) N-Mvalues(mindex)];  
pa = parentsrandom(N,k,M);  
pa = parentsCA(N,k,M);  
for i=1:pline  
if rule1bound(i) > 0
```



```

for j = 1:rule1bound(i)

flag = 0;

if Di,1(j) == 0

if Di,2(j) ~= 1

Ri,1(j,:)=[0 ones(1,floor((k(i)+1)*Di,2(j))) zeros(1, k(i) -
floor((k(i)+1)*Di,2(j)))];

Ri,2(j,:)=[ones(1,floor((k(i)+1)*Di,2(j))) zeros(1,k(i)-
floor((k(i)+1)*Di,2(j))) 0];

else

Ri,1(j,:) = [0 ones(1,k(i))];

Ri,2(j,:) = [ones(1,k(i)) 0];

end

flag = 1;

end

if Di,2(j) == 1

Ri,1(j,:)=[0 zeros(1,floor((k(i)+1)*Di,1(j))) ones(1, k(i) - floor((k(i)+1)*Di,1(j)))];

Ri,2(j,:)=[zeros(1,floor((k(i)+1)*Di,1(j))) ones(1,k(i) - floor((k(i)+1)*Di,1(j))) 0];

flag = 1;

end

if flag == 0;

```

```

Ri,1(j,:)= [0 zeros(1,floor((k(i)+1)*Di,1(j))) ones(1,floor((k(i)+1)*Di,2(j))-
floor((k(i)+1)*Di,1(j))) zeros(1, k(i)-floor((k(i)+1)*Di,2(j)))]];
Ri,2(j,:)= [zeros(1,floor((k(i)+1)*Di,1(j))) ones(1,floor((k(i)+1)*Di,2(j)) -
floor((k(i)+1)*Di,1(j))) zeros(1,k(i)-floor((k(i)+1)*Di,2(j))) 0];
end; end; end

if rule1bound(i) < length(Pi,1)
for j = rule1bound(i)+1:length(Pi,1)
Ri,1(j,:) = [zeros(1,floor((k(i)+1)*Di,2(j))+1) ones(1,k(i)-floor((k(i)+1)*Di,2(j)))]];
Ri,2(j,:) = [ones(1,floor((k(i)+1)*Di,2(j))) zeros(1,k(i)-floor((k(i)+1)*Di,2(j))+1)]];
end; end; end

clear v prob

steps = IT+1;

v = cell(steps,N);

prob = cell(1,steps);

L=1; for i = 1:length(initialproportions)

for m=1:N

u = rand;

if u <= initialproportions(i)

v1,L(m) = 0;

else

```

```
v1,L(m) = 1;

end

end

prob1,1(L) = sum(v1,L);

for step = 1:IT+1

vstep+1,L = oneiteration(vstep,L,xt,k,P,R,pa);

prob1,step+1(L) = sum(vstep+1,L);

end

L=L+1;

end

p = sort(prob1,1/N);

saveiterate(1,1:length(p)) = p;

p1 = pbnmodel(N,p,xt,k,M,P,R);

saveiterate(2,1:length(p1)) = p1;

for it=3:IT+1

p1 = pbnmodel(N,p1,xt,k,M,P,R);

saveiterate(it,1:length(p1)) = p1;

end

subplot(subplotline,subplotcol,numplot);

plot(p,saveiterate(IT+1,:), 'r');
```

```
hold;  
plot(prob1,1/N,prob1,IT+1/N,');  
axis([0 1 0 1]);  
numplot = numplot+1;  
hold off;  
axis off;  
display(M);  
end
```

Appendix C

Additional Codes for Model Simulations

C.1 Randomly Assigned Parents

This is the code for randomly assigned parents, which is labeled `parentsrandom` in the code for both fixed and variable connectivity.

```
function pa = parentsrandom(N,k,M)
pa = cell(1,N); for n = 1:M(1)
u = randperm(N);
pa{1,n} = u(1:k(1));
for i=1:k(1)
```

```
if pa1,n(i) == n;
pa1,n(i) = u(k(1)+1);
end
end
pa1,n = sort(pa1,n);
end if length(M) > 1
for j = 1:length(M)-1
for n = M(j)+1:M(j)+M(j+1)
u = randperm(N);
pa1,n = u(1:k(j+1));
for i=1:k(j+1)
if pa1,n(i) == n;
pa1,n(i) = u(k(j+1)+1);
end
end
pa1,n = sort(pa1,n);
end; end; end
```

C.2 k Nearest Neighbors Parents

This is the code for nearest neighbors assigned parents, which is labeled `parentca` in the code for both fixed and variable connectivity.

