Dynamic Spread of Social Behavior in Boolean Networks

A Thesis

Presented to the

Department of Mathematics

and the

Faculty of the Graduate College

University of Nebraska

in Partial Fulfillment

of the requirements for the degree

Masters of Arts

University of Nebraska at Omaha

Elizabeth Ball

May 2011

Supervisory Committee:

Dora Matache
Jack Heidel
Andrew Swift
Nick Stergiou
Abstract: Dynamic Spread of Social Behavior in Boolean Networks

Elizabeth Ball, M.A.

University of Nebraska, 2011

Advisor: Dora Matache, Ph.D.

A variety of systems evolve towards achieving a global organization or coordination without a centralized process or control. As the systems evolve they tend to reach a certain level of harmony and may exhibit a significant robustness towards perturbations. This type of behavior can be observed in many natural systems. This type of system is quite common in our society as well. It is very easy to think of such social systems as networks whose nodes are the people or social groups, while the social relationships represent the links between the nodes.

In this work we use a Cellular Automata and Boolean Network approach to model a social network. We will focus on a Cellular Automata neighborhood approach, where each node is linked to its nearest neighbors. We are interested to know how the happiness spreads throughout the network over time. This type of study has been done before under various network scenarios, but a mathematical model for the time evolution fraction of happiness in the network of a social behavior has not been discussed in related literature. We will show that this mathematical model is suitable for the network under consideration and we specify the parameter values that lead to a good fit of the model. We use the model to assess the dynamics of the network.

We show that under the particulars specified in this work, the network reaches mostly stability: all people are happy or all are sad, so there is consensus. Under
a simple noise procedure, the network does not reach consensus anymore; however there is a convergence towards a more or less stable fraction of happy people in the network. The convergence to a stable behavior may take a few hundred steps of iterations. Clusters of happiness or sadness may form in the network.
Acknowledgements

I would like to thank Dr. Dora Matache for her excellence as an educator and for her guidance in this project. While greatly enjoying her mathematics courses, I have learned a tremendous amount from her and without her help and support this would not have been accomplished.

I would also like to thank Drs. Dora Matache, Andrew Swift, Jack Heidel, and Nick Stergiou for being on my thesis committee and for the time they have sacrificed to review this work and to give feedback to refine it.

Thanks must also be given to my mother who has spent countless hours taking care of my baby girl so that I could focus for an extended amount of time.

Finally and most importantly, I would like to thank my husband, James Ball. Through all my frustration and hard work, he remained patient and understanding even when I would work until my head spun. He kindly listened to me talk about this project and tried to understand even though he could not comprehend what I was saying. Thank you!
# Contents

List of Figures iii

1. Introduction 1

2. Background 4

2.1. Cellular Automata 4

2.2. Boolean Networks 8

2.3. Dynamical Systems and Chaos 11

2.4. Overview of Relevant Literature and Motivation for Current Work 12

3. The BN Mean-Field Model 18

4. Numerical Investigation 23

4.1. Fitting the model 23

4.2. Pattern Formation Plots 30

4.3. Bifurcations and Lyapunov Exponents 36

4.4. Focus on Fixed Points 39

5. Robustness to Noise 41

6. Conclusions and Directions for Future Work 50

References 52
List of Figures

1. Rule 126. Black cells are ON, white cells are OFF.

2. Pattern formation plots and corresponding fractions of active nodes by time steps for two different initial states of an ECA with 100 nodes. The left graphs correspond to one single active node (black cell) in the middle at time 0, while the left graphs correspond to a random initial state at time 0. The network is evolved 100 time steps.

3. Pattern formation plots and corresponding fractions of active nodes by time steps for two different initial states of a BN with 100 nodes. Each node has exactly three randomly chosen inputs. The left graphs correspond to one single active node (black cell) in the middle at time 0, while the left graphs correspond to a random initial state at time 0. The network is evolved 100 time steps. Observe the non-ordered behavior of the two pattern formation plots, and the differences between the corresponding fractions of active nodes. This figure is the analog of Figure 2 for random input selection.

4. Iterations of the network (blue dots) and the model (red curve) starting with various initial conditions of the network. The parameters are specified in the titles. The correlation coefficient $r$ is computed for each shown iteration. We note that the first iteration matches almost perfectly. Further iterations are not necessarily that close, but the correlation coefficient is high. This is due to the fact that the mean-field approach ignores the potential local correlations between nodes, while in the network evolution the correlations may build up with time. The larger the network the better the match.

5. This is the analog of Figure 4 with $N = 100$ and $k = 11$. Larger connectivity leads to higher values of $r$.

6. This is the analog of Figure 4 with $N = 200$ and $k = 15$. Larger connectivity leads to higher values of $r$.

7. Cobweb plot for two initial conditions $p = 0.47$ and $p = 0.55$ under the map (5). The orbits go to 0 and 1 respectively, so in the long run, the network reaches consensus: if initially there are less than $1/2$ happy
people then they all become sad in the long run. On the other hand if more than 1/2 of the people are happy, then there is a consensus of happiness. Here $q = 1/2, k = 15$.

8 Iterations graphs where one of the weights besides the one associated to the node itself is equal to 1/2. All other weights are equal. The first iteration shows that almost all points are fixed, so the system is frozen from the very beginning.

9 Pattern formation plot and corresponding fraction of active nodes for a network with $N = 100$ nodes, connectivity $k = 3$, and equal weights of the input nodes. Here $q = 0.5$. Observe that the network reaches a frozen state, so it is ordered or stable in just a couple of time steps. In this scenario, it is only the isolated nodes (e.g. and ON node among OFF neighbors) that switch to the state of the majority around them, so a fixed state of the network is reached very quickly.

10 Pattern formation plot and corresponding fraction of active nodes for a network with $N = 100$ nodes, connectivity $k = 5$, and weights of the input nodes as specified in the first title. Here $q \sim 0.53$. The middle node has the greatest weight. In this case, in order for a node to maintain its own state at sequential time steps, at least three nodes in its neighborhood need to be in the same state.

11 Pattern formation plot and corresponding fraction of active nodes for a network with $N = 100$ nodes and parameters as specified in the first title. Here $q > 1/2$, although it is still close to 1/2. Observe that the density of ones is constant, but the state of the nodes continually shifts.

12 Pattern formation plot and corresponding fraction of active nodes for a network with $N = 100$ nodes and parameters as specified in the first title. Here $q > 1/2$, although it is still close to 1/2. Observe that consensus is reached after a gradual shift from sad to happy through the network.

13 Pattern formation plot and corresponding fraction of active nodes for a network with $N = 100$ nodes and parameters as specified in the first title. Here $q < 1/2$, although it is still close to 1/2. The network reaches consensus (happiness in this case).
LyE and Bifurcation diagram along the parameter $q$ for $k = 5$. As expected, the system is stable, and the network reaches consensus.

LyE and Bifurcation diagram along the parameter $q$ from 0.4 to 0.6 for $k = 7$, $\alpha = 1/2$, and $w = [1/2 1/12 1/12 1/12 1/12 1/12 1/12 1/12]$. No consensus is reached. However, the dynamics are not chaotic.

Plot of the function $f$ together with the lines through $\pm 1$. Any fixed point $p$ for which $|f'(p)| < 1$, meaning that the point on the graph falls between the two lines, is stable. Observe that $p = 0$ and $p = 1$ are both stable. Any point for which $|f'(p)| > 1$ is unstable.

Pattern formation plots for a network with the following parameters: $N = 100, k = 7, q = 0.53, j = 10$, and equal weights of the inputs. Each pattern corresponds to a different initial state of the network. The patterns seem to converge to a balanced behavior biased either to happiness or sadness.

Pattern formation plots for a network with the following parameters: $N = 100, k = 7, q = 0.53, j = 10$, and a larger weight for one of the inputs. This pattern is a basically a random version of the one in Figure 12 which has the same parameters.

Pattern formation plots for a network with the following parameters: $N = 100, k = 5, q = 0.53, j = 10$, and a random selection of weights. We start with six different initial states of the network.

Pattern formation plots for a network with the following parameters: $N = 100, k = 5, q = 0.53, j = 25$, and a random selection of weights.

LyE and bifurcation diagram for a noisy system with $k = 5, \alpha = 0.1$, and equal weights of the inputs.

Observe the existence of two stable fixed points $p = \alpha$ and $p = 1 - \alpha$.

LyE and bifurcation diagram for a noisy system with $k = 5, j/N = 0.1$, and equal weights of the inputs.

Observe the existence of three stable fixed points $p = j/N, p = 1 - j/N$, and $p = 1/2$. 
LyE and bifurcation diagram for a noisy system with $k = 5, j/N = 0.25$, and random weights of the inputs.

Observe the existence of multiple stable fixed points besides $p = j/N$, and $p = 1 - j/N$. 
1. INTRODUCTION

A variety of systems evolve towards achieving a global organization or coordination without a centralized process or control. As the systems evolve they tend to reach a certain level of harmony and may exhibit a significant robustness towards perturbations. It is important to understand the origin of this global coordination. This type of behavior can be observed in many natural systems, such as certain models of decentralized cellular activity in human growth (see Lampl et. al. [24]), genetic regulatory mechanisms where the genes are activated or inhibited by only a few neighboring genes, but the entire system reaches a significant level of global coordination (see Kauffman [20], or Helikar et. al. [18]), or the global organization observed in some ant colonies, where the individual ants interact with only a few neighbors at a time without exhibiting individual periodic behavior, but the entire colony exhibits a very organized activity, including working and resting phases (see Cole [8], or Cornforth et. al. [9], [10]). The ant colonies are an example of an organized global social behavior based on local interactions. This type of system is quite common in our society as well. In that context, the members of the system are people or social groups, and the interaction between them are social relationships or social behaviors. It is very easy to think of such social systems as networks whose nodes are the people or social groups, while the social relationships represent the links between nodes. Each node is influenced by the interaction with other nodes of the network. Since social networks can be very large, the number of individuals that any node can influence is rather reduced in comparison to the size of the network. However, global coordination and
consensus (all nodes in the same state) can be reached despite the local level of interactions and a decentralized system. Social networks are all over; workplaces, living situations, friendships, cities, and families are all social networks. It is of importance to understand how social behavior can spread throughout the network.

In this work we use a Cellular Automata (CA) and Boolean Network (BN) approach to model a social network. As specified in Beck and Matache [4], Boolean network models have been used for modeling networks in which the node or cell activity can be described by two states, 1 and 0, 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. In fact, Stuart Kauffman originally developed random Boolean networks as models for genetic regulatory networks [20]. Since then BN models have been used for a large class of real or artificial networks from the brain to ecology (see Wuensche [43], genetic regulatory networks (see Shmulevich, et. al. [37], [6], [23], Helikar et. al. [18], chemical processes (see Heidel et. al. [17]), biology (see Klemm and Bornholdt [21], [22], and Raeymaekers [34]), neural networks (see Aldana and Cluzel [1], Huepe and Aldana [19]), physical networks (see Marr and Hütt [25], [26]), artificial life (see Wolfram [42]), and social networks (see Moreira et. al. [31], or Green et. al. [16]).

CA are a special case of BN where the nodes are positioned on a grid (in one or more dimensions) and all nodes evolve according to the same rule. An extensive study of CA was performed by Wolfram in [42]. Although the CA have a more structured nature, they have been used in modeling social change (see Nowak and Lewenstein [33]). Using BN and CA the dynamic spread of social behavior can be modeled and analyzed to determine the affects over time. Various conditions including the number
of individuals initially affected, the neighborhood sizes, the impact of each neighbor on a given node, as well as the size of the network can be altered very easily when using a BN or CA, allowing observations for differing scenarios and conclusions on the overall behavior of the network. For example, in one study on the spread of happiness, it was shown that clusters of happy and unhappy people develop where people who are surrounded by happy people tend to become happy over time. It was found that people’s happiness depends upon the people with whom they are connected [13].

In this work we define a social network whose nodes are people or social groups, that are subject to local interactions. To keep this study focused, we use a CA neighborhood approach, that is, each node is linked to its nearest neighbors. The size of the neighborhood may vary. This scenario may represent a social gathering where small groups of people are located at separate tables, with some level of communication between the tables. We use a Boolean approach for the state of each node: happy or 1, and sad or 0. We are interested in knowing how the happiness spreads throughout the network over time. At each time step, all nodes are updated based on the local interactions. We denote by \( p(t) \) the fraction of happy people at time \( t \). Using a mean-field approximation specific to BN, we generate a map that gives \( p(t + 1) \) in terms of \( p(t) \). We show that this mathematical model is suitable for the network under consideration and we specify the parameter values that lead to a good fit of the model. We use the map to study the behavior of the network over time. More precisely, we analyze the map using tools from dynamical systems and chaos. In particular, we provide sample graphical representations of the system evolution, we generate bifurcation diagrams and Lyapunov exponent calculations, and we discuss
the stability of the fixed points of the map. We show that the network exhibits a
global coordination and consensus. We also study the impact of noise on the network
behavior.

