Check for updates

# Estimating Propensity Parameters Using Google PageRank and Genetic Algorithms

David Murrugarra *, Jacob Miller and Alex N. Mueller

*Department of Mathematics, University of Kentucky, Lexington, KY, USA*

Stochastic Boolean networks, or more generally, stochastic discrete networks, are an important class of computational models for molecular interaction networks. The stochasticity stems from the updating schedule. Standard updating schedules include the synchronous update, where all the nodes are updated at the same time, and the asynchronous update where a random node is updated at each time step. The former produces a deterministic dynamics while the latter a stochastic dynamics. A more general stochastic setting considers propensity parameters for updating each node. Stochastic Discrete Dynamical Systems (SDDS) are a modeling framework that considers two propensity parameters for updating each node and uses one when the update has a positive impact on the variable, that is, when the update causes the variable to increase its value, and uses the other when the update has a negative impact, that is, when the update causes it to decrease its value. This framework offers additional features for simulations but also adds a complexity in parameter estimation of the propensities. This paper presents a method for estimating the propensity parameters for SDDS. The method is based on adding noise to the system using the Google PageRank approach to make the system ergodic and thus guaranteeing the existence of a stationary distribution. Then with the use of a genetic algorithm, the propensity parameters are estimated. Approximation techniques that make the search algorithms efficient are also presented and Matlab/Octave code to test the algorithms are available at http://www.ms.uky.edu/~dmu228/GeneticAlg/Code.html.

Keywords: Boolean networks, stochastic systems, propensity parameters, Markov chains, Google PageRank, genetic algorithms, stationary distribution

## 1. INTRODUCTION

Mathematical modeling has been widely applied to the study of biological systems with the goal of understanding the important properties of the system and to derive useful predictions about the system. The type of systems of interest ranges from the molecular to ecological systems. At the cellular level, gene regulatory networks (GRN) have been extensively studied to understand the key mechanisms that are relevant for cell function. GRNs represent the intricate relationships among genes, proteins, and other substances that are responsible for the expression levels of mRNA and proteins. The amount of these gene products and their temporal patterns characterize specific cell states or phenotypes (Murrugarra and Dimitrova, 2015).

Gene expression is inherently stochastic with randomness in transcription and translation. This stochasticity is usually referred to as noise and it is one of the main drivers of variability (Raj and van Oudenaarden, 2008). Variability has an important role in cellular functions, and it can be beneficial as well as harmful (Kaern et al., 2005; Eldar and Elowitz, 2010). Modeling stochasticity is an important problem in systems biology. Different modeling approaches can be found in the literature. Mathematical models can be broadly divided into two classes: continuous, such as systems of differential equations and discrete, such as Boolean networks and their generalizations. This paper will focus on discrete stochastic methods. The Gillespie algorithm (Gillespie, 1977, 2007) considers discrete states but continuous time. In this work, we will focus on models where the space as well as the time are discrete variables. For instance, Boolean networks (BNs) are a class of computational models in which genes can only be in one of two states: ON or OFF. BNs and, in general, multistate models, which allow genes to take on more than two states, have been effectively used to model biological systems such as the *p53-mdm2* system (Abou-Jaoudé et al., 2009; Choi et al., 2012; Murrugarra et al., 2012), the *lac* operon (Veliz-Cuba and Stigler, 2011), the yeast cell cycle network (Li et al., 2004), the Th regulatory network (Mendoza, 2006), A. *thaliana* (Balleza et al., 2008), and many other systems (Albert and Othmer, 2003; Davidich and Bornholdt, 2008; Helikar et al., 2008, 2013; Zhang et al., 2008; Saadatpour et al., 2011).

Stochasticity in Boolean networks has been studied in different ways. The earliest approach to introduce stochasticity into BNs was the asynchronous update, where a random node is updated at each time step (Thomas and D'Ari, 1990). Another approach considers update sequences that can change from step to step (Mortveit and Reidys, 2007; Saadatpour et al., 2010). More sophisticated approaches include Probabilistic Boolean Networks (PBNs) (Shmulevich et al., 2002) and their variants (Layek et al., 2009; Liang and Han, 2012). PBNs consider stochasticity at the function level where each node can use multiple functions with a switching probability from step to step. SDDS (Murrugarra et al., 2012) is a simulation framework similar to PBNs but the key difference is how the transition probabilities are calculated. SDDS considers two propensity parameters for updating each node. These parameters resemble the propensity probabilities in the Gillespie algorithm (Gillespie, 1977, 2007). This extension gives a more flexible simulation setup as a generative model but adds the complexity of parameter estimation of the propensity parameters. This paper provides a method for computing the propensity parameters for SDDS.

For completeness, in the following subsection, we will define the stochastic framework to be used in remainder of the paper.

## Stochastic Framework

In this paper we will focus on the stochastic framework introduced in Murrugarra et al. (2012) referred to as Stochastic Discrete Dynamical Systems (SDDS). This framework is a natural extension of Boolean networks and is an appropriate setup to model the effect of intrinsic noise on network dynamics. Consider the discrete variables $x_1, \ldots, x_n$ that can take values in finite sets $S_1, \ldots, S_n$, respectively. Let $S = S_1 \times \cdots \times S_n$ be

the Cartesian product. A *SDDS* in the variables $x_1, \ldots, x_n$ is a collection of $n$ triplets

$$F = \{f_i, p_i^\uparrow, p_i^\downarrow\}_{i=1}^n$$

where

- $f_i : S \to S_i$ is the update function for $x_i$, for all $i = 1, \ldots, n$.
- $p_i^\uparrow$ is the activation propensity.
- $p_i^\downarrow$ is the degradation propensity.
- $p_i^\uparrow, p_i^\downarrow \in [0, 1]$. These are the parameters of interest in this paper.

