Hindawi Publishing Corporation
EURASIP Journal on Bioinformatics and Systems Biology
Volume 2007, Article ID 32454, 15 pages
doi:10.1155/2007/32454

Research Article

Inference of a Probabilistic Boolean Network from a Single Observed Temporal Sequence

Stephen Marshall,[1] Le Yu,[1] Yufei Xiao,[2] and Edward R. Dougherty[2, 3, 4]

[1] *Department of Electronic and Electrical Engineering, Faculty of Engineering, University of Strathclyde, Glasgow, G1 1XW, UK*
[2] *Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843-3128, USA*
[3] *Computational Biology Division, Translational Genomics Research Institute, Phoenix, AZ 85004, USA*
[4] *Department of Pathology, University of Texas M. D. Anderson Cancer Center, Houston, TX 77030, USA*

Received 10 July 2006; Revised 29 January 2007; Accepted 26 February 2007

Recommended by Tatsuya Akutsu

The inference of gene regulatory networks is a key issue for genomic signal processing. This paper addresses the inference of probabilistic Boolean networks (PBNs) from observed temporal sequences of network states. Since a PBN is composed of a finite number of Boolean networks, a basic observation is that the characteristics of a single Boolean network without perturbation may be determined by its pairwise transitions. Because the network function is fixed and there are no perturbations, a given state will always be followed by a unique state at the succeeding time point. Thus, a transition counting matrix compiled over a data sequence will be sparse and contain only one entry per line. If the network also has perturbations, with small perturbation probability, then the transition counting matrix would have some insignificant nonzero entries replacing some (or all) of the zeros. If a data sequence is sufficiently long to adequately populate the matrix, then determination of the functions and inputs underlying the model is straightforward. The difficulty comes when the transition counting matrix consists of data derived from more than one Boolean network. We address the PBN inference procedure in several steps: (1) separate the data sequence into "pure" subsequences corresponding to constituent Boolean networks; (2) given a subsequence, infer a Boolean network; and (3) infer the probabilities of perturbation, the probability of there being a switch between constituent Boolean networks, and the selection probabilities governing which network is to be selected given a switch. Capturing the full dynamic behavior of probabilistic Boolean networks, be they binary or multivalued, will require the use of temporal data, and a great deal of it. This should not be surprising given the complexity of the model and the number of parameters, both transitional and static, that must be estimated. In addition to providing an inference algorithm, this paper demonstrates that the data requirement is much smaller if one does not wish to infer the switching, perturbation, and selection probabilities, and that constituent-network connectivity can be discovered with decent accuracy for relatively small time-course sequences.

1. INTRODUCTION

A key issue in genomic signal processing is the inference of gene regulatory networks [1]. Many methods have been proposed and these are specific to the network model, for instance, Boolean networks [2–5], probabilistic Boolean networks [6–9], and Bayesian networks [10–12], the latter being related to probabilistic Boolean networks [13]. The manner of inference depends on the kind of data available and the constraints one imposes on the inference. For instance, patient data do not consist of time-course measurements and are assumed to come from the steady state of the network, so that inference procedures cannot be expected to yield networks that accurately reflect dynamic behavior. Instead, one might just hope to obtain a set of networks whose steady state distributions are concordant, in some way, with the data. Since inference involves selecting a network from a family of networks, it can be beneficial to constrain the problem by placing restrictions on the family, such as limited attractor structure and limited connectivity [5]. Alternatively one might impose a structure on a probabilistic Boolean network that resolves inconsistencies in the data arising from mixing of data from several contexts [9].

This paper concerns inference of a probabilistic Boolean network (PBN) from a single temporal sequence of network states. Given a sufficiently long observation sequence, the goal is to infer a PBN that is a good candidate to have generated it. This situation is analogous to that of designing a Wiener filter from a single sufficiently long observation of a wide-sense stationary stochastic process. Here, we will be dealing with an ergodic process so that all transitional relations will be observed numerous times if the observed sequence is sufficiently long. Should one have the opportunity to observe multiple sequences, these can be used individually in the manner proposed and the results combined to provide the desired inference. Note that we say we desire a good candidate, not the only candidate. Even with constraints and a long sequence, there are many PBNs that could have produced the sequence. This is typical in statistical inference. For instance, point estimation of the mean of a distribution identifies a single value as the candidate for the mean, and typically the probability of exactly estimating the mean is zero. What this paper provides, and what is being provided in other papers on network inference, is an inference procedure that generates a network that is to some extent, and in some way, consistent with the observed sequence.

