Journal of Biological Systems, Vol. 10, No. 4 (2002) 431–445
© World Scientific Publishing Company

CONTROL OF STATIONARY BEHAVIOR IN PROBABILISTIC BOOLEAN NETWORKS BY MEANS OF STRUCTURAL INTERVENTION

ILYA SHMULEVICH*, EDWARD R. DOUGHERTY† and WEI ZHANG*

*Cancer Genomics Laboratory, The University of Texas, M. D. Anderson Cancer Center, Houston, Texas, USA
†Department of Electrical Engineering, Texas A&M University, College Station, Texas, USA

Received 18 March 2002
Revised 30 May 2002

Probabilistic Boolean Networks (PBNs) were recently introduced as models of gene regulatory networks. The dynamical behavior of PBNs, which are probabilistic generalizations of Boolean networks, can be studied using Markov chain theory. In particular, the steady-state or long-run behavior of PBNs may reflect the phenotype or functional state of the cell. Approaches to alter the steady-state behavior in a specific prescribed manner, in cases of aberrant cellular states, such as tumorigenesis, would be highly beneficial. This paper develops a methodology for altering the steady-state probabilities of certain states or sets of states with minimal modifications to the underlying rule-based structure. This approach is framed as an optimization problem that we propose to solve using genetic algorithms, which are well suited for capturing the underlying structure of PBNs and are able to locate the optimal solution in a highly efficient manner. Several computer simulation experiments support the proposed methodology.

Keywords: Genetic network; Markov chain; genetic algorithm.

1. Introduction

It is becoming widely recognized that the study and modeling of genetic regulatory networks carries tremendous potential for gaining a deep understanding of biological processes and ergo, for developing effective therapeutic intervention in human diseases such as cancer. This inevitably entails using computational and formal methods in order to understand general principles governing the system under study and to make useful predictions about system behavior in the presence of known conditions.

Recently, a new class of models, called *Probabilistic Boolean Networks* (PBNs), was introduced as a model of gene regulatory networks [12]. This model class incorporates rule-based dependencies between genes, allows the systematic study of global network dynamics, is able to cope with uncertainty, and permits the quantification of the relative influence and sensitivity of genes in their interactions with other genes. The dynamics of these networks can be studied in the probabilistic context of Markov chains. They also exhibit an important connection with Bayesian Networks — another model class that has been recently used to model gene expression data. PBNs represent an interface between the absolute determinism of Boolean networks and the probabilistic nature of Bayesian networks, in that they incorporate rule-based uncertainty. This compromise is important because rule-based dependencies between genes are biologically meaningful, while mechanisms for handling uncertainty are conceptually and empirically necessary.

The dynamical properties of PBNs were further studied in [13], which considers the general question of the potential effect of individual genes on the global dynamical network behavior, both from the view of random gene perturbation as well as intervention in order to elicit desired network behavior. Namely, it develops a model for random gene perturbations and an explicit formula for the transition probabilities.

This result provides a building block for performing simulations and deriving other results concerning network dynamics. For example, the problem of (gene) intervention has been addressed via the development of several computational tools based on first-passage times in Markov chains. The consequence is a methodology for finding the best gene with which to intervene in order to most likely achieve desirable network behavior. In addition, the effect of gene perturbations on long-run network behavior has been assessed by providing a bound on the steady-state probabilities in terms of the probability of perturbation. This result demonstrates that states of the network that are more "easily reachable" from other states are more stable in the presence of gene perturbations. Such states are hypothesized to correspond to cellular functional states — a view that is in accord with existing findings [5, 6].

The type of intervention described in [13] — one that allows us to intervene with the *value* of a gene — can be useful for modulating the dynamics of the network, but it is not able to alter its underlying structure. Accordingly, the stationary distribution remains unchanged. However, a disbalance between certain sets of states, which is characteristic of neoplasia in view of gene regulatory networks, can be caused by mutations of the "wiring" of certain genes, thus permanently altering the state-transition structure and, consequently, the long-run behavior of the network [5].

Therefore, it is prudent to develop a methodology for altering the steady-state probabilities of certain states or sets of states with minimal modifications to the rule-based structure. The motivation is that these states may represent different phenotypes or cellular functional states, such as cell invasion and quiescence, and we would like to decrease the probability that the whole network will end up in an undesirable set of states and increase the probability that it will end up in a desirable set of states. The mechanism by which we accomplish this consists of altering some Boolean functions (predictors) in the PBN. An additional goal is to alter as few functions as possible. Such alterations to the rules of regulation may be possible by the introduction of a factor or drug that alters the extant behavior. Let us give an example.