The thesis is structured as follows. In Section 2 we lay down the foundation of this
research with background information including information on Cellular Automata
(2.1), Boolean Networks (2.2), Dynamical Systems and Chaos (2.3), and an overview
of relevant literature and the motivation for this work (2.4). Section 3 contains the
model used in our study along with its derivation. In Section 4 we determine whether
our model fits the system and for which parameters (4.1), the pattern formation
plots associated with the model (4.2), the Bifurcation and Lyapunov exponents that
go along with the model (4.3), and an analysis of the fixed points (4.4). Section 5
applies the flip-rule to our model and reviews the behavior under this application
of noise. And finally, Section 6 gives the final conclusions regarding the use of our
model.

2. Background

2.1. Cellular Automata. A CA is a group of cells (nodes) arranged on grid of any
dimension that evolves through discrete time steps. The grid can be of any shape
and the evolution of these cells adheres to strict rules based on the binary states of
the cells in the previous time step. Each cell follows a rule that uses its neighboring
cells; the neighborhood is relative to each specific cell and does not change. The
same rule is applied to every cell for updating. John Von Neumann was one of the
first individuals to study a CA model [40]. CA were used in the early 1950’s to
study biological systems, and have been used for many other studies since. Stephen
Wolfram was one of the main contributors to the analysis of CA by the composition of his text “A New Kind of Science” [42] in 2002.

Elementary Cellular Automata (ECA) are the simplest form of CA in a one-dimensional grid. For ECA, the neighborhood consists of the nearest neighbors, that is the cell itself and the cells directly adjacent to the cell. Given the binary state space of each cell (ON or black, OFF or white), there are $2^3 = 8$ different states of the cell itself and its two nearest neighbors. For a CA with $N$ nodes, each cell $c_n, n = 1, 2, \ldots N$ evolves according to a Boolean rule $f : \{0,1\}^3 \rightarrow \{0,1\}$. Hence there are $2^{27} = 256$ Boolean rules that can be assigned to an ECA. Wolfram created a scheme assigning each rule to the integers 0 through 255 corresponding to the binary representation of each integer. One such rule is rule 126 (01111110 in binary). This rule’s simple interpretation is that complete crowding (all cells in its neighborhood are ON) causes the death of the cell (OFF) in the next time step, and complete isolation (all cells OFF) causes the death of the cell (OFF). Otherwise the cell is ON.

The tabular representation below shows the ECA rule 126.

\[
\begin{array}{cccccccc}
\black & \black & \black & \black & \black & \black & \black & \black \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
\white & \black & \black & \black & \black & \black & \black & \white
\end{array}
\]

**FIGURE 1.** Rule 126. Black cells are ON, white cells are OFF.

In Figure 2 we show a couple of sample pattern formation plots for ECA rule 126, together with the corresponding fraction of active nodes by time points. We consider a CA with 100 nodes (the horizontal axis in the top graphs) and iterate the network 100 time steps (time evolves downward). We plot a black dot for an active node and
a yellow dot for an inactive node. The bottom graphs contain the fraction of active
nodes (black dots) by time steps. The left graphs correspond to an initial state of
the network consisting of a single black cell in the middle, while the right graphs
correspond to a random initial state. We immediately note the differences between
the two situations. In the first case there is a growing pattern and the fraction of
active nodes tends to stabilize into an eventually periodic evolution. As a matter of
fact the period is about 300 time steps. We choose to show only the first 100 time
steps in order to observe the pattern formation more clearly. On the other hand, a
random initial state generates a complex pattern whose fraction of active nodes does
not settle into a periodic (ordered) behavior.

ECA rule 126 and its generalization to arbitrary neighborhood sizes has been ex-
tensively studied in the past few years by Matache et. al. [4], [15], [27], [28], [29]
in the context of Boolean networks for varying neighborhood sizes, synchronous and
asynchronous updating of nodes, or under noise procedures.

There are many applications and uses of CA in modeling today. We list just a
few examples. Traditionally, modeling biological systems is done using differential
equations, but when working with intracellular networks of genes, proteins, and such
using differential equations is very difficult because of the complexity of the network.
Besides, a CA or BN approach allows for a variation of the underlying parameters
governing the Boolean rules that is not possible when using continuous models. On
the other hand it has been shown that under certain circumstances, a continuous
model and its discrete analog may generate similar results [39]. Thus using CA
to model these systems is much more effective as indicated by Bonchev et. al. in
[5]. CA can also be used to model viruses. Again, usually differential equations
are used to study the evolution of the diseased over time, but CA is more effective because the spatial relationships can affect the spread of the virus. So instead of differential equations a two dimensional CA model has been used by Beauchemin [3], for example. Similarly, CA can be used to model biochemical reactions such as diffusion, viscous drag, enzyme rate processes, and metabolism as shown by Wishart et. al. [41]. Unrelated to biological studies, the principles of CA are applied to some central issues of ecology such as the possibility of generating ecological laws by Rhode [35]. Last, but not least, Nowak and Lewenstein use CA to model social change in
A detailed description of their work is included in Section 2.4. These are just some of the fields of study that CA can be used, there are many others as well.

2.2. Boolean Networks. CA are a special case of BN. A BN is CA without some of the restrictive assumptions of CA. For example, BN does not need to have consistent neighborhood sizes and the neighbors of a cell can be selected at random, they are not necessarily the “nearest” neighbors. The neighborhoods may be fixed for each cell or they may be modified at each time step. Each cell may obey a different Boolean rule or it may have multiple rules. It is also unnecessary for all the cells to be updated simultaneously. A network whose nodes are updated at the same time is called synchronous. Otherwise it is called asynchronous. This broadening of guidelines makes up the difference between CA and BN.

Stuart A. Kauffman was one of the first to use BN in his work [20]. He modeled genetic regulatory networks using BN by relating the cells/nodes to genes, and the ON or OFF state of the cell to the genes activity (active or inactive). Each gene (node) has $k$ inputs representing genetic regulatory mechanisms. The variable $k$, the connectivity, does not need be constant, but can be varied. In the simplest case each gene is assigned, at random, $k$ regulatory inputs from among the $N$ genes, and one of the $2^{2^k}$ possible Boolean functions of $K$ inputs. A new set of rules may be chosen at each time step. These are known as Kauffman or $Nk$ networks [20].

As mentioned in Section 2.1, the ECA rule 126 can be easily generalized to a BN by allowing a random choice of inputs as opposed to the nearest neighbor scenario, by allowing an arbitrary connectivity $k$ which may vary from one node to another, by generating an asynchronous updating of the network, or by inducing perturbations
along the way. These generalizations are the subject of several papers by Matache et. al. [27], [28], [15], [27], [28], [29]. In these works it is shown that under variable connectivity the network may exhibit order or chaos, and that large connectivity values suppress chaos [27], [28]. Certain types of asynchrony may induce order in the system, while others may lead to chaos [28], [29]. On the other hand, certain types of perturbations may widen the range of parameters for which order is observed [15]. Thus, any type of modification may yield a different type of evolution of the system.

In Figure 3 we show how the random choice of inputs may affect the pattern formation plots. We consider a BN with 100 nodes each having exactly 3 inputs that are randomly chosen. This figure is the analog of Figure 2 for random input selection. Note that although the patterns look similar, the fraction of active nodes shows more variation for the case of a random initial state (right graphs). Note also that the random choice of inputs has a great impact on the patterns, which are not very easy to understand. However, it is easier to see what is happening to the fraction of active nodes. This is a very important observation, since in general, if a network is very large, like many real network are, including social networks, then the number of possible states of the network can be enormous. It could be very hard to analyze the behavior of the network by looking at the patterns, as opposed to considering some numerical characteristics that can give an overall view of the network behavior. That is, some numerical measures that replace the focus on individual nodes with a focus on the behavior of the entire system. This kind of approach is very common in statistical physics, and it is known as the mean-field approach. The main idea is to replace all interactions to any one body/particle (nodes in our terminology) with an average or effective interaction of neighboring bodies. This reduces any multi-body
problem into an effective one-body problem. The ease of solving mean-field theory problems means that some insight into the behavior of the system can be obtained at a relatively low cost.

One of the goals of BN modeling is to provide a fairly easy to understand way to describe the system behavior. One typical approach is to provide a mathematical formula for the so called density of ones, that is the probability of finding a node in

---

**Figure 3.** Pattern formation plots and corresponding fractions of active nodes by time steps for two different initial states of a BN with 100 nodes. Each node has exactly three randomly chosen inputs. The left graphs correspond to one single active node (black cell) in the middle at time 0, while the left graphs correspond to a random initial state at time 0. The network is evolved 100 time steps. Observe the non-ordered behavior of the two pattern formation plots, and the differences between the corresponding fractions of active nodes. This figure is the analog of Figure 2 for random input selection.
state 1 or ON at a given time $t$. The fraction of active nodes is an estimate for the density of ones. This can be done using a mean-field approximation. We will be using the mean-field approach for the network in the current study.

2.3. Dynamical Systems and Chaos. In this subsection we provide a brief review of some basic concepts from dynamical systems and chaos. We recall some facts from Alligood et. al. [2].

**Definition 1.** A map is a function $f : S \to S$, whose domain (input) space and range (output) space are the same ($S$). The orbit of a point $x$ under this map is \( \{x, f(x), f^2(x), \ldots, f^n(x), \ldots\} \), where $f^n$ represents the $n$-th composition (iteration) of a function with itself.

**Definition 2.** A point $p$ is a fixed point of the map $f$ if $f(p) = p$.

**Definition 3.** Let $f$ be a map on $\mathbb{R}$ and let $p$ be a real number such that $f(p) = p$. If all points sufficiently close to $p$ are attracted to $p$, then $p$ is called a stable or attracting fixed point. If all points sufficiently close to $p$ are repelled from $p$, then $p$ is called a source or a repelling fixed point.

**Theorem 1.** Let $f$ be a (smooth) map on $\mathbb{R}$, and assume that $p$ is a fixed point of $f$.

1. If $|f'(p)| < 1$, then $p$ is a sink.
2. If $|f'(p)| > 1$, then $p$ is a source.

In a similar manner one can define higher order fixed points as solutions of $f^n(p) = p$. These generate periodic orbits of period $n$, $\{p_1, p_2, \ldots, p_n\}$, that can be classified into attracting and repelling orbits, depending on the value of the derivative of $f^n$ at those points as in Theorem 1.
The stability of a system is influenced by the derivative of the map \( f \). If \( x_1 \) is a fixed point and the derivative \( f'(x_1) > 1 \), then the orbit of \( x \), a point near \( x_1 \), is moving away from \( x \), as opposed to moving closer. This is done at a rate of approximately \( f'(x_1) \) per iteration. In other words, the distance between \( f^n(x) \) and \( f^n(x_1) \) grows or shrinks according to the value of \( f' \) at the points of the orbit. The Lyapunov exponent is used to quantify this rate of drawing nearer to or falling away from an initial point, \( x_1 \).

**Definition 4.** The Lyapunov exponent \( \text{LyE}(x_1) \) of the orbit \( \{ x_1, x_2, x_3, \ldots \} \) (periodic or not) is defined as \( \text{LyE}(x_1) = \lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} \ln |f'(x_i)| \), if the limit exists.

Now, a \( \text{LyE} \leq 0 \) corresponds to stability or asymptotic stability (the orbit approaches a periodic behavior).

**Definition 5.** Let \( f \) be a map of the real line \( \mathbb{R} \), and let \( \{ x_1, x_2, x_3, \ldots \} \) be a bounded orbit of \( f \). The orbit is chaotic if \( \{ x_1, x_2, x_3, \ldots \} \) is not asymptotically periodic and \( \text{LyE}(x_1) > 0 \).

We will be using the terminology presented in this section throughout this work. For more on dynamical systems and chaos check [2].

2.4. **Overview of Relevant Literature and Motivation for Current Work.**

The dynamics of social behavior is the cause of great interest and the motivation for many works. Because of their simplicity, CA and BN have recently been a method of studying human behavior and their dynamics. An article written by Hegselmann and Flach, *Understanding Complex Social Dynamics: A Plea For Cellular Automata Based Modeling*, argues that using CA is a favorable approach to understanding social
dynamics [12]. There are several studies that are related to this work and contribute greatly to the conversation. There are others that will not be mentioned, but those discussed below are the most relevant to this particular study.

One such relevant study is *Modeling Social Change with cellular Automata* by Nowak and Lewenstein [33]. In this work the authors are seeking to model social processes in the area of social opinion and change. The authors assume that each individual in the system has an original opinion, they are either "for" a policy or "against" the policy. Each individual has a varying strength for which they are for or against this policy (thus there are leaders). They arrange their system in a two dimensional lattice where the location of the individuals matters; the closer the neighbor the more influence the neighbor has on the individual. The authors found that as this system evolves clusters of opinions appear. They find that if there are not differing strengths in opinions then even a minimal amount of noise leads to uniformity of opinion. They found that these differing strengths among the individuals led to the ability for minority opinion to survive, leading to clustering of opinion. This study is similar to ours in that different neighbors have differing impacts. However, the weights of the impacts are allowed to vary in the current study, as opposed to [33] where they are kept constant. This accounts for possible readjustment of trusts in the neighbors’ opinions over time.