We will not delve into arguments about Boolean or probabilistic Boolean network modeling, these issues having been extensively discussed elsewhere [14–21]; however, we do note that PBN modeling is being used as a framework in which to apply control theory, in particular, dynamic programming, to design optimal intervention strategies based on the gene regulatory structure [22–25]. With current technology it is not possible to obtain sufficiently long data sequences to estimate the model parameters; however, in addition to using randomly generated networks, we will apply the inference to data generated from a PBN derived from a Boolean network model for the segment polarity genes in drosophila melanogaster [26], this being done by assuming that some genes in the existing model cannot be observed, so that they become latent variables outside the observable model and therefore cause the kind of stochasticity associated with PBNs.

It should be recognized that a key purpose of this paper is to present the PBN inference problem in a rigorous framework so that observational requirements become clear. In addition, it is hoped that a crisp analysis of the problem will lead to more approximate solutions based on the kind of temporal data that will become available; indeed, in this paper we propose a subsampling strategy that greatly mitigates the number of observations needed for the construction of the network functions and their associated regulatory gene sets.

2. PROBABILISTIC BOOLEAN NETWORKS

A *Boolean network* (BN) consists of a set of n variables, $\{x_0, x_1, \dots, x_{n-1}\}$, where each variable can take on one of two binary values, 0 or 1 [14, 15]. At any time point t ($t = 0, 1, 2, \dots$), the *state* of the network is defined by the vector $\mathbf{x}(t) = (x_0(t), x_1(t), \dots, x_{n-1}(t))$. For each variable x_i, there exist a *predictor set* $\{x_{i0}, x_{i1}, \dots, x_{i,k(i)-1}\}$ and a transition function f_i determining the value of x_i at the next time point,

$$x_i(t+1) = f_i(x_{i0}(t), x_{i1}(t), \dots, x_{i,k(i)-1}(t)), \tag{1}$$

where $0 \le i0 < i1 < \cdots < i, k(i) - 1 \le n - 1$. It is typically the case that, relative to the transition function f_i, many of the variables are nonessential, so that $k(i) < n$ (or even $k(i) \ll n$). Since the transition function is homogeneous in time, meaning that it is time invariant, we can simplify the notation by writing

$$x_i^+ = f_i(x_{i0}, x_{i1}, \dots, x_{i,k(i)-1}). \tag{2}$$

The n transition functions, together with the associated predictor sets, supply all the information necessary to determine the time evolution of the states of a Boolean network, $\mathbf{x}(0) \to \mathbf{x}(1) \to \cdots \to \mathbf{x}(t) \to \cdots$. The set of transition functions constitutes the *network function*, denoted as $\mathbf{f} = (f_0, \dots, f_{n-1})$.

Attractors play a key role in Boolean networks. Given a starting state, within a finite number of steps, the network will transition into a cycle of states, called an *attractor cycle* (or simply, attractor), and will continue to cycle thereafter. Nonattractor states are transient and are visited at most once on any network trajectory. The *level* of a state is the number of transitions required for the network to transition from the state into an attractor cycle. In gene regulatory modeling, attractors are often identified with phenotypes [16].

A *Boolean network with perturbation* (BNp) is a Boolean network altered so that, at any moment t, there is a probability P of randomly flipping a variable of the current state $\mathbf{x}(t)$ of the BN. An ordinary BN possesses a stationary distribution but except in very special circumstances does not possess a steady-state distribution. The state space is partitioned into sets of states called *basins*, each basin corresponding to the attractor into which its states will transition in due time. On the other hand, for a BNp there is the possibility of flipping from the current state into any other state at each moment. Hence, the BNp is ergodic as a random process and possesses a steady-state distribution. By definition, the attractor cycles of a BNp are the attractor cycles of the BN obtained by setting $P = 0$.