```
function pa = parentsCA(N,k,M)
pa = cell(1,N); u = [1:N 1:N 1:N]; for n = 1:M(1)
pa1,n = sort([u(n+N-floor(k(1)/2):n+N-1) u(n+N+1:n+N+k(1)-floor(k(1)/2))]);
end if length(M) > 1
for j = 1:length(M)-1
for n = M(j)+1:M(j)+M(j+1)
pa1,n = sort([u(n+N-floor(k(1)/2):n+N-1) u(n+N+1:n+N+k(1)-floor(k(1)/2))]);
end; end; end
```

C.3 Probability Model Code

This is the code for the probability model labeled `pbnmodel`.

```
function poftplus1 = pbnmodel(N, p, xt, K, M, P, R)

[J,col] = size(R); clear fk for i=1:length(p)

for j=1:J
```

```

clear binomial

for s = 0:K(j)
    binomial(s + 1) = nchoosek(K(j), s) * p(i)^s * (1 - p(i))^(K(j)-s);
end

B = binomial;

[linephicolphi] = size(Rj,1);

if linephi > 1
    for count = 1:linephi-1
        binomial = [binomial; B];
    end
end

fk(j) = sum(sum((binomial .* ((1 - p(i)) * Rj, 1 + p(i) * Rj, 2))'). * Pj, 1);

end

pofplus1(i) = p(i) * (1 - xt/N) + xt/N^2 * sum(M. * fk); end

```


C.4 Matlab Code for Randomly Selecting a Single Rule

This is the code for a single rule for randomly assigned parents, which is labeled `oneiteration` in the code.

```
function vtplus1 = oneiteration(v,xt,K,P,R,pa)
format long for jindex=1:length(K)
Prob = cumsum(Pjindex,1);
z = rand;
m=1;
while z > Prob(m)
m = m+1;
end
rulechoice(jindex) = m;
end U = randperm(length(v)); vupdated = sort(U(1:xt));
vunchanged = sort(U(xt+1:length(U)));
vtplus1(vunchanged) = v(vunchanged); for index=1:length(vupdated)
s = sum(v(pa1,vupdated(index)));
classnumber = length(pa1,vupdated(index));
jindex = 1;
```

```
while classnumber  $\sim$  K(jindex)
jindex = jindex+1;
end
if v(vupdated(index)) == 0
vplus1(vupdated(index)) = Rjindex,1(rulechoice(jindex),s+1);
else vplus1(vupdated(index)) = Rjindex,2(rulechoice(jindex),s+1);
end; end
```

Appendix D

Lyapunov Exponent and Bifurcation Diagram Matlab Code

D.1 Lyapunov Exponents and Bifurcation Diagrams

The following codes are used to generate Lyapunov exponent and bifurcation diagrams for fixed connectivity.

```
clear; clf;
```

```
N = 128; d1 = [0.2 0.3]; d2 = [0.8 0.8];
```

```
P1,1 = [1];
```

```
M = N;
```

```
numpointsonxaxis=N;
```

```
xt = N;

Numberiterations = 29;

initialpoint = 0.2;

Mink = 1;

Maxk = 12;

stepk = 1;

fsize=14;

linew=1;

numlines = 2; numcolumns = 2;

markerpoints=5;

numplot = 1;

for dindex = 1:length(d1)

d1param = d1(dindex);

d2param = d2(dindex);

D1 = num2str(d1param);