The stochasticity originates from the propensity parameters $p_k^\uparrow$ and $p_k^\downarrow$, which should be interpreted as follows: If there would be an activation of $x_k$ at the next time step, i.e., if $s_1, s_2 \in S_k$ with $s_1 < s_2$ and $x_k(t) = s_1$, and $f_k(x_1(t), \ldots, x_n(t)) = s_2$, then $x_k(t+1) = s_2$ with probability $p_k^\uparrow$. The degradation probability $p_k^\downarrow$ is defined similarly. SDDS can be represented as a Markov chain by specifying its transition matrix in the following way. For each variable $x_i$, $i = 1, \ldots, n$, the probability of changing its value is given by

$$Prob(x_i \to f_i(x)) = \begin{cases} p_i^\uparrow, & \text{if } x_i < f_i(x), \\ p_i^\downarrow, & \text{if } x_i > f_i(x), \\ 1, & \text{if } x_i = f_i(x), \end{cases}$$

and the probability of maintaining its current value is given by

$$Prob(x_i \to x_i) = \begin{cases} 1 - p_i^\uparrow, & \text{if } x_i < f_i(x), \\ 1 - p_i^\downarrow, & \text{if } x_i > f_i(x), \\ 1, & \text{if } x_i = f_i(x). \end{cases}$$

Let $x, y \in S$. The transition from $x$ to $y$ is given by

$$a_{xy} = \prod_{i=1}^n Prob(x_i \to y_i). \tag{1}$$

Notice that $Prob(x_i \to y_i) = 0$ for all $y_i \notin \{x_i, f_i(x)\}$.

Then the transition matrix is given by

$$A = (a_{xy})_{x,y \in S} \tag{2}$$

The dynamics of SDDS depends on the transition probabilities $a_{xy}$, which depend on the propensity values and the update functions. Online software to test examples is available at http://adam.plantsimlab.org/ (choose Discrete Dynamical Systems (SDDS) in the model type).

In Markov chain notation, the transition probability $a_{xy} = p(X_t = x | X_{t-1} = y)$ represents the probability of being in state $x$ at time t given that system was in state $y$ at time $t - 1$. If $\pi_t = p(X_t = x)$ represents the probability of being in state $x$ at time $t$, then we will assume that $\pi$ is a row vector containing the probabilities of being in state $x$ at time $t$ for all $x \in S$. If $\pi_0$ is the initial distribution at time $t = 0$, then at time $t = 1$,

$$\pi_1 = \sum_{x \in S} \pi_0(x) a_{xy}. \tag{3}$$

If we iterate (Equation 3) and if we get to the point where

$$\pi = \sum_{x \in S} \pi(x) a_{xy} \qquad (4)$$

then we will say that the Markov chain has reached a stationary distribution and that $\pi$ is the stationary distribution.

## 2. METHODS

In this section we describe a method for estimating the propensity parameters for SDDS. The approach is based on adding noise to the system using the Google PageRank (Brin and Page, 2012; Lay, 2012; Murphy, 2012) strategy to make the system ergodic and thus guaranteeing the existence of a stationary distribution and then with the use of a genetic algorithm the propensity parameters are estimated. To guarantee the existence of a stationary distribution we use a special case of the Perron-Frobenius Theorem.

Theorem 2.1 (Perron-Frobenius). *If **A** is a regular $m \times m$ transition matrix with $m \geq 2$, then*

- *For any initial probability vector $\pi_0$, $\lim_{n \to \infty} A^n \pi_0 = \pi$.*
- *The vector $\pi$ is the unique probability vector which is an eigenvector of **A** associated with the eigenvalue 1.*

A proof of Theorem 2.1 can be found in Chapter 10 of Lay (2012).

Theorem 2.1 ensures a unique stationary distribution $\pi$ provided that we have a regular transition matrix, that is, if some power $\mathbf{A}^k$ contains only strictly positive entries. However, the transition matrix **A** of SDDS given in Equation (2) might not be regular. In the following subsection, we use a similar approach to the Google's PageRank algorithm to add noise to the system to obtain a new transition matrix that is regular.

**TABLE 1 | PageRank scores for the states of the attractors of the system.**

| Propensities | Attractor | Score |
|---|---|---|
| In Equation 10 | 0001110000 | 0.3346 |
| (all fixed to 0.9) | 1111000101 | 0.0463 |
| In Equation 11 | 0001110000 | 0.0199 |
| (genetic algorithm) | 1111000101 | 0.5485 |

*The order of variables in each vector state is M, P, B, C, R, $R_m$, A, $A_m$, L, $L_m$.*
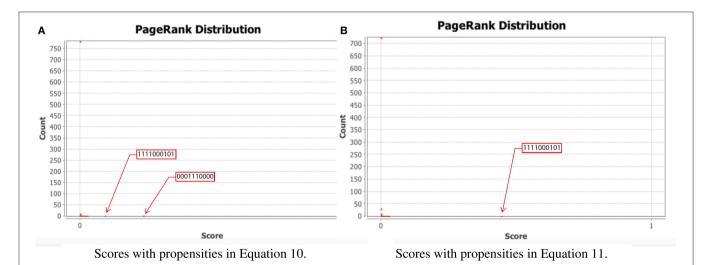
### 2.1. PageRank Algorithm