We know that women can age much faster after menopause. In developed countries, estrogen is often taken by women after menopause to alter this trend. However, the dose of estrogen is important because an overdose may increase the probabilities of developing breast and ovarian cancers. Although the mechanism is not clear yet, it is conceivable that this phenomenon has its gene regulation basis. Estrogen binds its receptors, the complex gets transported into the nucleus to bind the enhancer element (a short stretch of regulatory DNA sequence) on the target genes, and functions as transcriptional factors affecting genes such as the preproenkephalin (PENK) gene [14]. Interestingly, there are several different estrogen receptors that compete with each other for binding estrogen as well as for coactivator, which is also required for efficient transcriptional regulation by estrogen [16]. It can be envisioned that estrogen binds one receptor better than another and that these complexes bind DNA and coactivator with opposite efficiency. That is, complex C1 binds DNA better than complex C2, but complex C2 binds the coactivator better than complex C1. Thus, under low estrogen conditions, when there is not much competition for DNA binding, there would be sufficient binding of C2 to DNA so as to turn on the downstream target gene. However, when estrogen is present at high concentration, both complexes exist at very high levels and complex C2, taking up most of the coactivator away from C1, would have little chance to bind to DNA. Consequently, the better DNA binding complex (C1) would not have the necessary coactivator to activate the target gene. If the target gene plays a role in tumor suppression, for instance, this could explain why high levels of estrogen have a tumorigenic effect. Thus, by changing the concentration of estrogen, one is able to alter the rule determining the value of a gene (e.g., PENK) in terms of the levels of estrogen receptor complexes C1 and C2.

For example, under a low estrogen condition, assuming Boolean values for all genes, PENK can be expressed as $\text{PENK} = \text{C1} \vee \text{C2}$, where the symbol $\vee$ means logical OR. That is, the presence of at least one complex (C1 or C2) would be sufficient to turn on PENK. However, under a high estrogen condition, in view of the above squelching effect, $\text{PENK} = \text{C1} \oplus \text{C2}$, where the symbol $\oplus$ means exclusive OR. That is, when either C1 or C2 are individually present, with no competition from the other, PENK is turned on, but when both C1 and C2 are present together, PENK becomes turned off. The ability to alter such rules would provide a means of at least partially controlling the steady-state behavior of the network.

In this paper, we develop formal methods and algorithms for addressing such problems. In Sec. 2, we briefly review the background material for Probabilistic Boolean Networks. Section 3 formalizes the problem described above in terms of an optimization problem. Finally, Sec. 4 describes a solution to this problem via genetic algorithms. We have performed several computer simulation experiments to assess the efficiency of the proposed methodology.

2. Definitions and Background Material

We will give the basic definitions and notations for PBNs and refer the reader to [12] for more details and examples. A PBN $G(V, F)$ is defined by a set of binary-valued nodes $V = \{x_1, \ldots, x_n\}$ and a list $F = (F_1, \ldots, F_n)$ of sets $F_i = \{f_1^{(i)}, \ldots, f_{l(i)}^{(i)}\}$ of Boolean functions. Each node $x_i \in \{0, 1\}$ represents the state (expression) of gene i, where $x_i = 1$ means that gene i is expressed and $x_i = 0$ means it is not expressed. The set F_i represents the possible rules of regulatory interactions for gene x_i. That is, each $f_j^{(i)} : \{0,1\}^n \to \{0,1\}$ is a possible Boolean function determining the value of gene x_i in terms of some other genes and $l(i)$ is the number of possible functions for gene x_i. We will also refer to the functions $f_j^{(i)}$ as *predictors*. Thus, any given gene transforms its inputs (regulatory factors that bind to it) into an output, which is the state or expression of the gene itself. All genes (nodes) are updated synchronously in accordance with the functions assigned to them and this process is then repeated. At any given time step, one of the predictors for gene x_i is selected randomly from the set F_i, according to a predefined probability distribution, discussed below.

A *realization* of the PBN at a given instant of time is determined by a vector of Boolean functions. If there are N possible realizations, then there are N vector functions, $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_N$ of the form $\mathbf{f}_k = (f_{k_1}^{(1)}, f_{k_2}^{(2)}, \ldots, f_{k_n}^{(n)})$, for $k = 1, 2, \ldots, N$, $1 \le k_i \le l(i)$ and where $f_{k_i}^{(i)} \in F_i$ $(i = 1, \ldots, n)$. In other words, the vector function (also called multiple-output function) $\mathbf{f}_k : \{0,1\}^n \to \{0,1\}^n$ acts as a transition function (mapping) representing a possible realization of the entire PBN. Thus, given the values of all genes $(x_1, \ldots, x_n)$, $\mathbf{f}_k(x_1, \ldots, x_n) = (x'_1, \ldots, x'_n)$ gives us the state of the genes after one step of the network given by the realization $\mathbf{f}_k$. If the predictor for each gene is chosen independently of other predictors, then $N = \prod_{i=1}^{n} l(i)$. It should be noted that each predictor function $f_j^{(i)}$ typically has many fictitious variables. That is, although the domain of each predictor is $\{0,1\}^n$, there are only a few input genes that actually regulate gene x_i at any given time, implying that each predictor is a simple one. The biological and practical justifications for probabilistically choosing one of several simple predictors for each gene are discussed in [12].

Let $\mathbf{f} = (f^{(1)}, \ldots, f^{(n)})$ be a random vector taking values in $F_1 \times \cdots \times F_n$. That is, $\mathbf{f}$ can take on all possible realizations of the PBN. Then, the probability that predictor $f_j^{(i)}$ is used to predict gene i $(1 \le j \le l(i))$ is equal to

$$c_j^{(i)} = \Pr\{f^{(i)} = f_j^{(i)}\} = \sum_{k : f_{k_i}^{(i)} = f_j^{(i)}} \Pr\{\mathbf{f} = \mathbf{f}_k\}\,. \tag{1}$$

An approach for obtaining the probabilities $c_j^{(i)}$ from gene expression data, using the coefficient of determination [1, 10, 11] and in turn for using these probabilities to compute the probability that a particular network realization is selected, is discussed in [12]. It was also shown that the dynamics of PBNs can be modeled by Markov chains, consisting of 2^n states.