D2 = num2str(d2param);

step = 1;

for k = Mink:stepk:Maxk

flag = 0;

if d1param == 0
```

```

if d2param ~= 1
R1,1=[0 ones(1,floor((k+1)*d2param)) zeros(1, k - floor((k+1)*d2param))];
R1,2=[ones(1,floor((k+1)*d2param)) zeros(1,k-floor((k+1)*d2param)) 0];
else
R1,1=[0 ones(1,k(i))];
R1,2=[ones(1,k(i)) 0];
end

flag = 1;

end

if d2param == 1
R1,1=[zeros(1,floor((k+1)*d1param)+1) ones(1, k - floor((k+1)*d1param))];
R1,2=[zeros(1,floor((k+1)*d1param)) ones(1,k - floor((k+1)*d1param)) 0];

flag = 1;

end

if flag == 0;
R1,1=[0 zeros(1,floor((k+1)*d1param)) ones(1,floor((k+1)*d2param)-
floor((k+1)*d1param)) zeros(1, k-floor((k+1)*d2param))];
R1,2=[zeros(1,floor((k+1)*d1param)) ones(1,floor((k+1)*d2param) -
floor((k+1)*d1param)) zeros(1,k-floor((k+1)*d2param)) 0];
end

```

```

f = initialpoint;

h=f;

hprime = pbnmodelderivative(N, h, xt, k, M, P, R);

hh = log(abs(hprime));

for i=2:Numberiterations

h = pbnmodel(N, h, xt, k, M, P, R);

hprime = pbnmodelderivative(N, h, xt, k, M, P, R);

hh = hh + log(abs(hprime));

end

K(step) = k;

Hh(step) = hh/Numberiterations;

step = step+1;

end

subplot(numlines,numcolumns,numplot);

plot(K,Hh,'LineWidth',linewidth+1) hold on x = linspace(Mink,Maxk,100);

plot(x, zeros(size(x)), 'r-', 'LineWidth', linewidth);

hold off; xlabel('k', 'FontWeight', 'bold', 'FontSize', fsize);

ylabel('LyE', 'FontWeight', 'bold', 'FontSize', fsize);

title(['d1=', D1, ' d2=', D2], 'FontWeight', 'bold', 'FontSize', fsize) axis([0 Maxk -40 5]);

numplot = numplot + 1;

```

```

end

for dindex = 1:length(d1)

d1param = d1(dindex);

d2param = d2(dindex);

D1 = num2str(d1param);

D2 = num2str(d2param);

display('Starting bifurcations'); display(['d1=',D1,' d2=',D2]);

numpointsp = 100;

iterations = 1000;

toplot = 5;

p1 = linspace(0,1,numpointsp);

for k = Mink:stepk:Maxk

flag = 0;

if d1param == 0

if d2param ~ = 1

R1,1=[0 ones(1,floor((k+1)*d2param)) zeros(1, k - floor((k+1)*d2param))];

R1,2=[ones(1,floor((k+1)*d2param)) zeros(1,k-floor((k+1)*d2param)) 0];

else

R1,1=[0 ones(1,k(i))];

R1,2=[ones(1,k(i)) 0];

```

```

end

flag = 1;

end

if d2param == 1

R1,1=[zeros(1,floor((k+1)*d1param)+1) ones(1, k - floor((k+1)*d1param))];
R1,2=[zeros(1,floor((k+1)*d1param)) ones(1,k - floor((k+1)*d1param)) 0];

flag = 1;

end

if flag == 0;

R1,1=[0 zeros(1,floor((k+1)*d1param)) ones(1,floor((k+1)*d2param)-
floor((k+1)*d1param)) zeros(1, k-floor((k+1)*d2param))];
R1,2=[zeros(1,floor((k+1)*d1param)) ones(1,floor((k+1)*d2param) -
floor((k+1)*d1param)) zeros(1,k-floor((k+1)*d2param)) 0];

end

subplot(numlines,numcolumns,numplot);

ph = p1;

for j = 1:iterations

ph = pbnmodel(N, ph, xt, k, M, P, R);

end

Ah = k*ones(size(ph));

```



```

for j = 1:toplot
ph = pbnmodel(N, ph, xt, k, M, P, R);
plot(Ah,ph, '.', 'MarkerSize',markerpoints);
hold on;
end
end
xlabel('k', 'FontWeight', 'bold', 'FontSize', fsize);
ylabel('p(t)', 'FontWeight', 'bold', 'FontSize', fsize);
title(['d1=', D1, ' d2=', D2], 'FontWeight', 'bold', 'FontSize', fsize) axis([0 Maxk 0 1]);
hold off; numplot = numplot + 1;
end

```

D.2 Derivative Function Code for Lyapunov Exponents

This code is for the derivative function used to calculate the Lyapunov exponents.