Another such paper is *The Emergence of Social Consensus in Boolean Networks* by Green et. al. [16]. In this paper the authors state that social unity and order require consensus. The authors model this idea of consensus using a Boolean Network where the links are interactions between individuals. Each individual has an opinion. If the neighbor the individual interacts with is of the same opinion, then neither of
the individuals changes opinion. If the individuals are of differing opinions, then one of the individuals (selected randomly) changes opinion with an assigned probability. The states are then updated asynchronously where each time period consists of a fixed number of interactions. The authors found that the size of the network that reached consensus depended upon peer influence and the density of the social connections. When these two factors fell below a certain level, only small networks would reach consensus. They found that if there were more connections (more neighbors taken into account) then larger networks were able to reach consensus. The authors also introduced a "law enforcement" aspect and found that large networks can reach consensus if peer influence (differing weight of individuals or leaders) and law enforcement is introduced to the system. Without differing weights and this law enforcement aspect they concluded that large networks rarely reached consensus. This is similar to our study in that again we want to assign weights to individuals and vary the number of influencing numbers as well as the size of the network to see the dynamic change these bring to the network. On the other hand we are interested in a more static neighborhood formation that doesn’t change over time. Like a group of people one trusts (the neighborhood), although the level of trust may change in time. For the purpose of this study, we will assume a synchronous network, as opposed to [33], where asynchrony is incorporated. However, what asynchrony does is brings a level of uncertainty into the network, that is noise. We will incorporate noise into our system in two different ways: by allowing varying weights for the impact of neighbors, and by inducing random flips of opinions in the network. We will show that despite the lack of law enforcement, our network can reach consensus.
Efficient System-Wide Coordination of Noisy Environments by Moreira et. al. [31] is another article that has greatly influenced our work. The focus in that work is the origin of consensus that appears in many natural and social systems. In that model, the inputs depend upon the current state of the node and its neighbors using a majority rule that does not distinguish between the impacts of neighbors. The authors want to show that sophisticated strategies to reaching consensus are not robust under environmental changes (or noise). Each node is influenced by a specified number of nearest neighbors, and each node rewires (connects with different nodes) by a probability of $p$. The authors found that noise and topology greatly affect the ability for consensus to occur. These authors used the Gacs-Kurdyumov-Levin (GKL) rule [14] to achieve consensus. This rule was discovered as a way of reaching consensus in a network. It states that a node takes on the value of the majority of the neighborhood containing itself and the first and third neighbors. The node changes based only upon itself and its left or right neighbor depending upon its current state. Using this rule consensus is reached quite rapidly. The authors of [31] basically use the GKL rule but also includes noise and asynchronous updating. They conclude that noise and topology affect the ability of the global coordination rule. They find that the GKL rule is not effective when noise is introduced and in an environment with complex topology. The main reason is that the GKL rule is very restrictive and works in noiseless environments. However, the authors discover that the majority rule is effective in reaching global coordination under some level of noise and rewiring. The authors show that the environment must be taken into account when studying these consensus rules. They conclude that the rules and the environment co-evolve over time. In our work we consider a weighted combination of
inputs and apply a threshold function (majority rule) to decide the state of the node at the next time step. Although we do not allow rewiring, we do account for noise in the network. We show that consensus can be achieved with or without noise.

Using a longitudinal social network analysis, Christakis and Fowler [7] determined to see whether or not happiness is spread between people in a social network, and whether niches or groups of happy people appear. In this twenty year study that follows up with a group of individuals it was found that niches do occur, both groups of unhappy and of happy people. Those individuals who were surrounded by happy people tended to become happy as time progressed, and those individuals who were central in the network tended towards happiness. They verified that this is indeed the spread of happiness and not just the tendency of people who have like personalities to be drawn to one another. The proximity or geographical separation between individuals affected the impact that they had on one another. Another similar study by Fowler and Christakis found that both opinions and social structure affect the formation of social groups [36]. They analyzed the relationship and impact a dynamically changing social network has with the spread and information gathering aspect of the network. Their interest spiked from observing social groups with vary differing tastes in music, politics, and religion emerge and disappear from extreme subcultures and also in mainstream mass-cultures. Their analysis concluded that these emerge as a result of contact-seeking individuals who reinforce their beliefs among the individuals within their group, or with those who share their opinion. Although that study does not involve a CA or BN approach, it inspired the current work to consider the idea of spread of happiness throughout a CA or BN counterpart.
There is another important aspect that our work contains as opposed to all the studies mentioned above. Namely the idea of a mean-field approximation for the fraction of active (ON, happy, 1) nodes by time steps. This approach has not been used before in the context of specific social networks. However, the work by Matache et. al. [4], [11], [15], [27], [28], [29], provides and in-depth analysis of BN governed by generalized ECA rules 126 and 22 using mean-field formulae for the density of ones under a variety of scenarios, as mentioned before. These include: fixed or variable connectivity and neighborhoods, synchrony or (deterministic or stochastic) asynchrony, noiseless or noisy environments. We will use a similar approach for the current social network. First we will describe the network in detail and use tools from probability theory to generate the mean-field formula for the density of ones. Then we will show that the model provides a good fit for the actual fraction of active nodes as obtained from a network evolution. Once we know the model is a good fit, we can study the map using tools from dynamical systems and chaos. We look at the fixed points of the map, bifurcation diagrams, Lyapunov exponents. Then we induce noise in the system by random flips of node states. We show that order prevails.

The previously mentioned works are only a snapshot of the works that have been conducted on the topic of this thesis. There are numerous studies that have used BN and CA as models. In this work we seek to add to this conversation. Social interactions are incredibly complex and we seek to understand their dynamics a little more through this effort.
3. The BN Mean-Field Model

Consider a cellular automaton (CA) with \( N \) nodes (cells) \( V = \{x_1, x_2, \ldots, x_N\} \) and corresponding connectivity \( k = 2m + 1 \). The nodes represent individuals or social groups. The state of a node can be 1 (ON, happy, active) or 0 (OFF, sad, inactive). The inputs of node \( x_n \) are denoted by the vector

\[
(v_{n1}, v_{n2}, \ldots, v_{nk}) = (x_{n-m}, x_{n-m+1}, \ldots, x_n, \ldots, x_{n+m}).
\]

Thus the inputs are the node itself and the \( k - 1 \) nearest neighbors. The neighbors could be regarded as actual adjacent individuals around a table, or close friends or social contacts. For notation purposes, we assume that \( x_{N+1} \) is actually \( x_1 \), as if the nodes were ordered around a circle (round table). We assume a synchronous updating of the cells. The Boolean rule followed by all nodes is defined below.

Our goal is to model the way social behavior is spread throughout a social network. Say the behavior under consideration is for example good humor, and the neighbors are physically close to the others. Then the impact of those really close to an individual should have more weight than the impact of those located farther off. Therefore to each node \( x_n \) we associate a collection of \( k \) weights which indicate the impact that each input has on the output of node \( x_n \). Thus let the weights be \( w_n = (w_{n1}, w_{n2}, \ldots, w_{nk}) \). Then \( w_{ni} \in [0, 1] \) for all \( i \), and \( \sum_{i=1}^{k} w_{ni} = 1 \). Observe that \( w_{n\frac{k+1}{2}} \) is the weight corresponding to the node under consideration, \( x_n \). The Boolean rule giving the state of the node at \( t + 1 \) is

\[
x_n(t+1) = \chi_{(q,\infty)} \left( \sum_{i=1}^{k} w_{ni} v_{ni}(t) \right)
\]
where \( q \in (0, 1) \) and \( \chi_A(x) \) is the characteristic function of the set \( A \). That is

\[
\chi_A(x) = \begin{cases} 
1 & \text{if } x \in A \\
0 & \text{if } x \notin A
\end{cases}
\]

The rule basically says that if the weighted sum of all inputs \( \sum_{i=1}^{k} w_{ni} v_{ni}(t) \) surpasses a given threshold \( q \), then the node is turned ON, otherwise it is turned OFF. Thus, the Boolean rule is a linear threshold function, which is common to neural networks. In terms of the social network, if the threshold is fairly low, then we account for a more optimistic nature of the individual under consideration, with a bias towards happiness. If the threshold is fairly high, then there is a bias towards unhappiness.

Note that if a weight \( w_{ni} \) is greater than \( q \) and \( v_{ni}(t) = 1 \), then automatically \( x_n(t + 1) = 1 \). In particular, if \( x_n(t) = 1 \) and \( w_{ni} \frac{\sum_{i=1}^{k} w_{ni} v_{ni}(t)}{n} > q \), that is the node’s own weight is larger than the threshold value, then \( x_n(t + 1) = 1 \). This would account for self-confident individuals, for which their own opinion matters most.

As mentioned in Section 2.4, Moreira et al. [31] consider a version of this model in which the inputs depend on the current state of the node. This is the so called Gacs-Kurdyumov-Levin (GKL) rule. This rule was the starting point for the density-classification task, the benchmark for exploring cellular automata. The GKL rule was designed in 1978 [14] to study reliable computation and phase transitions in one-dimensional spatially extended systems.

In [31] it is shown through pattern formation plots that the network tends to reach consensus rather quickly. On the other hand, noisy environments are shown to lead to a more complex behavior. They also show that under asynchronous updating, the network may not reach consensus. We are basically generalizing their results to
arbitrary weights and add the mean-field analysis of the network behavior. To this aim, let us denote by \( p(t) \) the density of ones at time \( t \), or equivalently, the fraction of active cells at time \( t \). This is an estimate for the probability of finding a generic node ON at time \( t \). We determine \( p(t + 1) \) in terms of \( p(t) \) using a mean-field approach.

More precisely, if \( x \) is a generic node of the network, then we would like to compute \( p(t + 1) = \mathcal{P}(x(t + 1) = 1) \), where the notation \( \mathcal{P} \) stands for the probability of an event. By the law of total probability this is given by

\[
\mathcal{P}(x(t + 1) = 1|x(t) = 0) \cdot \mathcal{P}(x(t) = 0) + \mathcal{P}(x(t + 1) = 1|x(t) = 1) \cdot \mathcal{P}(x(t) = 1).
\]

But \( \mathcal{P}(x(t) = 1) = p(t) \) and thus \( \mathcal{P}(x(t) = 0) = 1 - p(t) \), which implies

\[
p(t + 1) = (1 - p(t)) \cdot \mathcal{P}(x(t + 1) = 1|x(t) = 0) + p(t) \cdot \mathcal{P}(x(t + 1) = 1|x(t) = 1).
\]

Thus to obtain \( p(t+1) \) we need to find out what are the expressions of \( \mathcal{P}(x(t+1) = 1|x(t) = 0) \) and \( \mathcal{P}(x(t+1) = 1|x(t) = 1) \). To this aim we will make the assumption that the inputs act independently of each other. This is a simplifying assumption that is typical for the mean-field approximation which replaces all interactions to any one node with an average or effective interaction of neighboring bodies. The reason is that if we assume \( N \) to be large enough we can ignore the potential local correlations that are formed between nodes.

Note that

\[
\mathcal{P}(x(t + 1) = 1|x(t) = 0) = \mathcal{P} \left( \sum_{i=1}^{k} w_{ni} v_{ni}(t) > q \right)
\]
and

\[ P(x(t+1) = 1 | x(t) = 1) = P \left( \sum_{\substack{i=1 \atop i \neq k+1 \atop j=1}}^{k} w_{ni}v_{ni}(t) > q - w_{n,k+1} \right) \]

Now, if \( q - w_{n,k+1} < 0 \Leftrightarrow w_{n,k+1} > q \) then \( P(x(t+1) = 1 | x(t) = 1) = 1 \). Thus we can write

\[ P(x(t+1) = 1 | x(t) = 1) = \chi_{(q,\infty)}(w_{n,k+1}) + \]

\[ (1 - \chi_{(q,\infty)}(w_{n,k+1})) P \left( \sum_{\substack{i=1 \atop i \neq k+1 \atop j=1}}^{k} w_{ni}v_{ni}(t) > q - w_{n,k+1} \right) . \]

Thus

\[ P(x(t+1) = 1) = (1 - p(t))P \left( \sum_{\substack{i=1 \atop i \neq k+1 \atop j=1}}^{k} w_{ni}v_{ni}(t) > q \right) + p(t)\chi_{(q,\infty)}(w_{n,k+1}) + \]

\[ p(t) \left( 1 - \chi_{(q,\infty)}(w_{n,k+1}) \right) P \left( \sum_{\substack{i=1 \atop i \neq k+1 \atop j=1}}^{k} w_{ni}v_{ni}(t) > q - w_{n,k+1} \right) . \]

Now we find the probabilities involved in (2).

\[ P \left( \sum_{\substack{i=1 \atop i \neq k+1 \atop j=1}}^{k} w_{ni}v_{ni}(t) > q \right) = \sum_{s=1}^{k-1} \sum_{\substack{i_1 < i_2 < \cdots < i_s \atop i_j \neq k+1, j=1,2,...,s}} p(t)^s(1 - p(t))^{k-1-s} \chi_{(q,\infty)} \left( \sum_{j=1}^{s} w_{ni,j} \right) \]

(3)

\[ = \sum_{s=1}^{k-1} p(t)^s(1 - p(t))^{k-1-s} \sum_{\substack{i_1 < i_2 < \cdots < i_s \atop i_j \neq k+1, j=1,2,...,s}} \chi_{(q,\infty)} \left( \sum_{j=1}^{s} w_{ni,j} \right) . \]

Observe that for a given set of weights, the quantity

\[ \sum_{\substack{i_1 < i_2 < \cdots < i_s \atop i_j \neq k+1, j=1,2,...,s}} \chi_{(q,\infty)} \left( \sum_{j=1}^{s} w_{ni,j} \right) \]

is independent of the actual state of the network, and can be computed from the very beginning.
Similarly,

\begin{equation}
\mathcal{P}\left(\sum_{i=1}^{k} w_{ni} v_{ni}(t) > q - w_{n \frac{k+1}{2}}\right) =
\end{equation}

\begin{align*}
\sum_{s=1}^{k-1} p(t)^s (1-p(t))^{k-s} \sum_{i_1 < i_2 < \cdots < i_s \atop i_j \neq \frac{k+1}{2}, j=1,2,\ldots,s} \chi(q - w_{n \frac{k+1}{2}}, \infty) \left(\sum_{j=1}^{s} w_{nj}\right),
\end{align*}

Multiplying equation (3) by \((1 - p(t))\) and equation (4) by \(p(t)\) we obtain

\begin{align}
P(x(t + 1) = 1) &= \sum_{s=1}^{k-1} p(t)^s (1-p(t))^{k-s} \sum_{i_1 < i_2 < \cdots < i_s \atop i_j \neq \frac{k+1}{2}, j=1,2,\ldots,s} \chi(q, \infty) \left(\sum_{j=1}^{s} w_{nj}\right) + \\
\left(1 - \chi(q, \infty)(w_{n \frac{k+1}{2}})\right) \sum_{s=1}^{k-1} p(t)^{s+1} (1-p(t))^{k-1-s} \sum_{i_1 < i_2 < \cdots < i_s \atop i_j \neq \frac{k+1}{2}, j=1,2,\ldots,s} \chi(q - w_{n \frac{k+1}{2}}, \infty) \left(\sum_{j=1}^{s} w_{nj}\right) + \\
p(t) \chi(q, \infty)(w_{n \frac{k+1}{2}}).
\end{align}

In the next section we check the accuracy of this model by comparing it to the actual fraction of ON nodes obtained from iterations of the network.