Further, an extension to the PBN model permitting so-called gene *perturbations* was proposed in [13], allowing any out of n possible genes to get perturbed with probability p, independently of other genes. In the Boolean setting, this is represented by a flip of value from 1 to 0 or vice versa and directly corresponds to the bit-flipping mutation operator in *NK* Landscapes [8, 9] as well as in genetic algorithms and evolutionary computing [2]. This situation is modeled as follows. Suppose that at every step of the network, we have a realization of a so-called random *perturbation vector* $\gamma \in \{0,1\}^n$. If the ith component of γ is equal to 1, then the ith gene is flipped, otherwise it is not. In general, γ need not be independent and identically distributed (i.i.d.), but we will assume this for now on for simplicity. Thus, we will suppose that $\Pr\{\gamma_i = 1\} = \mathrm{E}[\gamma_i] = p$ for all $i = 1, \ldots, n$. Let $x = (x_1, \ldots, x_n)$ be the state of the network (i.e., values of all the genes) at some given time. Then, the next state x' is given by

$$x' = \begin{cases} x \oplus \gamma\,, & \text{with probability } 1-(1-p)^n \\ \mathbf{f}_k(x_1, \ldots, x_n)\,, & \text{with probability } (1-p)^n \end{cases}\,, \tag{2}$$

where $\oplus$ is component-wise addition modulo 2 and $\mathbf{f}_k(x_1, \ldots, x_n)$, $k = 1, 2, \ldots, N$, is the transition function representing a possible realization of the entire PBN. The state transition matrix of the Markov chain can be expressed in terms of the Boolean functions and the probabilities of different network realizations, which are given in terms of $c_j^{(i)}$ in (1) [13]. The availability of the state transition matrix, in turn, allows us to compute the stationary probabilities $\pi(x)$ of all states $x \in \{0,1\}^n$. For large networks, computing the stationary probabilities via the transition matrix is impractical and Monte Carlo techniques can be used. It is relatively straightforward to show that for $p > 0$, the Markov chain corresponding to the PBN is ergodic [13]. Consequently, this fact simplifies the analysis of long-term behavior of the network.

3. Problem Setting

Consider two sets of states A, $B \subseteq \{0,1\}^n$. As mentioned above, each state $x \in \{0,1\}^n$ has a positive stationary probability $\pi(x)$. Thus, we can define $\pi(A) = \sum_{x \in A} \pi(x)$ and $\pi(B)$ similarly. Suppose that we are interested in altering the stationary probabilities of these two sets of states in such a way that the stationary probability of A is decreased and the stationary probability of B is increased by $0 < \lambda < 1$.[a] As already mentioned above, these two states may represent two different cellular functional states or phenotypes. In order to achieve this, suppose we alter function $f_{j_0}^{(i_0)}$ by replacing it with a new function $g_{j_0}^{(i_0)}$. The probability $c_{j_0}^{(i_0)}$ corresponding to $g_{j_0}^{(i_0)}$ must remain the same as for $f_{j_0}^{(i_0)}$, since $\sum_{j=1}^{l(i)} c_j^{(i)} = 1$.

[a]We could have also stated the problem in a more general manner, where instead of decreasing the stationary probability of A by λ and increasing that of B by λ, we would have specified the desired stationary probabilities of A and B separately, under the obvious constraint that their sum be less than or equal to 1. Indeed, this is what we will do in the simulation section.

Table 1. Truth tables of the predictors for the PBN in our example. The selection probabilities of each predictor are shown on the bottom row.

$x_1x_2x_3$	$f_1^{(1)}$	$f_2^{(1)}$	$f_1^{(2)}$	$f_1^{(3)}$	$f_2^{(3)}$
000	0	0	0	0	0
001	1	1	1	0	0
010	1	1	1	0	0
011	1	0	0	1	0
100	0	0	1	0	0
101	1	1	1	1	0
110	1	1	0	1	0
111	1	1	1	1	1
$c_j^{(i)}$	0.6	0.4	1	0.5	0.5

Thus, we have a new PBN whose stationary distribution we can denote by μ. Let $\mu(A)$ and $\mu(B)$ be the stationary probabilities of sets A and B under the altered PBN model. We can then pose the following optimization problem:

Given sets A and B, predictor functions $f_j^{(i)}$ together with their selection probabilities $c_j^{(i)}$, $i = 1, \ldots, n$, $j = 1, \ldots, l(i)$, and $\lambda \in (0, 1)$, select an i_0, j_0, and a function $g_{j_0}^{(i_0)}$ to replace $f_{j_0}^{(i_0)}$ such that

$$\varepsilon(\pi(A) - \lambda, \mu(A)) \tag{3}$$

and

$$\varepsilon(\pi(B) + \lambda, \mu(B)) \tag{4}$$

are minimum[b] among all i, j, $g_j^{(i)}$. Here, $\varepsilon(a, b)$ represents some chosen error function, such as the absolute error (i.e., $\varepsilon(a, b) = |a - b|$). An additional constraint can be that $g_{j_0}^{(i_0)}$ has no more essential variables than $f_{j_0}^{(i_0)}$. In this scenario, we are only allowing the alteration of one predictor function. More generally, we can preselect a number of predictor functions that we are willing to alter.