```
function poftplus1derivative = pbnmodelderivative(N, p, xt, K, M, P, R)
```

```
    r = length(R1,1); [J,col] = size(R); clear fk for i=1:length(p)
```

```
for j=1:J
```

```

clear binomial1 binomial2

if K(j) == 1

fk(j) = sum(sum((-Rj,1(1) + Rj,2(1)) * (1 - p(i))^(K(j)) -
((1 - p(i)) * Rj,1(1) + p(i) * Rj,2(1)) * K(j) * (1 - p(i))^(K(j)-1) + ...(-Rj,1(r) +
Rj,2(r)) * p(i)^(K(j)) + ((1 - p(i))
* Rj,1(r) + p(i) * Rj,2(r)) * K(j) * p(i)^(K(j)-1))'. * Pj,1);

else

for s = 1:K(j)-1

binomial1(s) = nchoosek(K(j), s) * p(i)^s * (1 - p(i))^(K(j)-s);

binomial2(s) = nchoosek(K(j), s) * (s * p(i)^(s-1)) * (1 - p(i))^(K(j)-s) - p(i)^s * (K(j) -
s) * (1 - p(i))^(K(j)-s-1);

end

B1 = binomial1;

B2 = binomial2;

[linephicolphi] = size(Rj,1);

if linephi > 1

for count = 1:linephi-1

binomial2 = [binomial2; B2];

end

end

```

```

fk(j) = sum(sum((-Rj,1(1) + Rj,2(1)) * (1 - p(i))^(K(j)) -
((1 - p(i)) * Rj,1(1) + p(i) * Rj,2(1)) * K(j) * (1 - p(i))^(K(j)-1) + ...
binomial1. * (-Rj,1(2 : r - 1) + Rj,2(2 : r - 1))
+ binomial2. * ((1 - p(i)) * Rj,1(2 : r - 1) + p(i) * Rj,2(2 : r - 1)) + ...
(-Rj,1(r) + Rj,2(r)) * p(i)^(K(j)) +
((1 - p(i)) * Rj,1(r) + p(i) * Rj,2(r)) * K(j) * p(i)^(K(j)-1)') . * Pj,1);
end
end
poftplus1derivative(i) = (1-xt/N)+xt/N^2*sum(M.*fk); end

```

Appendix E

Fixed Points

The following code is used to generate fixed points plots for fixed connectivity.

```
clear;
kbound=12;
N = 128;
kparam = [2 3 4 5 6 7 8 9 10];
d1 = [0.1 0.3 0.4 0.5 0.6 0.7 0.8 0.9];
d2 = [0.4 1 0.9 1 0.8 0.8 0.9 1];
numcolumns = 2;
numlines = length(d1)/numcolumns;
P1,1 = [1];
M = N;
```

```

numpointsonxaxis=N;

initialproportions = linspace(0,1,numpointsonxaxis);

xt = N;

fsize=14; marker=5; linewidth=3; numplot=1;

clf;

for dindex = 1:length(d1)

d1param = d1(dindex);

d2param = d2(dindex);

D1 = num2str(d1param);

D2 = num2str(d2param);

for k = kparam

KK = num2str(k);

flag = 0;

if d1param == 0

if d2param ~= 1

R1,1=[0 ones(1,floor((k+1)*d2param)) zeros(1, k - floor((k+1)*d2param))];

R1,2=[ones(1,floor((k+1)*d2param)) zeros(1,k-floor((k+1)*d2param)) 0];

else

R1,1=[0 ones(1,k(i))];

R1,2=[ones(1,k(i)) 0];

```

```

end

flag = 1;

end

if d2param == 1

R1,1=[zeros(1,floor((k+1)*d1param)+1) ones(1, k - floor((k+1)*d1param))];
R1,2=[zeros(1,floor((k+1)*d1param)) ones(1,k - floor((k+1)*d1param)) 0];

flag = 1;

end

if flag == 0;