For simplicity, consider a SDDS, $F = \{f_i, p_i^{\uparrow}, p_i^{\downarrow}\}_{i=1}^n$ where $f_i : S \to S_i$, $S = \mathbb{K}^n$, and $|\mathbb{K}| = p$. Then its transition matrix **A** given in Equation (2) is a $p^n \times p^n$ matrix. To introduce noise into the system we consider the Google Matrix

$$\mathbf{G} = g\mathbf{A} + (1 - g)\mathbf{K}, \qquad (5)$$

where $g$ is a constant number in the interval $[0, 1]$ and **K** is a $p^n \times p^n$ matrix all of whose columns are the vector $(1/p^n, \ldots, 1/p^n)$. The matrix **G** in Equation (5) is a regular matrix and then we can use Theorem 2.1 to get a stationary distribution for **G**,

$$\pi = \pi_{\mathbf{G}} = (\pi_1, \ldots, \pi_{p^n}) \qquad (6)$$

This stationary distribution reflects the dynamics of the SDDS $F = \{f_i, p_i^{\uparrow}, p_i^{\downarrow}\}_{i=1}^n$. The importance of a state $x \in S$ can be measured by the size of the corresponding entry $\pi_x$ in the stationary distribution of Equation (6). For instance, for ranking the importance of the states in a Markov chain one can use the size of the corresponding entries in the stationary distribution. We will refer to this entry $\pi_x$ as the PageRank score of $x$.



Scores with propensities in Equation 10.

Scores with propensities in Equation 11.

**FIGURE 1 | PageRank scores before and after the genetic algorithm.** In each panel, the $x$-axis shows the PageRank scores while the $y$-axis shows the frequencies of states with the given scores in the $x$-axis (the exact scores for the states of interest are given in **Table 1**). Left panel shows the PageRank scores where all the propensities are equal to 0.9 while the right panel shows the PageRank scores where the propensity parameters where estimated using the genetic algorithm. **(A)** Scores with propensities in Equation (10). **(B)** Scores with propensities in Equation (11).

## 2.2. Genetic Algorithm

The entries of the stationary distribution $\pi$ in Equation (6) can also be interpreted as occupation times for each state. Thus, it gives the probability of being at a certain state. Now suppose that we start with a desired stationary distribution $\pi^* = (\pi_1^*, \ldots, \pi_{p^n}^*)$. We have developed a genetic algorithm that initializes a population of random propensity matrices and searches for a propensity matrix $prop^*$ such that its stationary distribution $\pi = (\pi_1, \ldots, \pi_{p^n})$ gets closer to the desired stationary distribution $\pi^*$. That is, we search for propensity matrices such that the distance between $\pi$ and $\pi^*$ is minimized,

$$\min_{p_i^\uparrow, p_i^\downarrow} d(\pi, \pi^*) \text{ or } \min_{p_i^\uparrow, p_i^\downarrow} |\pi(j) - \pi^*(j)| \qquad (7)$$

The pseudocode of this genetic algorithm is given in Algorithm 1 and it has been implemented in Octave/Matlab and our code can be downloaded from http://www.ms.uky.edu/~dmu228/GeneticAlg/Code.html.

## 2.3. Estimating the Stationary Distribution

The genetic algorithm, Algorithm 1, uses the exact stationary distribution through PageRank (see Equation 6) which is computationally expensive for larger models. Here we present an efficient algorithm for estimating the stationary distribution based on a random walk. The expensive part of Algorithm 1 is the calculation of the stationary distribution $\pi$ in Equation (6). We have implemented an algorithm for estimating the stationary distribution by doing a random walk using SDDS as a generative model; see Algorithm 2. The idea behind Algorithm 2 is to use SDDS for simulating from an initial state according to the transition probabilities given in Equation (5). That is, we initialize the simulation at an initial state $x \in S$ and then with probability $g$ we move to another state $y \in S$ according to $a_{xy}$ (see Equation 1) and with probability $1 - g$ we jump to a random node. We repeat this process for a given number of iterations. At the end of a maximum number of iterations, we count how often we visited each state and the normalized frequencies will be the approximated stationary distribution.

To make the genetic algorithm more efficient, we used the estimated stationary distribution described in the previous paragraph. Thus, within the fitness function of Algorithm 1, we use the estimated stationary distribution to assess the fitness of the generated propensity matrices. The pseudocode for this new algorithm is the same as Algorithm 1, the only change is in the fitness function. This version of the algorithm has also been implemented in Octave/Matlab and our code can be found in http://www.ms.uky.edu/~dmu228/GeneticAlg/Code.html.

## RESULTS

We test our methods using two published models that are appropriate for changing the stationary distribution under the choice of different propensity parameters. The first model is

---

**Algorithm 1** Genetic Algorithm with PageRank.

**Require:** Functions: $F = (f_1, \ldots, f_n)$, number of generations: *NumGen*, population size: *PopSize*, states of interest: *States*, and desired probabilities: $\pi^* = \pi^*(States)$.