A *probabilistic Boolean network* (PBN) consists of a finite collection of Boolean networks with perturbation over a fixed set of variables, where each Boolean network is defined by a fixed network function and all possess common perturbation probability P [18, 20]. Moreover, at each moment, there is a probability q of switching out of the current Boolean network to a different constituent Boolean network, where each Boolean network composing the PBN has a probability (called *selection probability*) of being selected. If $q = 1$, then a new network function is randomly selected at each time point, and the PBN is said to be *instantaneously random*, the idea being to model uncertainty in model selection; if $q < 1$, then the PBN remains in a given constituent Boolean network until a network switch and the PBN is said to be *context sensitive*. The original introduction of PBNs considered only instantaneously random PBNs [18] and using this model PBNs were first used as the basis of applying control theory to optimal intervention strategies to drive network dynamics in favorable directions, such as away from metastatic states in cancer [22]. Subsequently, context-sensitive PBNs were introduced to model the randomizing effect of latent variables outside the network model and this leads to the development of optimal intervention strategies that take into account the effect of latent variables [23]. We defer to the literature for a discussion of the role of latent variables [1]. Our interest here is with context-sensitive PBNs, where q is assumed to be small, so that on average, the network is governed by a constituent Boolean network for some amount of time before switching to another constituent network. The perturbation parameter p and the switching parameter q will be seen to have effects on the proposed network-inference procedure.

By definition, the attractor cycles of a PBN are the attractor cycles of its constituent Boolean networks. While the attractor cycles of a single Boolean network must be disjoint, those of a PBN need not to be disjoint since attractor cycles from different constituent Boolean networks can intersect. Owing to the possibility of perturbation, a PBN is ergodic and possesses a steady-state distribution. We note that one can define a PBN without perturbation but we will not do so.

Let us close this section by noting that there is nothing inherently necessary about the quantization $\{0, 1\}$ for a PBN; indeed, PBN modeling is often done with the ternary quantization corresponding to a gene being down regulated (-1), up regulated (1), or invariant (0). For any finite quantization the model is still referred to as a PBN. In this paper we stay with binary quantization for simplicity but it should be evident that the methodology applies to any finite quantization, albeit, with greater complexity.

3. INFERENCE PROCEDURE FOR BOOLEAN NETWORKS WITH PERTURBATION

We first consider the inference of a single Boolean network with perturbation. Once this is accomplished, our task in the context of PBNs will be reduced to locating the data in the observed sequence corresponding to the various constituent Boolean networks.

3.1. *Inference based on the transition counting matrix and a cost function*

The characteristics of a Boolean network, with or without perturbation, can be estimated by observing its pairwise state transitions, $\mathbf{x}(t) \rightarrow \mathbf{x}(t+1)$, where $\mathbf{x}(t)$ can be an arbitrary vector from the n-dimensional state space $B^n = \{0, 1\}^n$. The states in B^n are ordered lexicographically according to $\{00 \cdots 0, 00 \cdots 1, \ldots, 11 \cdots 1\}$. Given a temporal data sequence $\mathbf{x}(0), \ldots, \mathbf{x}(N)$, a transition counting matrix C can be compiled over the data sequence showing the number c_{ij} of state transitions from the ith state to the jth state having occurred,

$$C = \begin{bmatrix} c_{00} & c_{01} & \cdots & c_{0,2^n-1} \\ c_{10} & c_{11} & \cdots & c_{1,2^n-1} \\ \cdots & \cdots & \cdots & \cdots \\ c_{2^n-1,0} & c_{2^n-1,1} & \cdots & c_{2^n-1,2^n-1} \end{bmatrix}. \tag{3}$$

If the temporal data sequence results from a BN without perturbations, then a given state will always be followed by a unique state at the next time point, and each row of matrix C contains at most one nonzero value. A typical nonzero entry will correspond to a transition of the form $a_0a_1 \cdots a_m \rightarrow b_0b_1 \cdots b_m$. If $\{x_{i0}, x_{i1}, \ldots, x_{i,k(i)-1}\}$ is the predictor set for x_i, because the variables outside the set $\{x_{i0}, x_{i1}, \ldots, x_{i,k(i)-1}\}$ have no effect on f_i, this tells us that $f_i(a_{i0}, a_{i1}, \ldots, a_{i,k(i)-1}) = b_i$ and one row of the truth table defining f_i is obtained. The single transition $a_0a_1 \cdots a_m \rightarrow b_0b_1 \cdots b_m$ gives one row of each transition function for the BN. Given deterministic nature of a BN, we will not be able to sufficiently populate the matrix C on a single observed sequence because, based on the initial state, the BN will transition into an attractor cycle and remain there. Therefore, we need to observe many runs from different initial states.