Observe that the fact that the neighbors are the adjacent nodes has not been explicitly used in the formula above. More precisely, regardless of the way the \(k\) neighbors of node \(x_n\) are chosen, they can be ordered and labeled as \((v_{n1}, v_{n2}, \ldots, v_{nk})\) with the associated weights \(w_n = (w_{n1}, w_{n2}, \ldots, w_{nk})\). So the formula (5) can be easily used in the context of a BN. Note also that one could actually consider time dependent weights in this model. Then all the sums involved in (5) would be time dependent and therefore would be updated every time step. The advantage of time dependent weights is that it allows more flexibility in the actual social network. The mutual influences between nodes may change dynamically based on likes and dislikes that can evolve in time for example. This induces perturbations in the natural development of the
network. We will explore these other possibilities in the numerical investigations to follow.

4. NUMERICAL INVESTIGATION

In this section we first assess the accuracy of the model. We do this by comparing the results for $p(t)$ obtained from the simulation of the network and compare them to the results of the model. Once the model is shown to be a good match for the actual system, we can use it to predict the dynamics of the network under various scenarios. All the simulations are performed using Matlab, a technical computer software produced by MathWorks [30]. All the Matlab codes used for simulations are included in Appendix 2.

4.1. FITTING THE MODEL. The first step in assessing the accuracy of the model is to see if the values generated for the density of ones using the model (5) are sufficiently close to the actual fractions of active nodes obtained from evolving a network. In order to assess the accuracy, we start with many different initial conditions $p(0)$ and iterate the model a number of times plotting the resulting $p(t)$ values. On the other hand, for the same initial values $p(0)$ we iterate a network under the same parameter values, find the fraction of ones and plot the results on the same graph. For the network we randomly select $p(0) \cdot N$ nodes to be ON at time $t = 0$. We average the resulting $p(t)$ values for up to 1000 such selections of $p(0)$. Then we graph the density of ones of the system versus the model in order to compare them. Using Matlab we determine the accuracy of the model for various values of $k, w, q$. For the weights $w$ we have considered fixed values, such as all nodes in a neighborhood have the same weight, as well as randomly generated values using a normal distribution. We will
show only typical results. The parameters of the normal distribution are chosen such
that the generated numbers are mostly positive. In case of a negative number, we
reassign it to be a very small positive number. The weights are obtained by scaling
the randomly generated numbers such that they are all in \([0,1]\) and add up to 1.
Some of the actual parameter combinations are included in Appendix 2, Tables 1
and 2. The threshold, \(q\), is chosen in Matlab by generating a sequence of numbers
starting with the weight of the middle node and then increasing by a small step until
a weight greater than 0.7 is reached. For example, if the middle node has a weight
of 0.2489, then the following \(q\) values will be used: 0.2489, 0.3489, 0.4489, 0.5489,
and 0.6489. A \(q\) value less than the weight of the middle node is not used because
this would cause any ON node to automatically be ON in the next iteration, so it
becomes automatically frozen. A \(q\) value larger than 0.7 would require a very large
majority of the nodes in the neighborhood to be ON in order for the node to be ON
in the next iteration, due to a significant bias towards the inhibition of the node. So
it is sufficient to use only these values for \(q\).

We use the Matlab code *iterations_SocialSpread.m* found in the appendix to
obtain results for a wide variety of parameter combinations. In Figure 4 we show
the first, second, third, and twentieth iteration for the following parameter values:
\(N = 100, k = 7, q = 0.5,\) and \(w = [0.13712 \ 0.14418 \ 0.17104 \ 0.14264],\) so the neighbors
have more or less similar impacts on the node. Observe the perfect match of the first
iteration. As the system is iterated further we lose some accuracy in the fitting of
the model. This is to be expected since the mean-field approach ignores the potential
local correlations between nodes, while in the network evolution the correlations may
build up with time, that is with iterations. The mean-field model works best for
Figure 4. Iterations of the network (blue dots) and the model (red curve) starting with various initial conditions of the network. The parameters are specified in the titles. The correlation coefficient $r$ is computed for each shown iteration. We note that the first iteration matches almost perfectly. Further iterations are not necessarily that close, but the correlation coefficient is high. This is due to the fact that the mean-field approach ignores the potential local correlations between nodes, while in the network evolution the correlations may build up with time. The larger the network the better the match.

Very large networks, and the larger the network the better the match. However, for practical purposes of simulations, a very large $N$ may be prohibitive, due to software limitations. We were able to check the accuracy up to $N = 300$, for which the graphs look similar, but the match is more accurate.

Aside from the visual inspection of the graphs for accuracy, for each parameter combination we compute the correlation coefficient $r$ between the values $p(t)$ obtained with the model and the actual network. The correlation coefficient for a set of $n$ data points $(x_i, y_i)$ where $x_i$ denotes the actual value of our system and $y_i$ represents the values produced by the model is defined as:

$$r^2 = \frac{SS_{xy}^2}{SS_{xx}SS_{yy}}$$
where

\[ SS_{xx} = \sum (x_i - \bar{x})^2, \quad SS_{yy} = \sum (y_i - \bar{y})^2, \quad SS_{xy} = \sum (x_i - \bar{x})(y_i - \bar{y}) \]

The correlation coefficient, \( r \), is used as a measure between two variables. This value, \( r \) will produce a number between 0 and 1, in which a value of 1 means that the two variables form a perfect relationship. Thus, a value close to 1 means a strong correlation between the density of ones obtained with the model versus the actual network. In Figure 4 we indicate the correlation coefficients in the titles. Note that they are all close to 1.

On the other hand, we have included a more comprehensive list of parameters values used in simulations and correlation coefficients in the appendix, to supplement the graphs in this section. As one can see in Table 1 and in Table 2 included in Appendix 2, some parameter values generate a better fit even for smaller values of \( N \). These are used mostly in further simulations. As seen in the tables, values of \( q \) that are close to 1/2, regardless of the values for \( k \) and the weights, lead to high values of \( r \). Thus, the model is a better fit in the case of a threshold value close to 1/2. Now, \( q \sim 1/2 \) means that each node (or individual in the social network) is unbiased with respect to positive versus negative inputs from its neighbors. This means there is no bias towards the activation or inhibition of the node. Of course, the weights may not be equal, so different neighbors may still have different impact. As our neighborhood becomes larger, or as \( k \) increases, one can see that the value of \( q \) can deviate further from one-half and produce a good fit. It can also be seen from the information in Table 1 that the values of the weights do affect \( r \), but this seems insignificant in comparison to the affects that \( q \) has upon \( r \). For smaller values of \( k \),
like $k = 3, 5, 7$, if $.40 < q < .55$, then $r > .9$ and the model is a good fit. Larger values of $k$ allow a greater range of $q$, but nonetheless, $.40 < q < .55$ works for all initial conditions. Moreover, the mean-field approximation assumes very large networks. The larger the network and the connectivity, the better the match between the model and the network, since the local correlations do not have such a great impact on the overall (average) behavior.

Using larger values of $k$ will produce better fits to our model as well. Figure 5 in which $N = 100$ and $k = 11$ shows higher correlation coefficients than the previous graphs. In Figure 6, $N = 200$ and $k = 17$. Notice again the better fit and higher values of $r$. Observe also that as the network and connectivity become larger, the graph approaches a step function. The actual step can be closer to zero or one depending on $q$. This indicates that there are only three fixed points, one of them being $p = 0$ and the another one being $p = 1$ of course. Higher order iterations do not add to the complexity of the behavior, so we expect stability. We can actually see this in a cobweb plot, which is a graphical representation of orbits under a given map. More precisely, we plot the actual function (5) together with the main diagonal to identify the fixed points, and then we generate the orbits of two initial conditions, namely $p = 0.47$ and $p = 0.55$. We see that the orbit of $p = 0.47$ converges to the origin, while the orbit of $p = 0.55$ converges to 1. All the points $p < 1/2$ are attracted to 0, while all the points $p > 1/2$ are attracted to 1. In this case $[0, 1/2)$ is the basin of attraction of 0, while $(1/2, 1]$ is the basin of attraction of 1. The Matlab code used here is `cobwebplot_SocialSpread.m`.

If one of the nodes in a given neighborhood has a significant larger impact, the situation is different. For example, in Figure 8 the weights of node $x_n$ are $w_n =$
Figure 5. This is the analog of Figure 4 with $N = 100$ and $k = 11$. Larger connectivity leads to higher values of $r$.

$(1/2, w, \ldots, w)$, where $w = \frac{1}{2(k-1)}$. Observe that the map almost coincides with the first diagonal. This means that almost every point $p \in [0, 1]$ is a fixed point of the map (5), so the system is frozen from the very beginning. Thus, such a combination of parameters is prohibitive for a dynamic evolution of the spread of happiness. The main reason why we actually see a departure of the model from the actual network in the twentieth iteration is that the parameters $N$ and $k$ are too small. However, as mentioned before, large values would remedy this situation. The main goal is to point out the different type of behavior observed in this scenario.

All the figures in this section are typical for all the parameter combinations used in simulations. Some plots using some of the values in Table 1 of Appendix 1 will be included in the next sections. Observe that the map (5) does not depend explicitly on $N$, but it does depend on $k$. If $N \to \infty$ then we can allow $k \to \infty$. We will explore higher values of $k$ in the next sections where we perform LyE computations to assess the behavior of the system for a variety of parameter combinations.
Figure 6. This is the analog of Figure 4 with $N = 200$ and $k = 15$. Larger connectivity leads to higher values of $r$.

Figure 7. Cobweb plot for two initial conditions $p = 0.47$ and $p = 0.55$ under the map (5). The orbits go to 0 and 1 respectively, so in the long run, the network reaches consensus: if initially there are less then $1/2$ happy people then they all become sad in the long run. On the other hand if more than $1/2$ of the people are happy, then there is a consensus of happiness. Here $q = 1/2, k = 15$. 
Figure 8. Iterations graphs where one of the weights besides the one associated to the node itself is equal to 1/2. All other weights are equal. The first iteration shows that almost all points are fixed, so the system is frozen from the very beginning.

4.2. Pattern Formation Plots. Here we include a few pattern formation plots for several parameter combinations for the actual network, so we can see at least in a few cases the types of patterns that can occur.

Recall that for generating a pattern formation plot we start with a random initial condition at time $t = 0$ and we iterate the network a number of times and plot a black dot if a node is ON and a yellow dot if a node is OFF. The nodes are arranged horizontally and the time evolves downwards. Remember that we are assigning weights to the $k$ nearest neighbors on each side of the node under consideration and the node itself. If the weighted sum of the input values is greater than the threshold $q$, then the node turns ON in the next time step. Otherwise, the node is OFF in the next time step. Each line of the plot represents the state of the system at time $t$. All the graphs have been obtained by running the Matlab code `PatternFormation_SocialSpread.m`.

When these plots were drawn in Matlab for this particular model, the results from above impacted what values we should choose. It was seen in the previous section
that using $q$ values close to $1/2$ produced a good fit (high $r$ values). If $q$ is much less
than $1/2$ then all the nodes in the system went to 1, or ON. If $q$ was much greater than
$1/2$, then the entire system tended to 0, or OFF. The previous section led us to the
conclusion that for practical purposes we should consider $q \sim 1/2$, so our discussion
in this section will be focused mainly on these values, since they produce a good fit
to the model even for smaller values of $N$. This is important for simulation purposes,
given that large $N$ values tend to take extensive computer time.