**Ensure:** Propensity parameters: **prop**$^*$

1: **procedure** GENETICGOOGLE($F$, *NumGen*, *PopSize*, $\pi^*$)
2:     *PopPropensities* $\leftarrow$ initialize a population of propensity matrices.
3:     [*fitnesses*,min(*PopPropensities*)] = FITNESSGOOGLE($F$, *PopPropensities*, $\pi^*$)
4:     **for** $i = 1, \ldots, NumGen$ **do**
5:         *NewPropensities* $\leftarrow$ initialize new population of propensities.
6:         **for** $j = 1, \ldots, PopSize$ **do**
7:             **if** $rand < fitnesses(j)$ **then**
8:                 $parent1(j) = PopPropensities(j)$
9:             **else**
10:                 $parent2(j) = PopPropensities(j)$
11:             $children = $ CROSSOVER($parent1$, $parent2$, $mut$, $\sigma$)
12:             $NewPropensities(j) = children$
13:         [*fitnesses*,min(*NewPropensities*)] = FITNESSGOOGLE($F$, *NewPropensities*, $\pi^*$)
14:         $PopPropensities = NewPropensities$.
15:     **prop**$^*$ = min(*NewPropensities*).
16: **function** FITNESSGOOGLE($F$, *PopPropensities*, $\pi^*$)
17:     **for** $i = 1, \ldots, $ length(*PopPropensities*) **do**          ▷ For each propensity matrix.
18:     $\pi = \begin{cases} PageRank(F, PopPropensities(i)) \\ \quad \text{for exact distribution, see Equation 6,} \\ \text{ESTIMATESTADIST}(F, c, NumIter, g) \\ \quad \text{for estimated distribution, see Algorithm 2.} \end{cases}$
19:         $d = d(\pi, \pi^*)$  ▷ We used a weighted distance to give predominance to important states.
20:         $fitnesses(i) = e^{(-d^2/s)}$
21:     **return** ([*fitnesses*,min(*PopPropensities*)])          ▷ Keep propensity with minimum fitness.
22: **function** CROSSOVER($parent1$, $parent2$, $mut$, $\sigma$)
23:     $NewProp \leftarrow$ initialize new propensity matrix.
24:     $DivLine = $ random integer between 1 and length($parent1$).
25:     **for** $i = 1, \ldots, $ length($parent1$) **do**
26:         **if** $i < DivLine$ **then**
27:             $NewProp(i) = parent1(i)$
28:         **else**
29:             $NewProp(i) = parent2(i)$
30:         **if** $rand < mut$ **then**
31:             $NewProp(i) = NewProp(i) + normrand(0, \sigma)$   ▷ Introduce mutation.
32:     **return** ($NewProp$)

---

a Boolean network while the second is a multistate model. In both models bistability has been observed but the basin size of one of the attractors under the synchronous update is

much larger than the basin of the other attractor, and thus the stationary distribution will be more concentrated in one of the attractors. We will use our methods to change the stationary distribution in favor of the attractor with a smaller basin.

Example 2.2. **Lac-operon network.** The lac-operon in *E. coli* (Jacob and Monod, 1961) is one of the best studied

gene regulatory networks. This system is responsible for the metabolism of lactose in the absence of glucose. This system exhibits bistability in the sense that the operon can be either ON or OFF, depending on the presence of the preferred energy source: glucose. A Boolean network for this system has been developed in Veliz-Cuba and Stigler (2011). This model considers the following 10 components
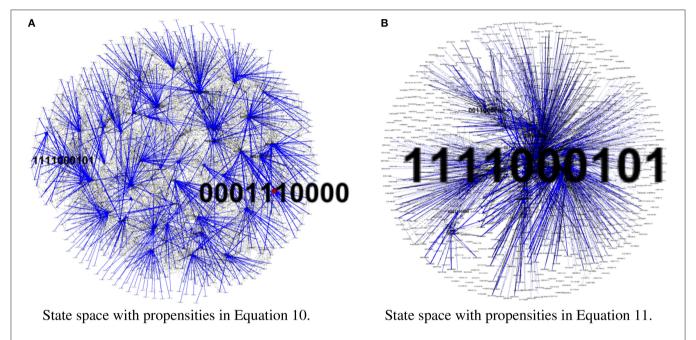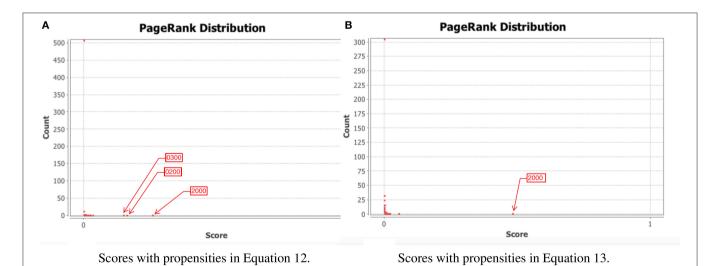


State space with propensities in Equation 10.

State space with propensities in Equation 11.

**FIGURE 2 | State space comparison before and after the genetic algorithm.** Left panel shows the state space where all the propensities are equal to 0.9 while the right panel shows the state space with the estimated propensity parameters using the genetic algorithm. The edges in blue represent the most likely trajectory. The size of the labels of the nodes were scaled according to their PageRank score. **(A)** State space with propensities in Equation (10). **(B)** State space with propensities in Equation (11).



Scores with propensities in Equation 12.

Scores with propensities in Equation 13.

**FIGURE 3 | PageRank scores before and after the genetic algorithm.** In each panel, the *x*-axis shows the PageRank scores while the *y*-axis shows the frequencies of states with the given scores in the *x*-axis (the exact scores for the states of interest are given in **Table 2**). Left panel shows the PageRank scores where all the propensities were equal to 0.9 while the right panel shows the scores where the propensity parameters where estimated using the genetic algorithm. The score for the state 2000 is 0.6040. **(A)** Scores with propensities in Equation (12). **(B)** Scores with propensities in Equation (13).