R1,1=[0 zeros(1,floor((k+1)*d1param)) ones(1,floor((k+1)*d2param)-
floor((k+1)*d1param)) zeros(1, k-floor((k+1)*d2param))];
R1,2=[zeros(1,floor((k+1)*d1param)) ones(1,floor((k+1)*d2param) -
floor((k+1)*d1param)) zeros(1,k-floor((k+1)*d2param)) 0];

end

clear f x p

syms f x

f = pbnmodel(N,x,xt,k,M,P,R)-x;

x=solve(f);

subplot(numlines,numcolumns,numplot);

for i=1:length(x)

```

```
p=subs(x(i));  
if imag(p) == 0 && real(p) >= 0 && real(p) <= 1  
plot(k,p,'*', 'MarkerSize',marker);  
hold on;  
end  
end  
end  
axis([0 max(k) 0 1])  
xlabel('k','FontSize',fsize, 'FontWeight','bold');  
ylabel('p','FontSize',fsize, 'FontWeight','bold');  
title(['d1 = ', D1, ' d2 = ', D2],'FontSize',fsize, 'FontWeight','bold');  
numplot = numplot+1;  
end
```

Appendix F

Delay Plots

The following code is used to generate delay plots for fixed connectivity.

```
clear;clf;
N = 128;
kvector = [2 3 4 5 6 7 8 9 10];
d1 = [0.3];
d2 = [1];
numberlinesgraph = length(kvector)/3;
numbercolumnsgraph = length(d1)*3;
p0 = 0.1;
iterations1=1000;
iterations2=100;
```



```

P1,1 = [1];

M = N;

numpointsonxaxis=N;

initialproportions = linspace(0,1,numpointsonxaxis);

xt = N;

marker = 14;

fsize=12;

    numberplot = 1; for kindex = 1:length(kvector)

k = kvector(kindex);

KK = num2str(k);

for dindex = 1:length(d1)

d1param = d1(dindex);

d2param = d2(dindex);

D1 = num2str(d1param);

D2 = num2str(d2param);

flag = 0;

if d1param == 0

if d2param ~= 1

R1,1=[0 ones(1,floor((k+1)*d2param)) zeros(1, k - floor((k+1)*d2param))];

```

```

R1,2=[ones(1,floor((k+1)*d2param)) zeros(1,k-floor((k+1)*d2param)) 0];

else

R1,1=[0 ones(1,k(i))];

R1,2=[ones(1,k(i)) 0];

end

flag = 1;

end

if d2param == 1

R1,1=[zeros(1,floor((k+1)*d1param)+1) ones(1, k - floor((k+1)*d1param))];

R1,2=[zeros(1,floor((k+1)*d1param)) ones(1,k - floor((k+1)*d1param)) 0];

flag = 1;

end

if flag == 0;

R1,1=[0 zeros(1,floor((k+1)*d1param)) ones(1,floor((k+1)*d2param)-

floor((k+1)*d1param)) zeros(1, k-floor((k+1)*d2param))];

R1,2=[zeros(1,floor((k+1)*d1param)) ones(1,floor((k+1)*d2param) -

floor((k+1)*d1param)) zeros(1,k-floor((k+1)*d2param)) 0];

end

clear p

p1 = p0;

```

```

for i = 1:iterations1

p1 = pbnmodel(N,p1,xt,k,M,P,R);

end

for i = 1:iterations2

p1 = pbnmodel(N,p1,xt,k,M,P,R);

P1(i) = p1;

end

LastPoint=iterations2;

colormap('hsv')

PrettyColors= colormap;

subplot(numberlinesgraph,numbercolumnsgraph,numberplot);

for j=1:length(P1)-2

ColorPoint=floor(63*j/LastPoint)+1;

Delta=(63*j/LastPoint)-ColorPoint+1;

RGB=PrettyColors(ColorPoint,:)+...

Delta*(PrettyColors(ColorPoint+1,:)-PrettyColors(ColorPoint,:));

plot3(P1(j),P1(j+1), P1(j+2), '.', 'MarkerEdgeColor',[RGB], 'MarkerSize',marker);

hold on;

end

axis([0 1 0 1 0 1]);

```

```
title(['k=',KK,', d1=',D1,', d2=',D2], 'FontSize',fsize,'FontWeight','bold');
```

```
numberplot = numberplot+1;
```

```
end
```

```
end
```