For a BNp with small perturbation probability, C will likely have some nonzero entries replacing some (or all) of the 0 entries. Owing to perturbation and the consequent ergodicity, a sufficiently long data sequence will sufficiently populate the matrix to determine the entries caused by perturbation, as well as the functions and inputs underlying the model. A mapping $\mathbf{x}(t) \rightarrow \mathbf{x}(t+1)$ will have been derived linking pairs of state vectors. This mapping induces n transition functions determining the state of each variable at time $t + 1$ as a function of its predictors at time t, which are precisely shown in (1) or (2). Given sufficient data, the functions and the set of essential predictors may be determined by Boolean reduction.

The task is facilitated by treating one variable at a time. Given any variable, x_i, and keeping in mind that some observed state transitions arise from random perturbations rather than transition functions, we wish to find the $k(i)$ variables that control x_i. The $k(i)$ input variables that most closely correlate with the behavior of x_i will be identified as the predictors. Specifically, the next state of variable x_i is a function of $k(i)$ variables, as in (2). The transition counting matrix will contain one large single value on each line (plus some "noise"). This value indicates the next state that follows the current state of the sequence. It is therefore possible to create a two-column next-state table with current-state column $x_0x_1 \cdots x_{n-1}$ and next-state column $x_0^+x_1^+ \cdots x_{n-1}^+$, there being 2^n rows in the table, a typical entry looking like $00101 \rightarrow 11001$ in the case of 5 variables. If the states are written in terms of their individual variables, then a mapping is produced from n variables to n variables, where the next state of any variable may be written as a function of all n input variables. The problem is to determine which subset consisting of $k(i)$ out of the n variables is the minimal set needed to predict x_i, for $i = 0, 1, \ldots, n-1$. We refer to the $k(i)$ variables in the minimal predictor set *essential predictors*.

To determine the essential predictors for a given variable, x_i, we will define a cost function. Assuming k variables are used to predict x_i, there are $n!/(n-k)!k!$ ways of choosing them. Each k with a choice of variables has a cost. By minimizing the cost function, we can identify k such that $k = k(i)$, as well as the predictor set. In a Boolean network without perturbation, if the value of x_i is fully determined

Table 1: Effect of essential variables.

Current state					Next state				
x_0	x_1	x_2	x_3	x_4	x_0^+	x_1^+	x_2^+	x_3^+	x_4^+
.	.	.	.	.					
0	0	1	1	0	1				
0	0	1	1	1	1				
.	.	.	.	.					
.	.	.		.	.				
.	.	.		.					
0	1	1	1	0	1				
				.	.				
0	1	1	1	1	1				
					.				
					.				

All inputs with same value of x_0, x_2, x_3 should result in the same output

by the predictor set, $\{x_{i0}, x_{i1}, \ldots, x_{i,k-1}\}$, then this set will not change for different combinations of the remaining variables, which are nonessential insofar as x_i is concerned. Hence, so long as $x_{i0}, x_{i1}, \ldots, x_{i,k-1}$ are fixed, the value of x_i should remain 0 or 1, regardless of the values of the remaining variables. For any given realization $(x_{i0}, x_{i1}, \ldots, x_{i,k-1}) = (a_{i0}, a_{i1}, \ldots, a_{i,k-1})$, $a_{ij} \in \{0, 1\}$, let

$$u_{i0,i1,\ldots,i(k-1)}(a_{i0}, a_{i1}, \ldots, a_{i,k-1}) = \sum_{x_{i0}=a_{i0},\ldots,x_{i,k-1}=a_{i,k-1}} x_i^+(x_0, x_1, \ldots, x_{n-1}). \tag{4}$$