**Algorithm 2** Estimate Stationary Distribution.

**Require:** Functions: $F = (f_1, \ldots, f_n)$, propensities: $c$, number of iterations: *NumIter*, noise: $g$.

**Ensure:** Estimated stationary distribution $\pi$

$\quad \pi = \text{ESTIMATESTADIST}(F, c, NumIter, g)$

2: **return** $\pi$

$\quad$ **function** ESTIMATESTADIST($F, c, NumIter, g$)

4: $\quad\quad$ *distribution* ← initialize frequency vector.

$\quad\quad$ $s$ ← initialize random initial state.

6: $\quad\quad$ **for** i = 1,..., *NumIter* **do**

$\quad\quad\quad$ **if** *rand* < $g$ **then**

8: $\quad\quad\quad\quad$ $y$ = random state between 1 and $p^n$.

$\quad\quad\quad$ **else**

10: $\quad\quad\quad\quad$ $y = SDDS$.nextstate($s, c$)

$\quad\quad\quad$ *distribution*($y$) = +1 increase state frequency.

12: $\quad\quad$ *sum* = total frequencies.

$\quad\quad$ $\pi = distribution/sum$

14: $\quad$ **return** $\pi$

---

$x_1 = M$: *lac* mRNA,

$x_2 = P$: *lac permease*,

$x_3 = B$: *lacβ-galactosidase*,

$x_4 = C$: CAP,

$x_5 = R$: repressor,

$x_6 = Rm$: repressor at medium concentration,

$x_7 = A$: allolactose,

$x_8 = Am$: allolactose at medium concentration,

$x_9 = L$: lactose,

$x_{10} = Lm$: lactose at medium concentration,

$$\tag{8}$$

and the Boolean rules are given by

$$
\begin{aligned}
f_1 &= x_4 \wedge \overline{x_5} \wedge \overline{x_6}, \\
f_2 &= x_1, \\
f_3 &= x_1, \\
f_4 &= \overline{G_e}, \\
f_5 &= \overline{x_7} \wedge \overline{x_8}, \\
f_6 &= (\overline{x_7} \wedge \overline{x_8}) \vee x_5, \\
f_7 &= x_9 \wedge x_3, \\
f_8 &= x_9 \vee x_{10}, \\
f_9 &= x_2 \wedge L_e \wedge \overline{G_e}, \\
f_{10} &= ((L_{em} \wedge P) \vee L_e) \wedge \overline{G_e}.
\end{aligned}
\tag{9}
$$

where $G_e$, $L_{em}$, and $L_e$ are parameters. $G_e$ and $L_e$ indicate the concentration of extracellular glucose and lactose, respectively. The parameter $L_{em}$ indicates the medium concentration of extracellular lactose. For medium extracellular lactose, that is when $L_{em} = 1$ and $L_e = 0$, this system has two fixed points: $s_1 = (0, 0, 0, 1, 1, 1, 0, 0, 0, 0)$ and $s_2 = (1, 1, 1, 1, 0, 0, 0, 1, 0, 1)$ that represent the state of the operon being OFF and ON, respectively.

To test our method we calculated the stationary distribution of the system using Equation (6) with $g = 0.9$ in Equation (5). First we used the propensity values given in Equation (10) where all propensities are fixed to 0.9. This choice of parameters approximates the synchronous dynamics in the sense that each function has a 90% change of being used during the simulations and 10% chance of maintaining its current value. Under this

**TABLE 2 | PageRank scores for the states of the attractors of the system.**

| Propensities | Attractor | Score |
|---|---|---|
| In Equation 12 | 2000 | 0.2772 |
| (all fixed to 0.9) | 0200 | 0.2185 |
| | 0300 | 0.2108 |
| In Equation 13 | 2000 | 0.6040 |
| (genetic algorithm) | 0200 | 0.0716 |
| | 0300 | 0.00016 |

*The order of variables in each vector state is CI, CRO, CII, N.*

selection of parameters, the fixed point $s_1$ has a PageRank score of 0.3346 while the other fixed point $s_2$ has a score of 0.0463, see **Figure 1A** and **Table 1**. Then we have applied our genetic algorithm to search for parameters that can increase the PageRank score of $s_2$ and decrease the score of $s_1$. After doing so, we found the propensity parameters given in Equation (11). With this new set of parameters, the fixed point $s_1$ has a score of 0.0199 while $s_2$ has a score of 0.5485, see **Figure 1B** and **Table 1**. To appreciate the impact of the change in propensity parameters, we have plotted the state space of the system with both propensity matrices in **Figure 2**. The edges in blue in **Figure 2** represent the most likely trajectory. Notice that in **Figure 2B** the trajectories are leading toward $s_2$ and the size of the labels of the nodes were scaled according to their PageRank score, see **Table 1**.

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ | $x_{10}$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $p_i^\uparrow$ | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | (10) |
| $p_i^\downarrow$ | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | |

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ | $x_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $p_i^\uparrow$ | 0.81 | 1.00 | 0.97 | 0.62 | 0.11 | 0.63 | 0.22 | 0.82 | 0.48 | 0.60 |
| $p_i^\downarrow$ | 0.17 | 0.59 | 0.03 | 0.98 | 0.39 | 1.00 | 0.33 | 0.07 | 0.52 | 0.06 |

$$\tag{11}$$

**Example 2.3. Phage lambda infection.** The outcome of phage lambda infection is another system that has been widely studied over the last decades (Ptashne and Switch, 1992; Thieffry and Thomas, 1995; St-Pierre and Endy, 2008; Zeng et al., 2010; Joh and Weitz, 2011). One of the earliest models that has been developed for this system is the logical model by Thieffry and Thomas (1995). The regulatory genes considered in this model are: *CI, CRO, CII,* and *N*. Experimental reports (Reichardt and Kaiser, 1971; Kourilsky, 1973; Thieffry and Thomas, 1995; St-Pierre and Endy, 2008) have shown that, if the gene *CI* is fully expressed, all other genes are OFF. In the absence of *CRO* protein, *CI* is fully expressed (even in the absence of *N* and *CII*). *CI* is fully repressed provided that CRO is active and CII is absent.