Let us illustrate these ideas with a small concrete example. We will use the same example as Example 1 in [12]. For convenience, we show the truth tables of the predictors and their selection probabilities in Table 1.

Assuming no perturbations ($p = 0$), the state transition diagram is shown in Fig. 1. As can be seen from this figure, two states, namely (000) and (111) are absorbing states. Let us hypothesize, for the sake of this example, that (111) corresponds to cell invasion (and rapid proliferation) and state (000) corresponds to quiescence. Let us fix a probability of perturbation $p = 0.01$. Then, a simple analysis based on the probability transition matrix reveals that the stationary probabilities of states (000) and (111) are 0.0752 and 0.7310, respectively. Thus, in the long run, the network will be in quiescence only 7% of the time and will be in proliferation 73%

[b]Since we want both (3) and (4) to be minimum, we can construct a single objective function that incorporates both. For example, the sum of the two functions would be an obvious choice.

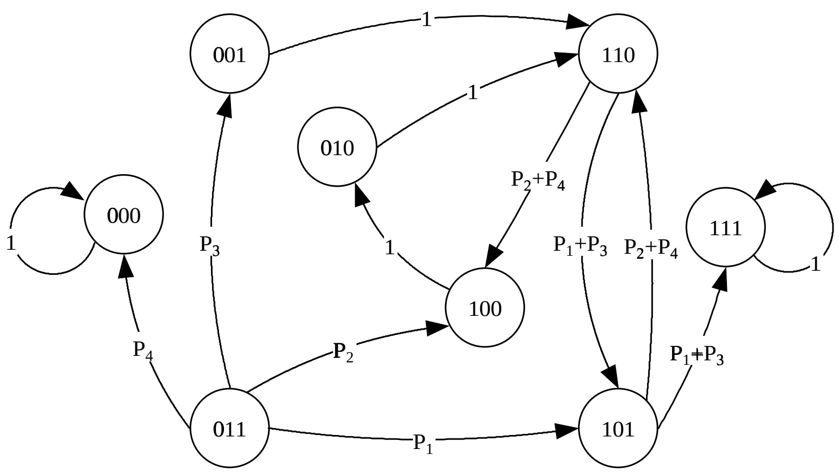

Fig. 1. State transition diagram (probability of perturbations is equal to zero).

of the time. Now, suppose we wish to alter this imbalance and require for the two stationary probabilities to be approximately 0.4 for both (000) and (111). The other 6 states will then be visited only 20% of the time. In the framework of the above optimization problem, $A = \{(111)\}$, $B = \{(000)\}$, $\pi(A) = 0.7310$, $\pi(B) = 0.0752$, $\mu(A) = \mu(B) = 0.4$, and $\lambda = 0.3279$. Finally, suppose we are allowed to change only one predictor function. In the truth tables shown in Table 1, this corresponds to changing only one of the columns, while keeping the selection probabilities $c_j^{(i)}$ unchanged. Thus, there are 5 possible columns (predictors) and 256 possibilities for each.

For the purposes of this example, we have generated each of the $5 \times 256 = 1280$ possible alterations. For each, we have computed and plotted the stationary probabilities $\mu(000)$ and $\mu(111)$, shown in Fig. 2. The optimal values of $\mu(000)$ and $\mu(111)$ for the error function $\varepsilon(a, b) = |a - b|$ are indicated by an arrow. The objective function to be minimized is

$$|\mu(000) - 0.4| + |\mu(111) - 0.4|\,, \tag{5}$$

which corresponds to the sum of the two objective functions in (3) and (4). The colors of the circles represent which predictor was altered. For example, the color red denotes that predictor $f_1^{(1)}$ was altered.

In this example, the optimal predictor is the one that alters $f_2^{(1)}$ for gene 1 (column 2 in the truth tables) and the truth table of the optimal predictor is $(00010101)^T$. This predictor achieves the stationary probabilities $\mu(000) = 0.4068$ and $\mu(111) = 0.4128$, which are quite close to the desired probabilities. The structure of the plot in Fig. 2 reveals an interesting phenomenon: the two stationary probabilities exhibit regularities, forming clusters of points arranged in a linear fashion, with different directions. In fact, this phenomenon has been observed in numerous