When all nodes have the same weight in the neighborhood, then for any value of
$q$, the system reaches a frozen state after just a few iterations. This is seen in Figure
9. When $k = 3$ it took two or more ON or OFF cells next to one another in order
to remain either ON or OFF (respectively). In other words, if an ON/OFF node was
isolated then it switched to the value of the nodes surrounding it. If two ON/OFF
nodes were next to one another then they would remain in the same state at the next
time step. Similar situations occurred for other values of $k$. When $k = 5$, it takes
three or more ON/OFF nodes next to one another in order for the nodes to remain
ON/OFF in the next time step, if there are only two ON/OFF nodes next to one
another they will switch values in the following time step. For example, if $k = 5$ and
if there are two ON nodes next to one another with an OFF node on either side of
the pair, then the ON nodes will be OFF in the next time step. However, if there
are three ON nodes together with an OFF node on either side of the group, then the
nodes will remain ON in the next time step. The same pattern holds if the nodes were
OFF as opposed to ON. Of course this happens due to the level of $q$, which indicates
the threshold that needs to be surpassed in order for a node to turn ON. Similarly,
for $k = 7$, it takes four or more grouped nodes to retain the ON/OFF value, and
the pattern continues for larger values of $k$. Note that the larger the neighborhood the more likely the neighborhood will go all ON or all OFF (a consensus) because it is necessary for a greater number of cells to be grouped together in order to remain (to not be washed out). This is because one node is input to many other nodes and therefore its value will impact more neighbors at the same time. So if the individuals in a network have a greater level of connectivity, then it is very likely that they will reach consensus, whether it be a happy or unhappy state. This can actually be seen from the iterations in the previous section. The larger the connectivity, the closer the graph of $p(t)$ and its iterations to a step function with values 0 and 1.

There were similar results when the node itself (the middle node) had the greatest weight. For any weights and any values of $k$, the system again reached a frozen state in a few iterations. Like before, ON/OFF nodes needed to be in groups in order to remain ON/OFF in the next time step. Unlike before, these ON/OFF nodes need only be grouped in a neighborhood, not necessarily right next to one another. This is seen in Figure 10. For example, for $k = 5$, it was necessary for three or more ON nodes to be in a neighborhood together in order to remain ON. It is more likely for an individual in a network to keep a behavior when the individual places the most weight or importance on themselves and less on the neighbors.

Other patterns occur when the node itself does not have the greatest weight. If one of adjacent nodes has a weight equal to $1/2$, then more interesting behavior occurs depending upon the value of $q$. If $q \geq 1/2$, then after a few iterations every node in the system turns OFF unless none of the OFF nodes is in a group including a sufficient number of OFF nodes (dependent upon $k$), in which case the density of ones reaches a constant. In this situation, the actual values of the nodes change, but the density
Network evolution, $N=100$, $k=2$, weights = [0.33333, 0.33333, 0.33333], $q = 0.5$

![Network evolution diagram](image)

**Figure 9.** Pattern formation plot and corresponding fraction of active nodes for a network with $N = 100$ nodes, connectivity $k = 3$, and equal weights of the input nodes. Here $q = 0.5$.

Observe that the network reaches a frozen state, so it is ordered or stable in just a couple of time steps. In this scenario, it is only the isolated nodes (e.g., an ON node among OFF neighbors) that switch to the state of the majority around them, so a fixed state of the network is reached very quickly.

This phenomenon can be seen in Figure 11. This is a shifting pattern that indicates a shift of happiness and sadness throughout the network. However, at any given time, the number of happy individuals is the same. In Figure 12 we are including another sample pattern with the same parameters as in Figure 11, but with a different initial condition. We can see that in this case the network reaches consensus after about 50 time steps. There is a gradual switch from sadness to happiness throughout the network. This type of behavior is typical for these parameters. There is a gradual shift to consensus (either 1 or 0). For $q < 1/2$, every node in the system turns ON
Network evolution, \(N = 100\), \(k = 4\), weights = \([0.17401, 0.251, 0.2317, 0.1416, 0.2017]\), \(q = 0.53171\)

\[
\begin{array}{ccccc}
\text{0} & \text{0.63} & \text{0.635} & \text{0.64} & \text{0.645} \\
\text{5} & \text{0.65} & \text{0.65} & \text{0.65} & \text{0.65} \\
\text{10} & \text{0.65} & \text{0.65} & \text{0.65} & \text{0.65} \\
\text{15} & \text{0.65} & \text{0.65} & \text{0.65} & \text{0.65} \\
\text{20} & \text{0.65} & \text{0.65} & \text{0.65} & \text{0.65} \\
\end{array}
\]

\[t \rightarrow p(t)\]

Density of ones

Figure 10. Pattern formation plot and corresponding fraction of active nodes for a network with \(N = 100\) nodes, connectivity \(k = 5\), and weights of the input nodes as specified in the first title. Here \(q \sim 0.53\). The middle node has the greatest weight. In this case, in order for a node to maintain its own state at sequential time steps, at least three nodes in its neighborhood need to be in the same state.

as seen in Figure 13, unless every ON node is isolated. If every ON node is isolated, then the same occurrence as previously mentioned takes place. If every individual in a network places half of their influence on another individual in their sphere of influence, then either consensus will be reached in the network, or if everyone is isolated in their behavior, the individuals will be continually changing their behavior according to the individuals around them, but a constant density of a happiness will be achieved. This would be the case of the iterations shown in Figure 8.

In summary, if an equal weight is placed on each individual in a neighborhood, then with large \(k\) values consensus is more easily reached. With smaller \(k\) values, the system reaches a frozen state. When the individuals in the system place the greatest weight
Figure 11. Pattern formation plot and corresponding fraction of active nodes for a network with \( N = 100 \) nodes and parameters as specified in the first title. Here \( q > \frac{1}{2} \), although it is still close to \( \frac{1}{2} \). Observe that the density of ones is constant, but the state of the nodes continually shifts.

on themselves, then they are less likely to change behaviors (states) and a frozen state is reached. Consensus is unlikely in this situation. Finally, if another individual (besides themselves) has the greatest weight, then the system reaches consensus rather quickly, except for the case where every ON/OFF node is isolated, then a constant density of ones is reached, but the actual states of the nodes is constantly changing uniformly. Overall, the system seems to reach a stable behavior, either a frozen state or a periodic evolution in the long run, regardless of the parameters used. We will support this conclusion in the next section, by calculating Lyapunov exponents paired with the asymptotic behavior of the map (5).
Network evolution, $N=100$, $k=7$, weights = $[0.5, 0.083333, 0.083333, 0.083333, 0.083333, 0.083333, 0.083333]$, $q = 0.53$

**Figure 12.** Pattern formation plot and corresponding fraction of active nodes for a network with $N=100$ nodes and parameters as specified in the first title. Here $q > 1/2$, although it is still close to $1/2$. Observe that consensus is reached after a gradual shift from sad to happy through the network.

4.3. **Bifurcations and Lyapunov Exponents.** In this section we support the previous findings with Lyapunov exponent calculations for various parameters, paired with bifurcation diagrams. A bifurcation diagram is a graphical tool to show the birth, evolution, and death of attracting sets for a map [2]. Bifurcations are significant changes that can occur in the fixed points or periodic orbits as parameters are varied. To generate a bifurcation diagram, one starts with a parameter value for the parameter under investigation on the horizontal axis, and a large number of initial points of orbits. The orbits are iterated a large number of times to surpass
Figure 13. Pattern formation plot and corresponding fraction of active nodes for a network with $N = 100$ nodes and parameters as specified in the first title. Here $q < 1/2$, although it is still close to 1/2. The network reaches consensus (happiness in this case).
comparing the two diagrams the LyE is positive and at the same point the bifurcation
diagram seems to be randomly filling up space, then we can quite confidently conclude
that the system has chaotic orbits for that particular parameter value.

Writing the model as a map on \([0, 1]\) we have

\[
 f(p) = \sum_{s=1}^{k-1} p^s (1 - p)^{k-s} \left( \sum_{i_j < i_2 < \cdots < i_s, i_j \neq \frac{k+1}{2}} \chi_{(q, \infty)} \left( \sum_{j=1}^{s} w_{ni_j} \right) \right) +
\]

\[
 \left(1 - \chi_{(q, \infty)}(w_{n+\frac{1}{2}}) \right) \sum_{s=1}^{k-1} p^{s+1} (1 - p)^{k-1-s} \left( \sum_{i_j < i_2 < \cdots < i_s, i_j \neq \frac{k+1}{2}, j=1,2,\ldots,s} \chi_{(q-w_{n+\frac{1}{2}}, \infty)} \left( \sum_{j=1}^{s} w_{ni_j} \right) \right) +
\]

\[
p \chi_{(q, \infty)}(w_{n+\frac{1}{2}}).
\]

Then the derivative of this function can be written as follows:

\[
 f'(p) = \sum_{s=1}^{k-1} p^{s-1} (1 - p)^{k-2-s} \left[ (1 - p)(s - kp) \sum_{i_j < i_2 < \cdots < i_s, i_j \neq \frac{k+1}{2}, j=1,2,\ldots,s} \chi_{(q, \infty)} \left( \sum_{j=1}^{s} w_{ni_j} \right) \right] +
\]

\[
p(s + 1 - kp) \sum_{i_j < i_2 < \cdots < i_s, i_j \neq \frac{k+1}{2}, j=1,2,\ldots,s} \chi_{(q-w_{n+\frac{1}{2}}, \infty)} \left( \sum_{j=1}^{s} w_{ni_j} \right) \right].
\]

This function is used in the numerical results below.

We are first interested in the effect of \(q\) on the behavior of the system. Figure 14
is typical for bifurcation diagrams along \(q\). Here we fix all parameters except \(q\). The
values of the fixed parameters are \(w = [0.2060 0.1504 0.2312 0.2385 0.1738]\), \(k = 5\),
and we allow \(q\) to take on several values between 0.2 and 0.8. For each of these values
we generate 100 iterations of the orbit of \(p(0) = 0.1\) and plot the last points of the
orbits. The result is expected: LyE= 0 and all the orbits are attracted to 0 or 1.

In the previous sections we have discussed the case where the network does not
reach consensus but instead has a constant density with changing states; see Figures
8 and 12. The corresponding LyE and bifurcation diagram is in Figure 15. We can see in this diagram that there is no convergence towards consensus as expected.

Similarly, we study the effect of $k$ on the long-run behavior of the system. In this case, for each value of the parameter $k$ we (randomly) generate one set of weights that are used in both the LyE calculation as well as the bifurcation diagram. We observe that the bifurcation diagrams along the parameter $k$ are similar to those of Figure 14, suggesting that a change in the connectivity does not have a qualitative impact on the long-run behavior of the system. Therefore we do not include sample graphs.

4.4. Focus on Fixed Points. Recall from the introduction that a fixed point is a point that does not change when it undergoes iterations of the system. In other words, a fixed point, $p$, occurs when $f(p) = p$, and thus $f(f(p)) = p$ and so on. The
Figure 15. LyE and Bifurcation diagram along the parameter \( q \) from 0.4 to 0.6 for \( k = 7 \).

\( \alpha = 1/2 \), and \( w = [1/2 \ 1/12 \ 1/12 \ 1/12 \ 1/12 \ 1/12 \ 1/12] \). No consensus is reached. However, the dynamics are not chaotic.

orbit then of a fixed point would be \( \{p, p, p, \ldots\} \) or just \( \{p\} \). If \( |f'(p)| > 1 \) the fixed point is unstable. If \( |f'(p)| < 1 \), then the fixed point is stable.

So let us look more closely at the fixed points of the map (6). If we replace \( p = 0 \) in the map, we can see immediately that \( f(0) = 0 \) since \( p \) occurs at a power greater than or equal to 1 in all terms. Thus \( p = 0 \) is a fixed point. Then \( f'(0) = 0 \) immediately from the formula (7). Thus \( |f'(0)| < 1 \), which means that 0 is an attracting fixed point. Similarly, if we replace \( p = 1 \) in (6), we see that all the terms are zero except the ones corresponding to \( s = k - 1 \) in the second and the third terms of that formula, which leads to

\[
 f(1) = \left(1 - \chi(q, \infty)(w_n^{k+1})\right) \sum_{i_1 < i_2 < \ldots < i_k - 1} \chi(q-w_n^{k+1}, \infty) \left(\sum_{j=1}^{s} w_{nij}\right) + \chi(q, \infty)(w_n^{k+1}).
\]
But observe that the sum over \( k - 1 \) elements of the vector \( w_n = (w_{n1}, w_{n2}, \ldots, w_{nk}) \), with the middle term removed, is simply the sum of all the weights except the middle one, and is equal to \( 1 - w_{n,\frac{k+1}{2}} \). Thus

\[
 f(1) = \left(1 - \chi(q,\infty)(w_{n,\frac{k+1}{2}})\right) \chi(q-w_{n,\frac{k+1}{2}},\infty) \left(1 - w_{n,\frac{k+1}{2}}\right) + \chi(q,\infty)(w_{n,\frac{k+1}{2}}).
\]

But since \( q < 1 \Rightarrow q - w_{n,\frac{k+1}{2}} < 1 - w_{n,\frac{k+1}{2}} \) so that \( \chi(q-w_{n,\frac{k+1}{2}},\infty) \left(1 - w_{n,\frac{k+1}{2}}\right) = 1 \). It follows that \( f(1) = 1 \) and \( p = 1 \) is a fixed point. Now, we know from Section 4.1, that there are other fixed points as well (the graph of the map intersects the main diagonal at points in \((0,1)\). Given the complicated formula for the map, instead of actually computing the fixed points (which is computationally prohibitive), we try to at least understand the nature of those points: stable or unstable. Using the MatLab code 

**fixedpoints_fprime_SocialSpread.m** the derivative \( f'(p) \) is plotted along with the lines -1 and 1. If a fixed point occurs for values of \( p \) that lead to \( f'(p) \) being between these two lines, then that fixed point, \( p \) is stable. In Figure 16 we show a typical situation. Observe that \( f'(0) = f'(1) = 0 \) which means that \( p = 0 \) and \( p = 1 \) are both stable. Of course they have disjoint basins of attraction. On the other hand, if another fixed point occurs in an interval for which the plot of \( f' \) is above 1, then that point is unstable. So far we have not been able to show analytically that any other fixed point is unstable, so that only consensus can occur. However, all the simulations lead to that conclusion.