According to this equation, $u_{i0,i1,\ldots,i(k-1)}(a_{i0}, a_{i1}, \ldots, a_{i,k-1})$ is the sum of the next-state values assuming $x_{i0}, x_{i1}, \ldots, x_{i,k-1}$ are held fixed at $a_{i0}, a_{i1}, \ldots, a_{i,k-1}$, respectively. There will be 2^{n-k} lines in the next-state table, where $(x_{i0}, x_{i1}, \ldots, x_{i,k-1}) = (a_{i0}, a_{i1}, \ldots, a_{i,k-1})$, while other variables can vary. Thus, there will be 2^{n-k} terms in the summation. For instance, for the example in Table 1, when $x_i = x_0$, $k = 3$, $i0 = 0$, $i1 = 2$, and $i2 = 3$, that is, $x_i^+ = f_i(x_0, *, x_2, x_3, *)$, we have

$$\begin{aligned} u_{10,12,13}(0,1,1) &= x_1^+(0,0,1,1,0) + x_1^+(0,0,1,1,1) \\ &\quad + x_1^+(0,1,1,1,0) + x_1^+(0,1,1,1,1). \end{aligned} \tag{5}$$

The term $u_{i0,i1,\ldots,i(k-1)}(a_{i0}, a_{i1}, \ldots, a_{i,k-1})$ attains its maximum (2^{n-k}) or minimum (0) if the value of x_i^+ remains unchanged on the 2^{n-k} lines in the next-state table, which is the case in the above example. Hence, the k inputs are good predictors of the function if $u_{i0,i1,\ldots,i(k-1)}(a_{i0}, a_{i1}, \ldots, a_{i,k-1})$ is close to either 0 or 2^{n-k}.

The cost function is based on the quantity

$$\begin{aligned} r_{i0,i1,\ldots,i(k-1)}&(a_{i0}, a_{i1}, \ldots, a_{i,k-1}) \\ &= u_{i0,i1,\ldots,i(k-1)}(a_{i0}, a_{i1}, \ldots, a_{i,k-1}) I \\ &\quad \left[u_{i0,i1,\ldots,i(k-1)}(a_{i0}, a_{i1}, \ldots, a_{i,k-1}) \le \frac{2^{n-k}}{2} \right] \\ &\quad + (2^{n-k} - u_{i0,i1,\ldots,i(k-1)}(a_{i0}, a_{i1}, \ldots, a_{i,k-1})) I \\ &\quad \left[u_{i0,i1,\ldots,i(k-1)}(a_{i0}, a_{i1}, \ldots, a_{i,k-1}) > \frac{2^{n-k}}{2} \right], \end{aligned} \tag{6}$$

where I is the characteristic function. Function $I(w) = 1$ if w is true and function $I(w) = 0$ if w is false. The term $r_{i0,i1,\ldots,i(k-1)}(a_{i0}, a_{i1}, \ldots, a_{i,k-1})$ is designed to be minimized if $u_{i0,i1,\ldots,i(k-1)}(a_{i0}, a_{i1}, \ldots, a_{i,k-1})$ is close to either 0 or 2^{n-k}. It represents a summation over one single realization of the variables $x_{i0}, x_{i1}, \ldots, x_{i,k-1}$. Therefore, we define the cost function R by summing the individual costs over all possible realizations of $x_{i0}, x_{i1}, \ldots, x_{i,k-1}$:

$$R(x_{i0}, x_{i1}, \ldots, x_{i,k-1}) = \sum_{a_{i0}, a_{i1}, \ldots, a_{i,k-1} \in \{0,1\}} r_{i0,i1,\ldots,i(k-1)}(a_{i0}, a_{i1}, \ldots, a_{i,k-1}). \tag{7}$$

The essential predictors for variable x_i are chosen to be the k variables that minimize the cost $R(x_{i0}, x_{i1}, \ldots, x_{i,k-1})$ and k is selected as the smallest integer to achieve the minimum. We emphasize on the smallest because if k ($k < n$) variables can perfectly predict x_i, then adding one more variable also achieves the minimum cost. For small numbers of variables, the k inputs may be chosen by a full search, with the cost function being evaluated for every combination. For larger numbers of variables, genetic algorithms can be used to minimize the cost function.

In some cases the next-state table is not fully defined, due to insufficient temporal data. This means that there are do-not-care outputs. Tests have shown that the input variables may still be identified correctly even for 90% of missing data.

Once the input set of variables is determined, it is straightforward to determine the functional relationship by Boolean minimization [27]. In many cases the observed data are insufficient to specify the behavior of the function for every combination of input variables; however, by setting the unknown states as do-not-care terms, an accurate approximation of the true function may be achieved. The task is simplified when the number k of input variables is small.