This network is a bistable switch between lysis and lysogeny. Lysis is the state where the phage will be replicated, killing the host. Otherwise, the network will transition to a state called lysogeny where the phage will incorporate its DNA into the

bacterium and become dormant. These cell fate differences have been attributed to the spontaneous changes in the timing of individual biochemical reaction events (Thieffry and Thomas, 1995; McAdams and Arkin, 1997).

In the model of Thieffry and Thomas (1995), the first variable, *CI*, has three levels {0, 1, 2}, the second variable, *CRO*, has four levels {0, 1, 2, 3}, and the third and fourth variables, *CII* and *N*, are Boolean. Since the nodes of this model have different number of states, in order to apply our methods, we have extended the model so that all nodes have the same number of states. We have used the method given in Veliz-Cuba et al. (2010) to extend the number of states such that all nodes have 5 states (the method for extending requires a prime number for number of states so we have chosen 5 states). The method for extending the number of states preserves the original attractors. The update rules for this model are available with our code that is freely available. The extended model has a steady state, 2000, and a 2-cycle involving 0200 and 0300. The steady state 2000 represents lysogeny where *CI* is fully expressed while the other genes are OFF. The cycle between 0200 and 0300 represents lysis where *CRO* is active and other genes are repressed.

To test our method we calculated the stationary distribution of the system using Equation (6) with $g = 0.9$ in Equation (5). First we used the propensity values given in Equation (12) where all propensities are fixed to 0.9. This choice of parameters approximates the synchronous dynamics in the sense that each function has a 90% change of being used during the simulations and 10% chance of maintaining its current value. Under this

selection of parameters, the fixed point 2000 has a PageRank score of 0.2772 while the states of the cycle 0200 and 0300 have scores of 0.2185 and 0.2108, respectively. Notice that this cycle attractor will have an overall score of 0.4293, see **Figure 3A** and **Table 2**. Then we have applied our genetic algorithm to search for parameters that can increase the PageRank score of the fixed point 2000 and found the propensity parameters given in Equation (13). With this new set of parameters, the fixed point 2000 has a score of 0.6040 while the states of the cycle 0200 and 0300 have scores of 0.0716 and 0.00016, respectively, see **Figure 3B** and **Table 2**. To appreciate the impact of the change in propensity parameters, we have plotted the state space of the system with both propensity matrices in **Figure 4**. The edges in blue in **Figure 4** represent the most likely trajectory. Notice that in **Figure 4B** the trajectories are leading toward 2000 and the size of the labels of the nodes were scaled according to their PageRank score, see **Table 2**.

|  | *CI* | *CRO* | *CII* | *N* |
|---|---|---|---|---|
| $p_i^\uparrow$ | 0.9 | 0.9 | 0.9 | 0.9 |
| $p_i^\downarrow$ | 0.9 | 0.9 | 0.9 | 0.9 |

$$(12)$$

|  | *CI* | *CRO* | *CII* | *N* |
|---|---|---|---|---|
| $p_i^\uparrow$ | 1.0000 | 0 | 0.4277 | 0.7968 |
| $p_i^\downarrow$ | 0.3962 | 1.0000 | 0.6063 | 0.6946 |

$$(13)$$



State space with propensities in Equation 12.          State space with propensities in Equation 13.
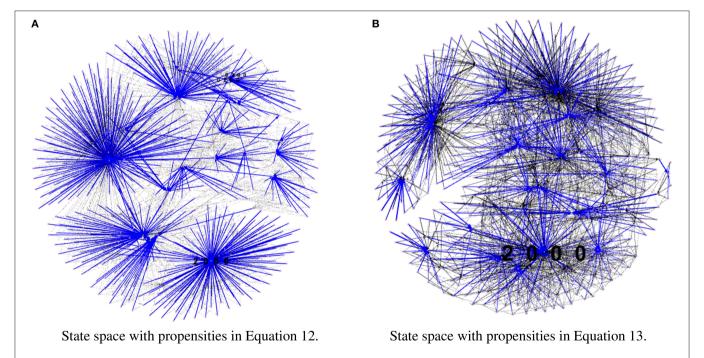
**FIGURE 4 | State space comparison before and after the genetic algorithm.** Left panel shows the state space where all the propensities are equal to 0.9 while the right panel shows the state space with the estimated propensity parameters using the genetic algorithm. The edges in blue represent the most likely trajectory. The size of the labels of the node were scaled according to their PageRank score. **(A)** State space with propensities in Equation (12). **(B)** State space with propensities in Equation (13).

# 3. DISCUSSION

Parameter estimation for stochastic models of biological networks is in general a very hard problem. This paper focuses on a class of stochastic discrete models, which is an extension of Boolean networks. The methods presented here use a well stablished approach for introducing noise into a system in a way that ergodicity and thus the existence of a unique stationary distribution is guaranteed. Then a genetic algorithm for calculating a set of parameters able to approximate a desired stationary distribution was developed. Also, techniques for approximating the stationary distribution at each iteration of the genetic algorithm that make the search process more efficient was applied.