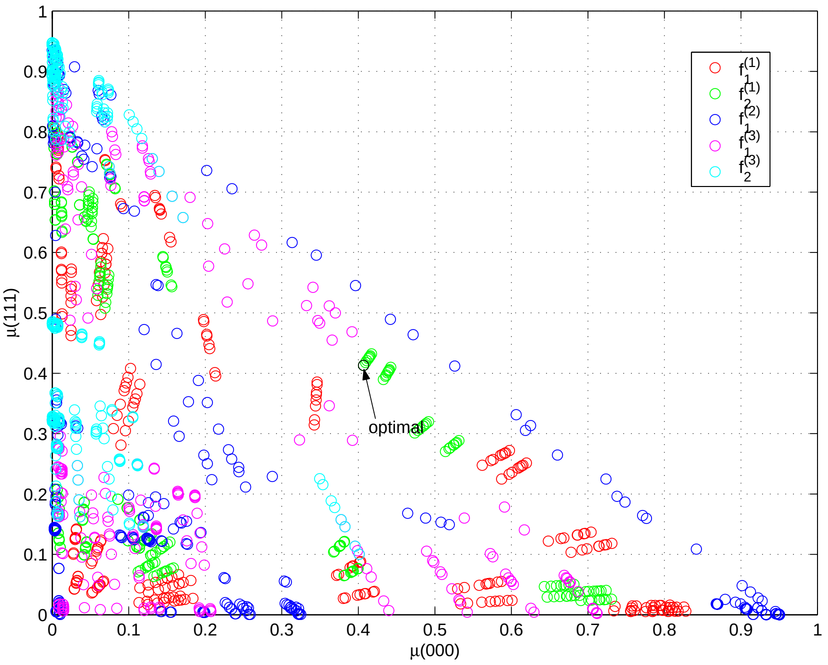

Fig. 2. Each circle represents one of the 1280 possible alterations to the predictors. The x-axis is $\mu(000)$ and the y-axis is $\mu(111)$. The optimal choice is shown with an arrow, as it comes closest to 0.4 for both stationary probabilities. The colors of the circles represent the predictor that is altered (see legend).

examples. As another example, Fig. 3 shows a 3-D scatter plot, corresponding to stationary probabilities of three different states, for a randomly generated PBN with 3 genes and 3 predictors per gene (a total of 9 predictors). As in Fig. 2, each circle represents the stationary probabilities after one of the predictors has been altered and the color of each circle represents which of the predictors got altered.

It is apparent, at least qualitatively, that the alterations of different predictors tend to occupy different parts of the space, implying that for a given predictor, there is a certain "range of action" that can be achieved by manipulating it. These facts, in turn, seem to suggest that a brute-force search for the optimal predictor alteration may very well be avoided. That is, a number of search directions should be followed simultaneously and the more promising ones should be explored further. Such a strategy is the hallmark of genetic algorithms (GAs), which have been successfully used in many optimization problems [2]. We turn to this next.

4. Solution via Genetic Algorithms

GAs, first pioneered by John Holland [4], are abstractions of biological evolution. In a GA, a population of chromosomes, which are represented as binary vectors (cf.,

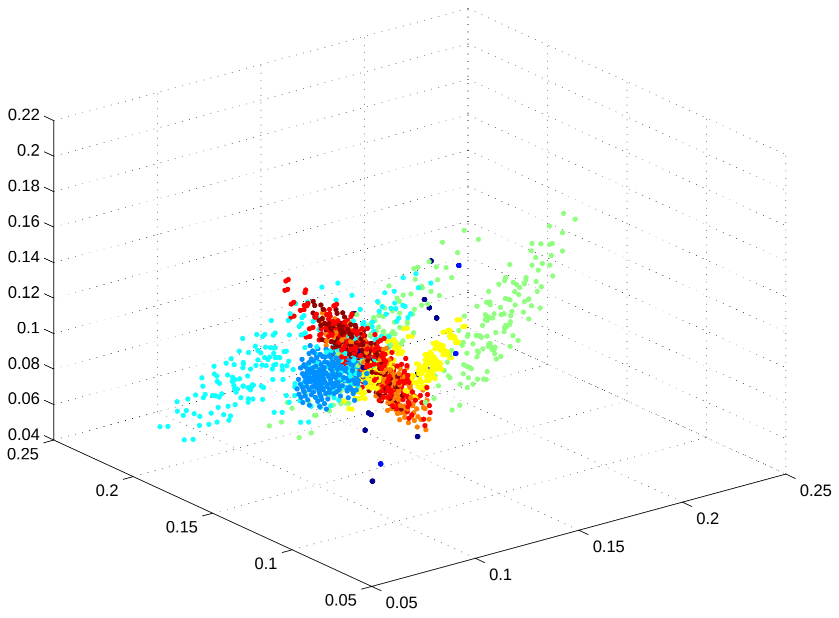

Fig. 3. Stationary probabilities of three different states, for a random PBN with 3 genes and 3 predictors per gene. Each circle represents the stationary probabilities after an alteration of one of the predictors. The color represents which of the predictors is altered.

truth tables of our gene predictors), moves to a new population of chromosomes using "natural selection" strategies, such as crossover and mutation. Some chromosomes are "selected" and allowed to reproduce and the more "fit" chromosomes produce more offspring. "Recombination" is achieved via the crossover operator, which exchanges parts of two chromosomes. Mutation can randomly change some of the locations or "alleles" (genes) in the chromosome and is very much like the perturbation in our Probabilistic Boolean Networks.

Chromosomes are used to encode possible solutions to an optimization problem. At each stage in a search for a solution, the choice of the possible chromosomes depends on the results of the previous stage of the search. That is, if the "parents" represent promising solutions in different regions of the search space, then their "offspring", produced by recombination, are likely to be promising solutions as well. The quality of a solution (chromosome) is represented by its fitness, which in the case of an optimization problem, is the value of the objective function, such as in (5), at that proposed solution.