### 5. Robustness to Noise

Real networks do not simply follow fixed rules exactly as we see in CA and BN. Instead, real systems undergo disturbances and perturbations due to environmental
Figure 16. Plot of the function $f$ together with the lines through $\pm 1$. Any fixed point $p$ for which $|f'(p)| < 1$, meaning that the point on the graph falls between the two lines, is stable.

Observe that $p = 0$ and $p = 1$ are both stable. Any point for which $|f'(p)| > 1$ is unstable.

changes or the like. They have the ability to reach functional diversity, and aim to maintain the same state under environmental noise. Hence it is very important to submit this system to these disturbances, or noise. Introducing noise to a system can greatly impact the outcome of the system. For example, in [15] it is shown that for a BN governed by a generalized ECA rule 126, certain type of noise can either reduce the amount of chaos in the system or it can completely abolish chaos. In [4] the authors apply a particular random noise procedure typical for neural networks, to a Boolean network governed by the extended generalized ECA rule 22. It is shown that there is no critical value of the noise parameter that differentiates between ordered and random behavior of the system.

It is known that the study of the robustness of a Boolean network to various types of perturbations is an important aspect of the evolution of systems under Boolean models, since the systems have to respond and adapt to disturbances which could be simple random fluctuations inherent to the system. For example, random fluctuations in genetic networks are inevitable as chemical reactions are probabilistic and many
genes or proteins are present in low numbers per cell. Such disturbances affect all life processes. The presence of a small amount of noise (drug) can positively enhance a cancer treatment, driving the system to a state of tumor extinction. In such a case one would like to force the system into a desired ordered regime. On the other hand there could be situations in which one would like to force the system into a chaotic regime. For instance, typically developed babies around 8 months exhibit a chaotic movement of their center of pressure when placed on a measuring device, due to multiple degrees of freedom as they become more sure on themselves. On the other hand, babies suffering from cerebral palsy, would exhibit a more ordered behavior and thus their treatment would require noise (physical therapy) that allow them to develop multiple degrees of freedom (e.g. Harbourne and Stergiou [38]).

So does the BN transition to a different dynamical behavior under noise? Thus, is the network sensitive to noise or not? We will explore the answers to these questions in what follows.

Inducing disturbance in the system by changing the value of certain nodes in the network (according to some deterministic or random rule) is a good model for an environmental or intrinsic type of perturbation. We consider the following environmental noise procedure, called the “flip rule”. It has been used before in [15], where it is shown that the noise can actually induce stability in a network evolving under a generalized ECA rule 126. According to the “flip rule” at each time point \( t \) we randomly select \( j \) nodes of the network and flip their value (from 0 to 1 or from 1 to 0). In this case, the parameter \( j \) can be generated in a deterministic or stochastic fashion. An equivalent noise procedure has been used in [4], where it is shown that
under certain conditions, a network governed by a generalized ECA rule 22 may exhibit a phase transition at a critical value of the noise intensity parameter $\eta_c$. On the other hand, there are situations when a phase transition may not exist. This type of noise has been also considered by Huepe and Aldana-González [19] in the case of a neural network model with noise.

We need to understand how the model 6 is affected by the noise. In other words, after application of the noise, the density of ones $p$ may change to new value $p_{\text{new}}$ which is the actual input to the next iteration of the map. To determine $p_{\text{new}}$ we observe that if $j$ nodes are chosen at random, then $j \cdot p$ of them are in state 1 and $j \cdot (1 - p)$ are in state 0. By the “flip rule”, the total number of nodes in state 1 is decreased by $j \cdot p$ since they are changed to 0. On the other hand the number is increased by $j \cdot (1 - p)$ since the zeros become ones. Thus, the proportion of nodes in state 1, that is $p$, is modified as follows [15]:

$$p_{\text{new}} := p - \frac{j \cdot p}{N} + \frac{j \cdot (1 - p)}{N} = p + \frac{j}{N} (1 - 2p).$$

Observe that $p_{\text{new}} \geq 0$ since all the terms are nonnegative in the above computation and since $j < N$. We also have that $p_{\text{new}} \leq p + (1 - 2p) = 1 - p \leq 1$. Thus $0 \leq p_{\text{new}} \leq 1$.

Once $p_{\text{new}}$ is computed the network is again subject to the Boolean rules. Therefore the new density of ones map is as follows.

**Proposition 1.** Consider a Boolean network with $N$ nodes governed by the Boolean rules described in this work. If at each time point $t$ we perturb $j$ nodes according to the “flip rule”, then the probability that a node is ON (i.e., the node is in state 1) at
time \( t + 1 \) is given by the map:

\[
(8)\quad f(p) = \sum_{s=1}^{k-1} \left( p + \frac{j}{N} (1 - 2p) \right)^s (1 - p - \frac{j}{N} (1 - 2p))^{k-s} \sum_{\substack{i_1 < i_2 < \ldots < i_s \in \{1, \ldots, k+1\} \setminus \{j\}, j=1,2,\ldots,s}} \chi_{(q,\infty)} \left( \sum_{j=1}^{s} w_{ni_j} \right) + \\
(1 - \chi_{(q,\infty)}(w_{n+1})) \sum_{s=1}^{k-1} \left( p + \frac{j}{N} (1 - 2p) \right)^{s+1} (1 - p - \frac{j}{N} (1 - 2p))^{k-1-s} \sum_{\substack{i_1 < i_2 < \ldots < i_s \in \{1, \ldots, k+1\} \setminus \{j\}, j=1,2,\ldots,s}} \chi_{(q-w_{n+1},\infty)} \left( \sum_{j=1}^{s} w_{ni_j} \right) + \\
(p + \frac{j}{N} (1 - 2p)) \chi_{(q,\infty)}(w_{n+1}).
\]

Note here that an equivalent way of understanding this noise procedure is as follows:

at each time step, each node is flipped with probability \( j/N \) and left unchanged with probability \( 1 - j/N \).

From Formula 8 we can see immediately that neither \( p = 0 \) nor \( p = 1 \) are fixed points, which means that consensus is not reached. First we look at some pattern formation plots. In Figure 17 we show sample plots for a network with \( N = 100 \) nodes, connectivity \( k = 7 \), and equal weights of the inputs. The threshold is set to \( q = 0.53 \) in this figure, and at each time step exactly \( j = 10 \) nodes are randomly chosen and their values flipped. The network is evolved 200 time steps. Observe that the network does not reach consensus; however the plots suggest that the network reaches some sort of balanced behavior, with a bias towards either happiness or sadness in the long run. We note that some clusters of happiness or sadness are also possible. An increase in \( j \) tends to induce more “random” patterns (like in the middle column of Figure 17), while an increase in the weight of one of the inputs generates a more random version of the diagonal pattern seen in Figure 12. Such a sample is shown in Figure
Figure 17. Pattern formation plots for a network with the following parameters: $N = 100, k = 7, q = 0.53, j = 10,$ and equal weights of the inputs. Each pattern corresponds to a different initial state of the network. The patterns seem to converge to a balanced behavior biased either to happiness or sadness.

Figure 18. Pattern formation plots for a network with the following parameters: $N = 100, k = 7, q = 0.53, j = 10,$ and a larger weight for one of the inputs. This pattern is a basically a random version of the one in Figure 12 which has the same parameters.

18. Finally, a random selection of inputs generates patterns similar to Figure 19 for small values of $j/N$ and to Figure 20 for larger values of $j/N$. Note that for the latter case small clusters of happiness or sadness are formed but they shift throughout the network in a random like fashion.
Let us focus on LyE and bifurcation diagrams. From the bifurcation diagrams along the parameter $q$, which are similar to Figure 14, we note the occurrence of two stable fixed points at $p = j/N$ or $p = 1 - j/N$ regardless of the value of $q$ or $k$. A typical graph is shown in Figure 21 where $k = 5$ and the weights are all equal. In this case, the map has two stable fixed points and the range of $q$ is divided into
the corresponding basins of attraction of the two fixed points. This supports the results seen in the patterns of Figure 17, where the network is biased either towards happiness or sadness depending on the initial state.

Larger values of \( j \) reveal further fixed points as seen in Figure 22. Note that besides the two fixed points mentioned before, \( p = 1/2 \) is another one. Although the actual density of ones computed from iterating the network may not converge to a fixed value, it fluctuates around the values suggested by the bifurcation diagrams. The model simplifies the reality, so this kind of situation is to be expected. However, the model can give us a good idea of the approximate long-run behavior of the network.

In Figure 23 we show a sample bifurcation diagram for a network with \( k = 5, j/N = 0.25 \) and random weights of the inputs. Note immediately the more complexity of the bifurcation diagram, with a variety of fixed points. Each of them has its own basin of attraction indicated on the \( q \) axis. This situation corresponds to the pattern seen in Figure 20.
Figure 22. LyE and bifurcation diagram for a noisy system with $k = 5, j/N = 0.1$, and equal weights of the inputs. Observe the existence of three stable fixed points $p = j/N$, $p = 1 - j/N$, and $p = 1/2$.

Figure 23. LyE and bifurcation diagram for a noisy system with $k = 5, j/N = 0.25$, and random weights of the inputs. Observe the existence of multiple stable fixed points besides $p = j/N$, and $p = 1 - j/N$.

The bifurcation diagrams suggest that we are dealing with multiple fixed points, and a more complex behavior of the system. The impact of noise will be further explored in future work. Given the various parameters it will be of interest to understand
precisely what is the impact of each parameter. The simulations performed suggest
that the system does not exhibit a transition from stability to chaos under the current
noise procedure, and therefore it is robust to noise. The actual type of stability may
change though.

6. CONCLUSIONS AND DIRECTIONS FOR FUTURE WORK

In this thesis we consider a Boolean network model for a social network and explore
the spread of happiness throughout the network. Each node (person) is influenced
by several nearest neighbors that may have different impacts on the node. The nodes
are updated based on a threshold function. We provide a mean-field mathematical
model for the density of ones and show that it is a good fit under certain parameter
combinations. We show that under the particulars specified in this work, the network
reaches mostly stability: all people are happy or all are sad, so there is consensus.
Under a simple noise procedure, the network does not reach consensus anymore;
however there is a convergence towards a more or less stable fraction of happy people
in the network. The convergence to a stable behavior may take a few hundred steps
of iterations. Clusters of happiness or sadness may form in the network.

There are several directions of investigation that are a natural continuation of this
research. First, the nearest neighbor assumption can be dropped and the nodes in a
neighborhood be ordered according to their weights. The sizes of the neighborhoods,
that is the connectivity, can be varied from one node to another, and even during
the evolution of the network. This would account for people getting in contact with
other people during the same event. On the other hand, the weights of the nodes
in a neighborhood need not be constant over time, even though the neighborhoods
may be. This accounts for changes in the impact that various people may have over one’s opinion as time goes by. By the same token, the thresholds can be chosen dynamically.

On the other hand, accounting for more noise in the network is an essential step. This can be done by a more sophisticated random procedure for modifying artificially the values of chosen nodes, or could be driven by asynchrony in the network. It is known that an asynchronous update of the nodes may lead to a significantly different behavior of the overall system. At the same time, one could account for “mutations” in the social network. For example, some people may be very stubborn in their own opinion, so they place all the weight on themselves. These are frozen nodes in the network. But new nodes in the network may induce even further perturbations. One can use a simple procedure for attachment of new nodes to old ones (that is their neighborhoods). This way, the connectivity of the network may evolve from random connections, to say a power-law distribution of the connectivity. Moreover, one could consider the impact of “leaders”, and the time it takes the network to reach consensus.

Another possible direction for future research is the choice of more than two possible states. This goes beyond BN, but one can use similar approaches for multiple states of the network. It would be interesting to look at the possible clustering effect of people of the same opinion.
REFERENCES


[39] Laubenbacher, R., Murrugarra, D., & Veliz-Cuba, A. Structure and dynamics of polynomial dynamical systems N.SF grant report, University of Nebraska Lincoln.


Appendix: Matlab Codes

In this appendix we include the Matlab codes used to generate the figures in this thesis.

(1) iterations_SocialSpread.m

This Matlab code makes use of the following Matlab functions:

parents_CA_includingthenode.m,
one_network_iteration_SocialSpread.m, model...
SocialSpread.m, SumWeightsByThreshold.m.