One shortcoming of the method is that it is a stochastic method. That is, each time we run the algorithm we get a different result. An exhaustive investigation about the variance of the results is still missing. For the examples that we presented in the results section, the output of the algorithm might vary in about 20% of the reported propensities.

In parameter estimation, sometimes, it is useful to identify a smaller set of key parameters to estimate. This problem is out of the scope of the paper. However, for Boolean network models, one way to address this problem could be by using the different network reduction algorithms, for instance see Veliz-Cuba et al. (2014) and Saadatpour et al. (2010), to identify a smaller "core" network that preserves the important features of the dynamics of the original network. And then one could apply the parameter estimation techniques that are described in this paper. This type of approach could be especially useful if dealing with very large networks where running the genetic algorithm is computationally expensive.

# 4. CONCLUSIONS

In this paper we present an efficient method for estimating the parameters of a stochastic framework. The modeling framework is an extension of Boolean networks that uses propensity parameters for activation and inhibition. Parameter estimation techniques are needed whenever one needs to tune the propensity parameters of the stochastic system to reproduce a desired stationary distribution. For instance, if dealing with a bistable system and if it is desired to have the stationary distribution that have the PageRank score concentrated in one of the attractors of the system, then one needs to estimate the propensity parameters that represent such a desired distribution. Parameter estimation methods for this purpose were not available. In this paper we present a method for estimating propensity parameters given a desired stationary distribution for the system. We tested the method in one Boolean network with 10 nodes (where the size of the state space is $2^{10} = 1024$) and a multistate network with 4 nodes where each node has 5 states (where the size of the state space is $4^5 = 625$). For each system, we were able to redirect the system toward the attractor with the smaller PageRank score. The method is efficient and for the examples we have shown it can be run in few seconds in a laptop computer. Our code is available at http://www.ms.uky.edu/~dmu228/GeneticAlg/Code.html.

# AUTHOR CONTRIBUTIONS

DM, JM, and AM designed the project. JM developed the genetic algorithm and implemented the Matlab code. AM run simulations for the results section. DM wrote the paper. All authors approved the final version of the manuscript.

# FUNDING

# ACKNOWLEDGMENTS

# REFERENCES

Abou-Jaoudé, W., Ouattara, D. A., and Kaufman, M. (2009). From structure to dynamics: frequency tuning in the p53-mdm2 network I. Logical approach. *J. Theor. Biol.* 258, 561–577. doi: 10.1016/j.jtbi.2009.02.005

Albert, R., and Othmer, H. G. (2003). The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in drosophila melanogaster. *J. Theor. Biol.* 223, 1–18. doi: 10.1016/S0022-5193(03)00035-3

Balleza, E., Alvarez-Buylla, E. R., Chaos, A., Kauffman, S., Shmulevich, I., and Aldana, M. (2008). Critical dynamics in genetic regulatory networks: examples from four kingdoms. *PLoS ONE* 3:e2456. doi: 10.1371/journal.pone.0002456

Brin, S., and Page, L. (2012). Reprint of: the anatomy of a large-scale hypertextual web search engine. *Comput. Netw.* 56, 3825–3833. doi: 10.1016/j.comnet.2012.10.007

Choi, M., Shi, J., Jung, S. H., Chen, X., and Cho, K.-H. (2012). Attractor landscape analysis reveals feedback loops in the p53 network that control the cellular response to dna damage. *Sci. Signal.* 5:ra83. doi: 10.1126/scisignal.2003363

Davidich, M. I., and Bornholdt, S. (2008). Boolean network model predicts cell cycle sequence of fission yeast. *PLoS ONE* 3:e1672. doi: 10.1371/journal.pone.0001672

Eldar, A., and Elowitz, M. B. (2010). Functional roles for noise in genetic circuits. *Nature* 467, 167–173. doi: 10.1038/nature09326

Gillespie, D. T. (1977). Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.* 81, 2340–2361. doi: 10.1021/j100540a008

Gillespie, D. T. (2007). Stochastic simulation of chemical kinetics. *Ann. Rev. Phys. Chem.* 58, 35–55. doi: 10.1146/annurev.physchem.58.032806.104637

Helikar, T., Kochi, N., Kowal, B., Dimri, M., Naramura, M., Raja, S. M., et al. (2013). A comprehensive, multi-scale dynamical model of erbb receptor signal transduction in human mammary epithelial cells. *PLoS ONE* 8:e61757. doi: 10.1371/journal.pone.0061757

Helikar, T., Konvalina, J., Heidel, J., and Rogers, J. A. (2008). Emergent decision-making in biological signal transduction networks. *Proc. Natl. Acad. Sci. U.S.A.* 105, 1913–1918. doi: 10.1073/pnas.0705088105

Jacob, F., and Monod, J. (1961). Genetic regulatory mechanisms in the synthesis of proteins. *J. Mol. Biol.* 3, 318–356. doi: 10.1016/S0022-2836(61)80072-7

Joh, R. I., and Weitz, J. S. (2011). To lyse or not to lyse: transient-mediated stochastic fate determination in cells infected by bacteriophages. *PLoS Comput. Biol.* 7:e1002006. doi: 10.1371/journal.pcbi.1002006

Kaern, M., Elston, T. C., Blake, W. J., and Collins, J. J. (2005). Stochasticity in gene expression: from theories to phenotypes. *Nat. Rev. Genet.* 6, 451–464. doi: 10.1038/nrg1615

Kourilsky, P. (1973). Lysogenization by bacteriophage lambda. I. Multiple infection and the lysogenic response. *Mol. Gen. Genet.* 122, 183–195. doi: 10.1007/BF00435190

Lay, D. C. (2012). *Linear Algebra And Its Applications, 4th Edn.* New York, NY: PEARSON.