The general operation of a GA can be described as follows. First, an initial population of chromosomes (encoded as binary vectors of a certain length) is generated and the fitness of each chromosome is calculated. Then, a pair of parent chromosomes is selected such that more fit chromosomes are more likely to be selected. With a certain probability, the two parents are recombined via the crossover operator at a random point (locus) in the chromosomes, producing two offspring. If the parents are not recombined, then their offspring are identical copies of the parents. Finally, the offspring are mutated at each locus with a certain mutation probability. This process is repeated until the number of offspring is equal to the number of parents and the old population is replaced by the new population, forming a new "generation". GAs typically make no assumptions about the characteristics of the problem, such as continuity or differentiability, and are well-suited for multi-modal function optimization as they are much less likely to get stuck in local optima. Thus, they are quite promising for finding good solutions to the problem described in Sec. 3.

Let us return to the example we considered above. A candidate solution essentially consists of two parts: the choice of the predictor to be altered (here, we have a total of $N = 5$ choices) and the truth table of the new, altered predictor. The former can be coded with $\lceil \log_2 N \rceil$ bits and the latter with 2^k bits, where k is the maximum number of input variables over all predictors. In the above example, $N = 5$ and $k = 3$, so each candidate solution (chromosome) would be encoded with $\lceil \log_2 5 \rceil + 2^3 = 11$ bits. One minor inconvenience is the fact that 3 bits is more than we need to code the choice of the predictor (5 possibilities). Thus, the mapping between the length-3 binary strings and the integers $1, \ldots, 5$ should be as uniform as possible so as to avoid certain predictors being chosen more often than others. This means that several different length-3 binary strings will code for the same integer. Of course, if N is a power of 2, this problem does not exist.

By applying a GA to this example, using the above encoding scheme and a fitness function given in (5), we obtained the correct result ($f_2^{(1)} = (00010101)^T$) typically after about 300 fitness function evaluations. Thus, only $300/1280 \approx 23\%$ of the work is performed, compared to the brute-force approach, even for such a small example. This is no small gain, considering that every fitness function evaluation entails recomputing the entire state-transition matrix and then finding its left eigenvector to obtain the stationary probabilities. When the numbers of genes and predictors per gene get large, the advantages of GAs should become dramatically more apparent. For example, in [7], GAs were used to design two-dimensional cellular automata for a density classification task. In that work, the search space consisted of 2^{512} states — a hyper-astronomical number that is not even remotely enumerable. Yet, the GA was able to find good solutions in a reasonable amount of time. Let us illustrate the approach with some more computer experiments.

4.1. *Computer experiments*

One of our goals is to assess the efficiency of the GAs for the problem of determining the optimal predictor, in terms of controlling the stationary probabilities of certain states. We measure the efficiency by counting the number of fitness function evaluations performed until the GA finds the correct predictor and dividing that number by the total number of possible predictor alterations. We note that many times, there are several optimal solutions. For example, it is possible that two different predictors (truth tables) for the same gene, or even for different genes, can produce the same value of the objective function (i.e., the same stationary probabilities). Any of these optimal solutions is considered to be the correct one.

As a first experiment, we constructed PBNs consisting of $n = 4$ genes, $l(i) = 2$ $(i = 1, \ldots, 4)$ predictors for each gene, and $k = 2$ input variables per predictor. We also observed the dependence of the efficiency on the probability of perturbation p. Thus, we have varied p between 10^{-4} and 10^{-1} in a logarithmically spaced manner. In total, 10 values of p were used. For each value of p, we constructed 100 randomly generated PBNs with the above parameters. For each PBN, we first found the optimal predictor alterations using a brute-force enumeration. Then, we applied the GA and recorded the number of fitness function evaluations necessary to obtain the optimal solution. The objective function was similar to the one in Eq. (5), except that the stationary probabilities of (0000) and (1111) were desired to be as small as possible (i.e., close to zero). In other words, the function to be minimized was $\mu(0000) + \mu(1111)$.

The results are shown in Fig. 4. The horizontal axis shows the different values of p. The vertical axis shows the number of evaluations necessary for the GA to

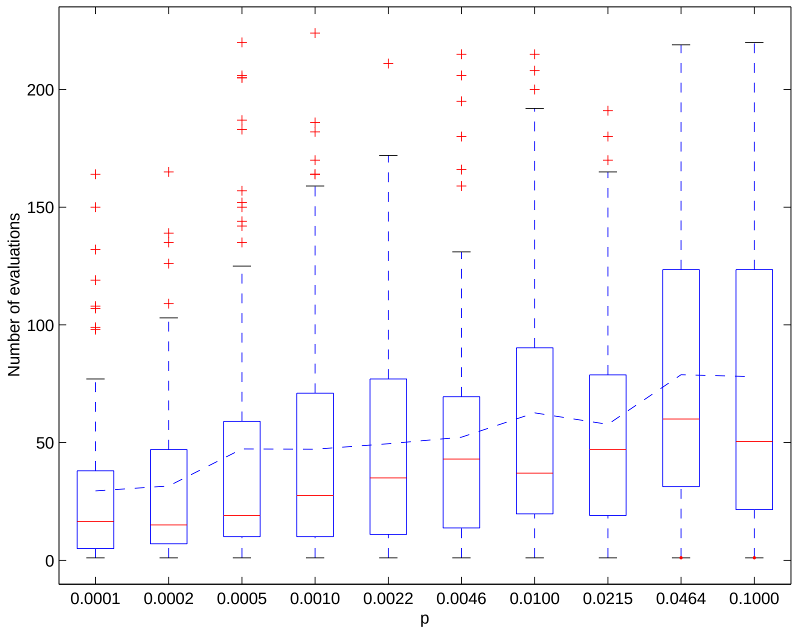

Fig. 4. A box and whisker plot showing the results of the computer experiment. The probability of perturbation logarithmically ranged between $p = 0.0001$ and $p = 0.1$. For each value of p, 100 random PBNs with 4 genes, 2 predictors for each gene, and 2 input variables per predictor, were produced. The vertical axis shows the number of fitness function evaluations necessary for the GA to obtain the optimal solution. The dotted line shows the mean number of evaluations.