The functions will be listed at the end of this code.

% "iterations_SocialSpread.m"
% Generate iterations of the system and the model
% for mean field approximation of the social spread...
% model and network

clear;
% initialize parameters
N = 100; %network size
kparam = [2 6 12 20]; %number of inputs
IT =19; %gives IT+1 iterations
fontsize=10;
numlines = 2; %number lines for subplots
numcolumns = 2; %number columns for subplots
initialproportions = (1:3:N)/N;  \%initial values...
of orbits

numberones = 1:3:N;

xt = N;  \% #nodes to be updated at the same time;

xt=N means synchrony

for k = kparam

clear pa W

\%W = (1/(k+1))*ones(1,k+1);  \%equal weights...
- all nodes have the same

W = random('Normal',100,20,1,k+1);  \%unequal...
weights - Normal pdf values

for j=1:length(W)
    if W(j) <= 0
        W(j) = 0.01;
    end
end

W = W/sum(W);

Q = W((k+2)/2):0.1:0.7;  \%threshold > weight...
of the middle node

\%Q = 0.1:0.2:0.9;

\%pa = parents_random(N,k,N);

pa = parents_CA_includingthenode(N,k,N);  \%CA:...
k adjacent nodes
for q = Q
\%
\% pa = parents_random(N,k,N); \% activated...

here if we want a new set of
\%
\% parents for each q as well.
\%
\% display([’k=’,num2str(k),’ ,q=’,num2str(q)]);
\%
\%
figure
\%
\% numplot = 1;
\%
\% clear v prob

\%
\% simulate the network
\%
\%
prob = cell(1,IT+1);
\%
\% numberrepetitions = 1000; \% number of initial...
\%
\% conditions to average
\%
\% L=1; \% this flags where we are in the...
\%
\% computation =
\%
\% number of initial conditions
\%
\%
for i = numberones
\%
\% tempmatrix = combnk(1:N,i);
\%
\% [linestemp, colstemp] = size(tempmatrix);
\%
\% prob{1,1}(L) = i/N;
\%
\% if linestemp <= numberrepetitions
\%
\% for step = 1:IT+1
\% clear tempvector Tempvector v tempprob
for j = 1:linestemp
    tempvector = zeros(1,N);
    tempvector(tempmatrix(j,:)) = 1;
    Tempvector = ...
    one_network_iteration_SocialSpread...
    (tempvector,pa,W, q);
    tempprob(j) = sum(Tempvector)/N;
end
prob{1,step+1}(L) = mean(tempprob);
end
else
    temprand = randperm(linestemp);
    Tempmatrix = tempmatrix(temprand...
    (1:numberrepetitions),:);
    clear tempmatrix
tempmatrix = Tempmatrix;
[linestemp, colstemp] = size(tempmatrix);
for step = 1:IT+1
    clear tempvector Tempvector v tempprob
    for j = 1:linestemp
        tempvector = zeros(1,N);
        tempvector(tempmatrix(j,:)) = 1;
        Tempvector = ...
        one_network_iteration_SocialSpread...
(tempvector, pa, W, q);

    tempprob(j) = sum(Tempvector)/N;
end

    prob{1,step+1}(L) = mean(tempprob);
%we average over a number of initial...
    conditions
end
end

L=L+1
end

[m,n] = size(prob);
for u=1:n
    prob{1,u} = [0 prob{1,u}];
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% simulate the model
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

p = sort(prob{1,1}); %initial values model...
    (same w/ the system)
p1 = model_SocialSpread(p, W, q); %application...
    of the model
saveiterate(1,1:length(p1)) = p1;
subplot(numlines,numcolumns,numplot);
plot(p,saveiterate,'r','LineWidth',2);
hold;

plot(prob{1,1},prob{1,2},'.','MarkerSize',5);
X=prob{1,2}; Y=saveiterate;
Sxy = sum((X-mean(X)).*(Y-mean(Y)));  
Sxx = sum((X-mean(X)).^2);
Syy = sum((Y-mean(Y)).^2);
r = Sxy/(Sxx*Syy)^(1/2);

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

xlabel('p(t)','FontWeight','bold','FontSize',fontsize);
ylabel('p(t+1)','FontWeight','bold','FontSize',fontsize);
title(['w=',num2str(W(1)),',',num2str(W(2)),',',num2str(W(3)),',r=',num2str(r)],
      'FontWeight','bold','FontSize',fontsize);
hold off;

numplot = numplot+1;

for it=2:IT+1
    p1 = model_SocialSpread(p1, W, q);
saveiterate = p1;

    plot the system vs model for various iterations
    if (it == 2) || (it == 3) || (it == IT+1)
    subplot(numlines,numcolumns,numplot);
    plot(p,saveiterate,'r','LineWidth',2);
    hold;
plot(prob{1,1},prob{1,it+1},'.','MarkerSize',5);
X=prob{1,it+1}; Y=saveiterate;
Sxy = sum(((X-mean(X)).*(Y-mean(Y))));
Sxx = sum(((X-mean(X)).^2));
Syy = sum(((Y-mean(Y)).^2));
r = Sxy/(Sxx*Syy)^{(1/2)};

xlabel('p(t)','FontWeight','bold','FontSize',fontsize);
ylabel(['p(t+',num2str(it),')'],'FontWeight','bold',...
'FontSize',fontsize);
title(['k=',num2str(k),' ,q=',num2str(q),',r=',num2str(r)],...
'FontWeight','bold','FontSize',fontsize);

axis([0 1 0 1]);

numplot = numplot+1;
end

end

end %dindex

end %kindex

(2) parents_CA_includingthenode.m

% "parents_CA_includingthenode.m"

%pa = parents_CA_includingthenode(N,k,M)

%N=number nodes, k=number parents, M=#nodes w/ k parents
%generate the k nearest nodes as parents
%like in cellular automata

function pa = parents_CA_includingthenode(N,k,M)

pa = cell(1,N); u = [1:N 1:N 1:N]; for n = 1:M(1)
    pa{1,n} = [u(n+N-floor(k(1)/2):n+N-1) n ... 
        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)
            pa{1,n} = [u(n+N-floor(k(1)/2):n+N-1) n ... 
                u(n+N+1:n+N+k(1)-floor(k(1)/2))];
        end
    end
end

(3) one_network_iteration_SocialSpread.m

% "one_network_iteration_SocialSpread.m"

% Inputs:
% inputvector = the initial state, a vector
% parents = neighborhood of each node, obtained with...
% a different
% function
% W = weights, q = threshold for the rule
function outputvector = ...
    one_network_iteration_SocialSpread(inputvector,...
    parents,W,Q)

for index=1:length(inputvector)
    inputnodes = parents{1,index};
    flag = 0;
    for j = parents{1,index}
        if j == index
            flag = 1;
        end
    end
    if flag == 0
        inputnodes = [inputnodes index];
    end
    inputnodes = sort(inputnodes);
    t = inputvector(inputnodes);
    if sum(W.*t) > Q
        outputvector(index) = 1;
    else
        outputvector(index) = 0;
    end
end
(4) model_SocialSpread.m

%model_derivative_SocialSpread.m
%function that yields p(t+1) given p(t)
%poftplus1 = model_SocialSpread(inputvector,...

    numberinputs, weights, threshold)

function derivative = model_derivative...
_SocialSpread(inputvector, weights, threshold)

p = inputvector;
W = weights;
Q = threshold;
K = length(W);

[m,n] = size(p);
for i=1:m
    for ii = 1:n
        clear binomial
        for s = 1:K-1
            Sum1(s) = SumWeightsByThreshold...
                (s,W,Q);
            Sum2(s) = SumWeightsByThreshold...
                (s,W,Q-W((K+1)/2));
            binomial(s) = p(i,ii)^(s-1)*...
\begin{align*}
(1-p(i,ii))^{(K-2-s)} & \ldots \\
((1-p(i,ii))*(s-K*p(i,ii)) & \ldots \\
\text{Sum1}(s)+p(i,ii) & \ldots \\
(s+1-K*p(i,ii))*\text{Sum2}(s) & \); \\
\end{align*}

\text{end}

derivative(i,ii) = \text{sum(binomial)};
\text{end}
\text{end}

(5) \textit{SumWeightsByThreshold.m}

\%\textit{SumWeightsByThreshold.m}

\%function that computes sums of weights by threshold... for the Social Spread

\%\textit{SumWeights} = \textit{model_SocialSpread}(\textit{inputvector}, \textit{numberinputs}... , \textit{weights}, \textit{threshold}, \textit{numberupdatednodes})

function \textit{SumWeights} = \textit{SumWeightsByThreshold}(\textit{sumlevel},... \textit{weights}, \textit{threshold})

\text{s} = \text{sumlevel};
\text{W} = \text{weights};
\text{Q} = \text{threshold};

\text{k} = \text{length(W)};
\text{w} = [W(1:(k+1)/2-1) W((k+1)/2+1:length(W))]; \%\text{eliminate the...
middle weight

\[ C = \text{sum}\left(\text{combnk}(w, s), 2\right); \]

\[ [m, n] = \text{size}(C); \]

\[ q = Q \ast \text{ones}(m, n); \]

for \( i=1:m \)

\hspace{1em} for \( j=1:n \)

\hspace{2em} if \( \text{abs}(q(i, j) - C(i, j)) \leq 0.000001 \)

\hspace{3em} \( C(i, j) = q(i, j); \)

\hspace{1em} end

end

end

\[ \text{SumWeights} = \text{sum}(\text{lt}(q, C)); \]

(6) **PatternFormation_SocialSpread.m**

This Matlab code makes use of the following Matlab functions:

- `parents_CA_includingthenode.m`,
- `one_network_iteration_SocialSpread.m`,
- `randominitialstate.m`.

The functions are listed above and below.

```
\% Program PatternFormation_SocialSpread.m

clear;clf;

\%Parameters to play with:

N = 100; \%#nodes

k = 2; \%#parents for each node ...
(neighborhood size)

\[
\text{it} = 20; \quad \% \text{#iterations}
\]

\[
W = \left( \frac{1}{(k+1)} \right) \times \text{ones}(1, k+1);
\]

\% equal weights - all nodes have the same

\[
W = \left[ \frac{1}{4} \ 2/4 \ 1/4 \right]; \quad \% \text{example of unequal} \ldots
\]

weights for 3 parents

\[
Q = \frac{1}{3}; \quad \% \text{the threshold for the rule} - \ldots
\]

the same for all nodes

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\% plotting parameters

\[
\text{marker} = 2; \quad \text{fsize} = 14; \quad \text{numlines} = 2; \ldots
\]

\[
\text{numcolumns} = 1; \quad \text{numplot} = 1;
\]

\[
\text{linewidth} = 2;
\]

\% generate the neighborhood for each node

\[
\text{parents} = \text{parents}\_CA\_including\_thenode(N, k, N);
\]

\% CA: k adjacent nodes

\[
\text{parents} = \text{parents}\_random(N, k, N);
\]

\% random choice of k parents

\% generate the initial state of the network

\[
A = \text{zeros}(\text{it}, N); \quad A(1, \text{floor}(N/2)) = 1;
\]

\% one black cell
A = randominitialstate(N);

% random initial state
p(1) = sum(A(1,:))/N;

% iterate the network and count the ON ... cells at each step
for j=2:it A(j,:) =
    one_network_iteration_SocialSpread(A(j-1,:),
    ,parents,W,Q); p(j) =
    sum(A(j,:))/N; end

% adjust the matrix for plots
A=A'; B=[]; [d1,d2]=size(A); for i=1:d1
    B(i,:)=A(N-(i-1),:);
end

% plot the pattern formation and the ... density of ones
subplot(numlines,numcolumns,numplot); for j=1:d1
for i=1:d2
    if B(j,i) == 1
        plot(d1-(j-1),d2-(i-1),'sk',...
        'MarkerSize',marker, ...
        'MarkerEdgeColor','k', 'MarkerFaceColor...
        ','k');
hold on;
else
    plot(d1-(j-1),d2-(i-1), 'sy', '...
MarkerSize', marker, ...
'MarkerEdgeColor', 'y', 'MarkerFaceColor', 'y');
end
end
end
axis equal; axis([1 N 1 it]); axis off;

title(['Network evolution, N=', num2str(N), ', k=', num2str(k),...
     ', weights = [', num2str(W), '], q = ', num2str(Q), ']
     'FontSize', 8, 'FontWeight', 'bold');

numplot = numplot+1; subplot(numlines, numcolumns, numplot);
plot(p, 'LineWidth', linewidth);
xlabel('t', 'FontSize', fsize, 'FontWeight', 'bold');
ylabel('p(t)', 'FontSize', fsize, 'FontWeight', 'bold');
title('Density of ones', 'FontSize', fsize, 'FontWeight...
     ', 'bold');

(7) randominitialstate.m

% "randominitialstate.m"

% Inputs:
%
% N = number of nodes

function initialstate = randominitialstate(N)
numberblackcells = unidrnd(N); x = randperm(N);...
blackcells =
x(1:numberblackcells); initialstate = zeros(1,N);
initialstate(blackcells) = ones(1,numberblackcells);

(8) cobwebplot_SocialSpread.m

This Matlab code makes use of the following Matlab function:

model_SocialSpread.m.

The function is listed above.

% Program "cobwebplot_SocialSpread.m"
% Graphical illustration of the iterations

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Can be used for other maps as well and in that case
% the lines marked with %%%%% need to be changed accordingly
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear;
k = 14;
W = random('Normal',100,20,1,k+1); %unequal...
weights - Normal pdf values
for j=1:length(W)
    if W(j) <= 0
        W(j) = 0.01;
    end
end

%clear W; W = (1/(k+1))*ones(1,k+1);
%equal weights - all nodes have the same
W = W/sum(W);

%Q = W((k+2)/2):0.01:0.7; %threshold > weight...
    of the middle node
%Q = 0.1:0.2:0.9;
q = .5; %%%%% number of iterations
xinitial = input('Enter the initial value of x...
(between 0 and 1) = ');
linwidth=2;

%generate consecutive iterations of f
x = xinitial;
y = 0;
xx = x;

yy = model_SocialSpread(xx, W, q); %%%%%
x = [x xx];
y = [y yy];
for i = 3:20
if mod(i,2) == 0
    yy = model_SocialSpread(xx, W, q);
    x = [x xx];
    y = [y yy];
else
    xx = yy;
    x = [x xx];
    y = [y yy];
end
end

x0 = linspace(0,1,100); % range of initial values
y0 = model_SocialSpread(x0, W, q);

clf;
plot(x0,y0,'LineWidth',linwidth);
hold on;
plot(x0,x0,'r','LineWidth',linwidth);
plot(x,y,'k','LineWidth',linwidth);
title(['Cobweb plot for the orbit of x = ',num2str(xinitial)] 'FontWeight','bold','FontSize',14)
xlabel('x','FontWeight','bold','FontSize',14)
ylabel('y','FontWeight','bold','FontSize',14)
hold off;

(9) **LyE_bifurcation_SocialSpread.m**

This Matlab code makes use of the following Matlab functions:

- `model_SocialSpread.m`,
- `SumWeightsByThreshold.m`,
- `model_derivative_SocialSpread.m`.

The functions are listed above and below.