Layek, R., Datta, A., Pal, R., and Dougherty, E. R. (2009). Adaptive intervention in probabilistic boolean networks. *Bioinformatics* 25, 2042–2048. doi: 10.1093/bioinformatics/btp349

Li, F., Long, T., Lu, Y., Ouyang, Q., and Tang, C. (2004). The yeast cell-cycle network is robustly designed. *Proc. Natl. Acad. Sci. U.S.A.* 101, 4781–4786. doi: 10.1073/pnas.0305937101

Liang, J., and Han, J. (2012). Stochastic boolean networks: an efficient approach to modeling gene regulatory networks. *BMC Syst. Biol.* 6:113. doi: 10.1186/1752-0509-6-113

McAdams, H. H., and Arkin, A. (1997). Stochastic mechanism in gene expression. *Proc. Natl. Acad. Sci. U.S.A.* 94, 814–819. doi: 10.1073/pnas.94.3.814

Mendoza, L. (2006). A network model for the control of the differentiation process in th cells. *Biosystems* 84, 101–114. doi: 10.1016/j.biosystems.2005.10.004

Mortveit, H. S., and Reidys, C. M. (2007). *An Introduction to Sequential Dynamical Systems.* Secaucus, NJ: Springer-Verlag New York, Inc.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective.* Cambridge, MA: The MIT Press.

Murrugarra, D., and Dimitrova, E. S. (2015). Molecular network control through boolean canalization. *EURASIP J. Bioinform. Syst. Biol.* 2015:9. doi: 10.1186/s13637-015-0029-2

Murrugarra, D., Veliz-Cuba, A., Aguilar, B., Arat, S., and Laubenbacher, R. (2012). Modeling stochasticity and variability in gene regulatory networks. *EURASIP J. Bioinform. Syst. Biol.* 2012:5. doi: 10.1186/1687-4153-2012-5

Ptashne, M., and Switch, A. G. (1992). *Phage Lambda and Higher Organisms.* Cambridge, MA: Cell & Blackwell Scientific.

Raj, A., and van Oudenaarden, A. (2008). Nature, nurture, or chance: stochastic gene expression and its consequences. *Cell* 135, 216–226. doi: 10.1016/j.cell.2008.09.050

Reichardt, L., and Kaiser, A. (1971). Control of λ repressor synthesis. *Proc. Natl. Acad. Sci. U.S.A.* 68, 2185–2189. doi: 10.1073/pnas.68.9.2185

Saadatpour, A., Albert, I., and Albert, R. (2010). Attractor analysis of asynchronous boolean models of signal transduction networks. *J. Theor. Biol.* 266, 641–656. doi: 10.1016/j.jtbi.2010.07.022

Saadatpour, A., Wang, R.-S., Liao, A., Liu, X., Loughran, T. P., Albert, I., et al. (2011). Dynamical and structural analysis of a t cell survival network identifies novel candidate therapeutic targets for large granular lymphocyte leukemia. *PLoS Comput. Biol.* 7:e1002267. doi: 10.1371/journal.pcbi.1002267

Shmulevich, I., Dougherty, E. R., Kim, S., and Zhang, W. (2002). Probabilistic boolean networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics* 18, 261–274. doi: 10.1093/bioinformatics/18.2.261

St-Pierre, F., and Endy, D. (2008). Determination of cell fate selection during phage lambda infection. *Proc. Natl. Acad. Sci. U.S.A.* 105, 20705–20710. doi: 10.1073/pnas.0808831105

Thieffry, D., and Thomas, R. (1995). Dynamical behaviour of biological regulatory networks–II. Immunity control in bacteriophage lambda. *Bull. Math. Biol.* 57, 277–297.

Thomas, R., and D'Ari, R. (1990). *Biological Feedback.* Boca Raton, FL: CRC Press.

Veliz-Cuba, A., Aguilar, B., Hinkelmann, F., and Laubenbacher, R. (2014). Steady state analysis of boolean molecular network models via model reduction and computational algebra. *BMC Bioinformatics* 15:221. doi: 10.1186/1471-2105-15-221

Veliz-Cuba, A., Jarrah, A. S., and Laubenbacher, R. (2010). Polynomial algebra of discrete models in systems biology. *Bioinformatics* 26, 1637–1643. doi: 10.1093/bioinformatics/btq240

Veliz-Cuba, A., and Stigler, B. (2011). Boolean models can explain bistability in the *lac* operon. *J. Comput. Biol.* 18, 783–794. doi: 10.1089/cmb.2011.0031

Zeng, L., Skinner, S. O., Zong, C., Sippy, J., Feiss, M., and Golding, I. (2010). Decision making at a subcellular level determines the outcome of bacteriophage infection. *Cell* 141, 682–691. doi: 10.1016/j.cell.2010.03.034

Zhang, R., Shah, M. V., Yang, J., Nyland, S. B., Liu, X., Yun, J. K., et al. (2008). Network model of survival signaling in large granular lymphocyte leukemia. *Proc. Natl. Acad. Sci. U.S.A.* 105, 16308–16313. doi: 10.1073/pnas.0806447105