obtain the optimal solution. The box and whisker plots have the usual meaning: they show the lower quartile, median, and upper quartile, and the whiskers extend to 1.5 times the inter-quartile range (IQR). The symbols "+" indicate those values that are beyond the ends of the whiskers. Finally, the dotted line shows the mean number of evaluations, for each value of p. On the average, the GA finds the optimal solution in much fewer steps than is required by brute-force enumeration. For example, for $p = 0.0001$, the median number of evaluations is 29, whereas the total number of possibilities is 128 — a 23% efficiency. At the same time, we see cases where the GA obviously failed. For example, for $p = 0.0001$, there are three cases which required more than 128 evaluations to obtain the optimal solution — in those cases, the GA was worse than a brute-force enumeration. Because the GA is a stochastic search method, such cases will always exist. The real advantage of GAs comes from the fact that they perform quite well *most* of the time.

We also see that the number of necessary evaluations tends to increase somewhat as a function of p. This can be seen from the medians as well as the means. One possible explanation of this phenomenon is that as p is increased, the underlying "structure" of the PBN becomes effaced and it is this very structure that the GA replies upon to quickly locate the optimum. Indeed, if we increase p to the unrealistically high value of $p = 0.5$, meaning that every gene gets determined by a fair coin flip, we would expect the GA to amount to nothing more than a random search. Finally, we also observe an increased IQR, with increasing p. Using the same reasoning as above, the ability of the GA to locate the optimal solution becomes compromised due to increased prevalence of randomness in the data.

Let us illustrate our approach with another simple example. Consider a PBN with $n = 5$ genes. We will use only one predictor per gene and will allow each predictor to have a full set of 5 input variables. Thus, the PBN is, in fact, a deterministic Boolean network. Since there are 5 predictors and each predictor has a truth table containing 32 bits, there are $2^{32} \times 5 \approx 21$ billion possible alterations. Clearly, even for this simple example, doing a full combinatorial search for the optimal alteration would be highly impractical.

Having generated a random PBN, with a perturbation probability of $p = 0.001$, we can compute its stationary distribution vector, which is shown in Fig. 5(a). As can be seen, this particular PBN has a probability of almost 1 of being in state (11010) (26 in decimal), in the long run. The only reason this probability is not exactly equal to 1 is because of perturbations (each gene gets flipped one in a thousand times). The network can find itself occasionally in other states, which consequently have very small, but non-zero stationary probabilities.[c] Since this network is just a deterministic Boolean network, the state (11010) is called an *attractor* or *fixed point* [9, 15]. Since there is only one such attractor, all the other states constitute its *basin of attraction.* Now, suppose that we would like to change the network, by changing only one rule (predictor), such that state (11101) (29 in

[c]In fact, we can say that $\lim_{p\to 0} \pi(11010) = 1$ and $\lim_{p\to 0} \pi(x) = 0$ for all $x \neq (11010)$.

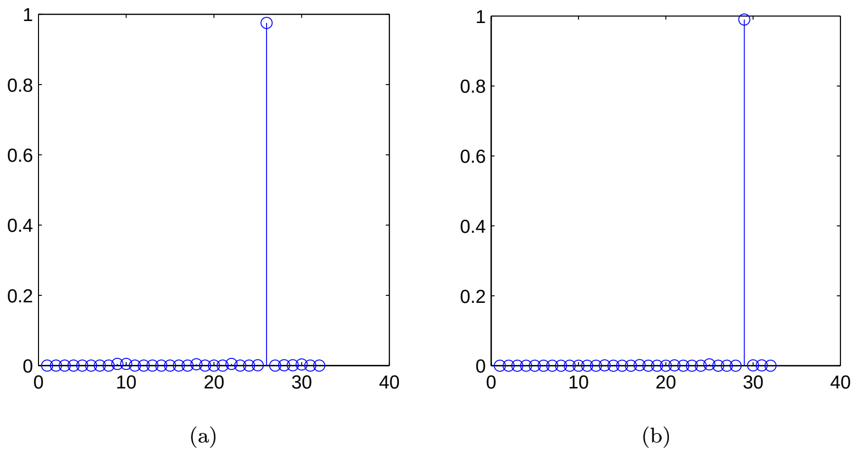

Fig. 5. (a) The stationary distribution vector for a Boolean network with 5 genes and $p = 0.001$. The only attractor is state (11010). (b) The stationary distribution vector after only one predictor is changed such that (11101) becomes the new attractor.

decimal) becomes the new and only attractor. Now, although we know that this is possible in advance, because of the way we have designed the network, the real test is to see whether the GA is able to determine the correct predictor. As an objective function, we used the Euclidean (L_2) norm between the solution and the length 32 unit vector that has a 1 in position 29. The GA finds the correct gene and determines the 32-bit truth table for its predictor after about 200 evaluations of the objective function. This is quite remarkable considering that there are over 21 billion possibilities. The resulting stationary distribution vector is shown in Fig. 5(b).