3.2. Complexity of the Procedure

We now consider the complexity of the proposed inference procedure. The truth table consists of n genes and therefore

Table 2: Values of $\Xi_{n,k}$.

k	n									
	5	6	7	8	9	10	15	20	30	50
2	11430	86898	5.84×10^5	3.61×10^6	2.11×10^7	1.18×10^8	4.23×10^{11}	1.04×10^{15}	3.76×10^{21}	1.94×10^{34}
3	16480	141210	1.06×10^6	7.17×10^6	4.55×10^7	2.74×10^8	1.34×10^{12}	4.17×10^{15}	5.52×10^{21}	1.74×10^{35}
4	17545	159060	1.28×10^5	9.32×10^6	6.35×10^7	4.09×10^8	2.71×10^{12}	1.08×10^{16}	6.47×10^{22}	1.09×10^{35}

Table 3: Computation times.

k	n						
	5	6	7	8	9	10	11
2	< 1 s	< 1 s	< 1 s	2 s	12 s	69 s	476 s
3	< 1 s	< 1 s	< 1 s	6 s	36 s	214 s	2109 s
4	< 1 s	< 1 s	< 1 s	9 s	68 s	472 s	3097 s

has 2^n lines. We wish to identify the k predictors which best describe the behavior of each gene. Each gene has a total of $C_k^n = n!/(n-k)!k!$ possible sets of k predictors. Each of these sets of k predictors has 2^k different combinations of values. For every specific combination there are 2^{n-k} lines of the truth table. These are lines where the predictors are fixed but the values of the other (nonpredictor) genes change. These must be processed according to (5), (6), and (7).

The individual terms in (5) are binary values, 0 or 1. The cost function in (7) is designed to be maximized when all terms in (5) are either all 0 or all 1; that is, the sum is either at its minimum or maximum value. Simulations have shown that this may be more efficiently computed by carrying out all pairwise comparisons of terms and recording the number of times they differ. Hence a summation has been replaced by a computationally more efficient series of comparison operations. The number of pairs in a set of 2^{n-k} values is $2^{n-k-1}(2^{n-k}-1)$. Therefore, the total number of comparisons for a given n and k is given by

$$\begin{aligned}\xi_{n,k} &= n\frac{n!}{(n-k)!k!}2^k 2^{n-k}2^{n-k-1}\left(2^{n-k}-1\right)\\ &= n\frac{n!}{(n-k)!k!}\cdot 2^{2n-k-1}\left(2^{n-k}-1\right).\end{aligned} \tag{8}$$

This expression gives the number of comparisons for a fixed value of k; however, if we wish to compute the number of comparisons for all values of predictors, up to and including k, then this is given by

$$\Xi_{n,k} = \sum_{j=1}^{k} n\frac{n!}{(n-j)!j!}2^{2n-j-1}\left(2^{n-j}-1\right). \tag{9}$$

Values for $\Xi_{n,k}$ are given in Table 2 and actual computation times taken on an Intel Pentium 4 with a 2.0 GHz clock and 768 MB of RAM are given in Table 3.

The values are quite consistent given the additional computational overheads not accounted for in (9). Even for 10 genes and up to 4 selectors, the computation time is less than 8 minutes. Because the procedure of one BN is not dependent on other BNs, the inference of multiple BNs can be run in parallel, so that time complexity is not an issue.

4. INFERENCE PROCEDURE FOR PROBABILISTIC BOOLEAN NETWORKS

PBN inference is addressed in three steps: (1) split the temporal data sequence into subsequences corresponding to constituent Boolean networks; (2) apply the preceding inference procedure to each subsequence; and (3) infer the perturbation, switching, and selection probabilities. Having already treated estimation of a BNp, in this section we address the first and third steps.