```matlab
% LyE_bifurcation_SocialSpread.m

clear;
IT=1000;  %#iterations
kparam = [2 4 6];  %number of inputs (except the node ... itself) - even numbers
fsise=14;

for k = kparam
    clear W
    %W = (1/(k+1))*ones(1,k+1);  %equal weights -...
    all nodes have the same
    W = random('Normal',100,20,1,k+1);  %unequal weights...
    - Normal pdf values
    for j=1:length(W)
```

```matlab
end
end
```
if $W(j) \leq 0$

$$W(j) = 0.01;$$

end
end

$W = W/\text{sum}(W);$ 

$Q = W((k+2)/2):0.01:0.7; \ %\text{threshold > weight of...}$ 

the middle node 

$%Q = 0.1:0.2:0.9;$

figure

for q = Q

step = 1;

$f = 0.1; \ %\text{initial point that can be changed}$

$h = \log(\text{abs(model_derivative_SocialSpread}(f,W,q)));$

for i=2:IT

$f = \text{model_SocialSpread}(f, W, q);$ 

$h = h + \log(\text{abs(model_derivative_SocialSpread...}$ 

$(f,W,q)));$

end

$A(\text{step}) = q;$

$H(\text{step}) = h/\text{IT};$

$\text{step} = \text{step}+1;$

end

subplot(2,1,1)

plot(A,H)
hold on
x = linspace(Q(1),Q(length(Q)),100);
plot(x, zeros(size(x)),'r');
hold off;
xlabel('q','FontWeight','bold','FontSize',fsize);
ylabel('LyE','FontWeight','bold','FontSize',fsize);
title(['Lyapunov Exponents for k = ',num2str(length(W))],...
'FontWeight','bold','FontSize',fsize)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%bifurcation diagram with the same values of the... 

% varied parameter
numpointsp = 100; %number of points in [0, 1]...
to be iterated
subplot(2,1,2)
p1 = linspace(0,1,numpointsp); %initial points...
in [0, 1]
for q = Q
    p = p1;
    for j = 1:IT
        p = model_SocialSpread(p, W, q);
    end
    A = q*ones(size(p));
    plot(A,p,'.');
hold on;
end
xlabel('q','FontWeight','bold','FontSize',fsize);
ylabel('p(t)','FontWeight','bold','FontSize',fsize);
title(['Bifurcation diagram for k = ',num2str(length(W))],...
     'FontWeight','bold','FontSize',fsize)
hold off;
end

(10) model_derivative_SocialSpread.m

function derivative = model_derivative_SocialSpread...
(inputvector, weights, threshold)

p = inputvector;
W = weights;
Q = threshold;
K = length(W);
[m,n] = size(p);
for i=1:m
    for ii = 1:n
        clear binomial
        for s = 1:K-1
            Sum1(s) = SumWeightsByThreshold(s,W,Q);
            Sum2(s) = SumWeightsByThreshold(s,W,Q-W((K+1)/2));
            binomial(s) = p(i,ii)^(s-1)*(1-p(i,ii))^(K-2-s)*...
            ((1-p(i,ii))*(s-K*p(i,ii))*Sum1(s)+p(i,ii)*...
            (s+1-K*p(i,ii))*Sum2(s));
        end
        derivative(i,ii) = sum(binomial);
    end
end

(11) LyE_bifurcation_SocialSpread_Noise.m
This Matlab code makes use of the following Matlab functions:
model_SocialSpread.m, SumWeightsByThreshold.m,
model_derivative_SocialSpread.m.
The functions are listed above.

% LyE_bifurcation_SocialSpread_Noise.m
% LyE for the logistic map
% graph LyE vs. a
% graph the bifurcation diagram

% Noise: flip rule and stochastic flip rule

clear;
IT=1000; %#iterations
kparam = [2 4 6]; %number of inputs (except the node...
 itself) - even numbers
alpha = 1/2; %alpha = fraction of nodes to be disturbed...
 at each iteration
fsize=14;

for k = kparam

clear W

%W = (1/(k+1))*ones(1,k+1); %equal weights -...
 all nodes have the same
W = random('Normal',100,20,1,k+1);
%unequal weights - Normal pdf values
for j=1:length(W)
    if W(j) <= 0
        W(j) = 0.01;
    end
end
W = W/sum(W);
Q = W((k+2)/2):0.01:0.7; %threshold > weight...
of the middle node

%Q = 0.1:0.2:0.9;

figure

for q = Q
    step = 1;
    f = 0.1; % initial point that can be changed
    h = log(abs(model_derivative_SocialSpread...
            (f,W,q)));
    for i=2:IT
        f = model_SocialSpread(f, W, q);
        h = h + log(abs(model_derivative_Social...
                     Spread(f,W,q)));
    end
    A(step) = q;
    H(step) = h/IT;
    step = step+1;
end

subplot(2,1,1)
plot(A,H)
hold on
x = linspace(Q(1),Q(length(Q)),100);
plot(x, zeros(size(x)),'r');
hold off;
xlabel('q','FontWeight','bold','FontSize',fsize);
ylabel('LyE','FontWeight','bold','FontSize',fsize);

title([['Lyapunov Exponents for k = ',
    num2str(length(W))],
    'FontWeight','bold','FontSize',fsize))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% bifurcation diagram with the same values of the... 
  varied parameter

numpointsp = 100; %number of points in [0, 1] ... 
  to be iterated

subplot(2,1,2)
p1 = linspace(0,1,numpointsp); %initial points ... 
  in [0, 1]

for q = Q

  p = p1;

  for j = 1:IT

    p = model_SocialSpread(p, W, q); 

    p = p + alpha * (1-2*p); %apply the flip rule

  end

  A = q*ones(size(p));

  plot(A,p,'.');

  hold on;

end

xlabel('q','FontWeight','bold','FontSize',fsize);
ylabel('p(t)','FontWeight','bold','FontSize',fsize);
title(['Bifurcation diagram for k = ',num2str...
(length(W))], 'FontWeight', 'bold', 'FontSize', fsize)
hold off;
end
Appendix: Tables

In this appendix we include some supplementary data tables referenced in the text.

Table 1 includes values for varying weights, $k$ values, and $q$ values with the resulting correlation coefficient, $r$, after the twentieth iteration. These are some of the values that were used to decide which parameters of the model are a good fit for the system.

Table 2 is similar to the previous table, but includes larger values of $k$. 
# Table 1. Iterations Trial Values.

<table>
<thead>
<tr>
<th>Weights</th>
<th>q</th>
<th>r</th>
<th>Weights</th>
<th>q</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>.4153</td>
<td>.2489</td>
<td>.3358</td>
<td>.24887</td>
<td>.53742</td>
<td></td>
</tr>
<tr>
<td>.34887</td>
<td>.45285</td>
<td>.2597</td>
<td>.2307</td>
<td>.1665</td>
<td>.2077</td>
</tr>
<tr>
<td>.44887</td>
<td>.90753</td>
<td>.26649</td>
<td>.66652</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.54887</td>
<td>.90766</td>
<td>.36649</td>
<td>.77158</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.64887</td>
<td>.45274</td>
<td>.46649</td>
<td>.91749</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.29792</td>
<td>.30328</td>
<td>.3988</td>
<td>.50328</td>
<td>.90759</td>
<td>.60328</td>
</tr>
<tr>
<td>.30328</td>
<td>.45277</td>
<td>.9073</td>
<td>.1221</td>
<td>.1716</td>
<td>.1278</td>
</tr>
<tr>
<td>.40328</td>
<td>.90839</td>
<td>.22775</td>
<td>.52287</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.2815</td>
<td>.4329</td>
<td>.2856</td>
<td>.43285</td>
<td>.9075</td>
<td>.53285</td>
</tr>
<tr>
<td>.63285</td>
<td>.45175</td>
<td>.52775</td>
<td>.93209</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.4864</td>
<td>.2869</td>
<td>.2267</td>
<td>.28686</td>
<td>.45228</td>
<td>.28686</td>
</tr>
<tr>
<td>.48686</td>
<td>.90761</td>
<td>.26966</td>
<td>.70882</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.58686</td>
<td>.45256</td>
<td>.36966</td>
<td>.90748</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.68686</td>
<td>.45257</td>
<td>.46966</td>
<td>.9338</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.17401</td>
<td>.2510</td>
<td>.2317</td>
<td>.23171</td>
<td>.51138</td>
<td>.23171</td>
</tr>
<tr>
<td>.33171</td>
<td>.78963</td>
<td>.66966</td>
<td>.86398</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.43171</td>
<td>.89693</td>
<td>.1318</td>
<td>.1356</td>
<td>.2210</td>
<td>.1575</td>
</tr>
<tr>
<td>.53171</td>
<td>.91757</td>
<td>.25751</td>
<td>.70905</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.63171</td>
<td>.77138</td>
<td>.35751</td>
<td>.8486</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.1316</td>
<td>.2344</td>
<td>.2448</td>
<td>.1152</td>
<td>.2740</td>
<td>.24483</td>
</tr>
<tr>
<td>.4483</td>
<td>.78917</td>
<td>.55751</td>
<td>.9101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.4483</td>
<td>.89846</td>
<td>.65751</td>
<td>.76614</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.5483</td>
<td>.89881</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.6483</td>
<td>.77086</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 2. Iterations Trial Values, large k.

<table>
<thead>
<tr>
<th>k</th>
<th>Weights</th>
<th>q</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>[.1316 .1179 .1409 .0719 .1302 .1279 .0661 .0946]</td>
<td>.13016</td>
<td>.66513</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.23016</td>
<td>.76384</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.33016</td>
<td>.87776</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.43016</td>
<td>.92834</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.53016</td>
<td>.94421</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.63016</td>
<td>.91617</td>
</tr>
<tr>
<td></td>
<td>[.0990 .1148 .1272 .0899 .1052 .0830 .1058 .1520 .1230]</td>
<td>.10521</td>
<td>.77953</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.20521</td>
<td>.55464</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.30521</td>
<td>.80359</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.40521</td>
<td>.94772</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.50521</td>
<td>.95709</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.60521</td>
<td>.89696</td>
</tr>
<tr>
<td></td>
<td>[.0958 .1498 .1137 .0841 .1184 .1140 .1450 .0904 .0887]</td>
<td>.11838</td>
<td>.66512</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.21838</td>
<td>.52621</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.31838</td>
<td>.78332</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.41838</td>
<td>.93586</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.51838</td>
<td>.95458</td>
</tr>
<tr>
<td>11</td>
<td>[.1140 .1003 .1039 .0985 .0804 .0969 .650 .1106 .0770 .0798 .0737]</td>
<td>.0969</td>
<td>.79882</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.1969</td>
<td>.78141</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.2969</td>
<td>.90791</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.3969</td>
<td>.95628</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.4969</td>
<td>.96787</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.5969</td>
<td>.95288</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.6969</td>
<td>.90373</td>
</tr>
<tr>
<td></td>
<td>[.1179 .0848 .1005 .0881 .0703 .1006 .1272 .0828 .0565 .0820 .0803]</td>
<td>.10058</td>
<td>.70145</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.20058</td>
<td>.53684</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.30058</td>
<td>.77768</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.40058</td>
<td>.95465</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.50058</td>
<td>.96673</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.60058</td>
<td>.95435</td>
</tr>
</tbody>
</table>