5. Concluding Remarks

We have developed a formalism for controlling long-run behavior of Probabilistic Boolean Networks. The formalism is, in fact, quite generic in the sense that any objective function, using any number of states and any number of possible predictors to be modified can be specified. Section 4.1 contains an example specifying the desired stationary probabilities for two particular states and another example specifying the stationary behavior for all states. The control of long-run behavior is accomplished by structural manipulation of the underlying rules of the network. In view of genetic regulatory networks, it is advisable to minimize this type of structural manipulation as much as possible, which translates into limiting the number of gene predictors that can be simultaneously modified or altered. Clearly, this seems to work against the goals of the optimization — the fewer choices we have (at any one particular time), the less power we have to alter the long-run behavior of the network. Thus, it may be the case that, given the constraints on the number of predictors that can be simultaneously altered, some solutions may not be feasible. However, in practice, we would only be interested in approximate solutions. For example, if we wish to make the stationary probability of a certain (undesirable) state be zero, we would be content if it was simply close to zero. Thus, many "nearly optimal" solutions may exist. Genetic algorithms indeed do not always guarantee a global optimum, but may find many good sub-optimal solutions, as they explore a number of promising regions of the search space.

In addition, we may wish to only allow the use of predictors belonging to a certain class of Boolean functions, such as canalizing functions or functions with a limited number of input variables. We can even insist that the new, altered predictor must have the same input variables as the predictor it is replacing. It is quite straightforward to incorporate such constraints during the GA optimization stage. Finally, for larger networks, parallel implementations of genetic algorithms can significantly improve the performance [3]. In such algorithms, a number of sub-populations evolve in parallel and some highly fit individuals can migrate between sub-populations. This carries additional benefit in that migrants from other sub-populations, which have evolved independently, help maintain genetic diversity. Perhaps not surprisingly, genetic algorithms, themselves inspired by biological evolution, hold great promise for tackling challenging problems in biology.

References

[1] Dougherty E. R., Kim S. and Chen Y., Coefficient of determination in nonlinear signal processing, *Signal Processing*, **80(10)** (2000) pp. 2219–2235.

[2] Goldberg D., *Genetic Algorithms in Search, Optimization, and Machine Learning* (Addison-Wesley, Reading, MA, 1989).

[3] Gorges-Schleuter M., Explicit parallelism of genetic algorithms through population structures. In *Parallel Problem Solving from Nature* (Springer Verlag, 1991) pp. 150–159.

[4] Holland J. H., *Adaptation in Natural and Artificial Systems* (University of Michigan Press, 1975; Second edition: MIT Press, 1992.)

[5] Huang S., Gene expression profiling, genetic networks, and cellular states: An integrating concept for tumorigenesis and drug discovery, *Journal of Molecular Medicine* **77** (1999) pp. 469–480.

[6] Huang S. and Ingber D. E., Regulation of cell cycle and gene activity patterns by cell shape: Evidence for attractors in real regulatory networks and the selective mode of cellular control, *InterJournal Genetics* (2000), MS: 238, http://www.interjournal.org.

[7] Jiménez-Morales F., Crutchfield J. P. and Mitchell M., Evolving two-dimensional cellular automata to perform density classification: A report on work in progress, *Parallel Computing* **27(5)** (2001) pp. 571–585.

[8] Kauffman S. A. and Levin S., Towards a general theory of adaptive walks on rugged landscapes, *Journal of Theoretical Biology* **128** (1987) pp. 11–45.

[9] Kauffman S. A., *The Origins of Order: Self-Organization and Selection in Evolution* (Oxford University Press, Oxford, 1993).

[10] Kim S., Dougherty E. R., Chen Y., Sivakumar K., Meltzer P., Trent J. M. and Bittner M., Multivariate measurement of gene expression relationships, *Genomics* **67** (2000) pp. 201–209.

[11] Kim S., Dougherty E. R., Bittner M. L., Chen Y., Sivakumar K., Meltzer P. and Trent J. M., General nonlinear framework for the analysis of gene interac-

tion via multivariate expression arrays, *Journal of Biomedical Optics* **5(4)** (2000) pp. 411–424.

[12] Shmulevich I., Dougherty E. R., Kim S. and Zhang W., Probabilistic Boolean Networks: A rule-based uncertainty model for gene regulatory networks, *Bioinformatics* **18(2)** (2002) pp. 261-274.

[13] Shmulevich I., Dougherty E. R., and Zhang W., Gene perturbation and intervention in Probabilistic Boolean Networks, *Bioinformatics*, in press.

[14] Vasudevan, N., Zhu Y. S., Daniel S., Koibuchi N., Chin W. W., and Pfaff D., Crosstalk between oestrogen receptors and thyroid hormone receptor isoforms results in differential regulation of the preproenkephalin gene, *Journal of Neuroendocrinology* **13** (2001) pp. 779–790.

[15] Wuensche A., Genomic regulation modeled as a network with basins of attraction. In *Proc. Pacific Symposium on Biocomputing* **3** (1998) pp. 89–102.

[16] Zhang Z. and Teng C. T., Estrogen receptor alpha and estrogen receptor-related receptor alpha1 compete for binding and coactivator, *Molecular and Cellular Endocrinology* **172** (2001) pp. 223–233.