4.1. *Determining pure subsequences*

The first objective is to identify points within the temporal data sequence where there is a switch of constituent Boolean networks. Between any two successive switch points there will lie a *pure* temporal subsequence generated by a single constituent network. The transition counting matrix resulting from a sufficiently long pure temporal subsequence will have one large value in each row, with the remainder in each row being small (resulting from perturbation). Any measure of purity should therefore be maximized when the largest value in each row is significantly larger than any other value. The value of the transition counting matrix at row i and column j has already been defined in (3) as c_{ij}. Let the largest value of c_{ij} in row i be defined as c_i^1 and the second largest value be c_i^2. The quantity $c_i^1 - c_i^2$ is proposed as the basis of a *purity function* to determine the likelihood that the temporal subsequence lying between two data points is pure. As the quantity relates to an individual row of the transition matrix, it is summed over all rows and normalized by the total value of the elements to give a single value P for each matrix:

$$P = \frac{\sum_{i=0}^{2^n-1}\left(c_i^1 - c_i^2\right)}{\sum_{j=0}^{2^n-1}\sum_{i=0}^{2^n-1} c_{ij}}. \tag{10}$$

The purity function P is maximized for a state transition matrix when each row contains only one single large value and the remaining values on each row are zero.

To illustrate the purity function, consider a temporal data sequence of length N generated from two Boolean networks. The first section of the sequence, from 0 to N_1, has been generated from the first network and the remainder of the sequence, from N_1+1 to $N-1$, has been generated from the second network. We desire an estimate η of the switch point N_1. The variable η splits the data sequence into two parts and $0 \le \eta \le N-1$. The problem of locating the switch point, and hence partitioning the data sequence, reduces to a search to locate N_1. To accomplish this, a trial switch point, G, is varied and the data sets before and after

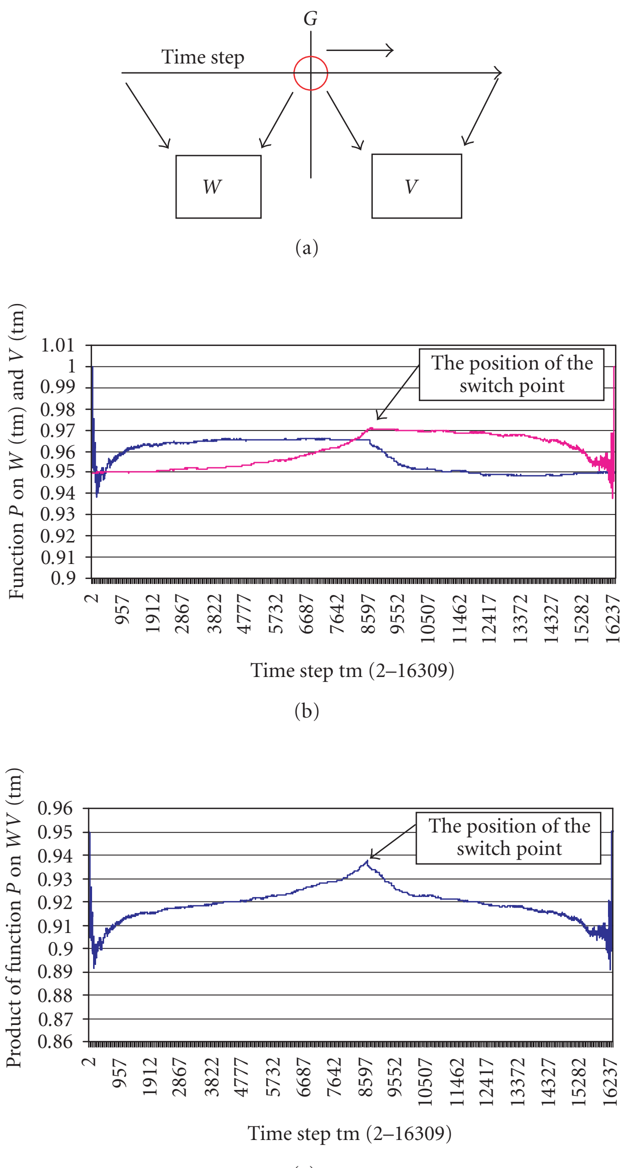


Figure 1: Switch point estimation: (a) data sequence divided by a sliding point G and transition matrices produced by for the data on each side of the partition; (b) purity functions from W and V; (c) simple function of two purity functions indicating switch point between models.

it are mapped into two different transition counting matrices, W and V. The ideal purity factor is a function which is maximized for both W and V when $G = N_1$. The procedure is illustrated in Figure 1. Figure 1(a) shows how the data are mapped from either side of a sliding point into the transition matrices. Figure 1(b) shows the purity functions derived from the transition counting matrices of W and V. Figure 1(c) shows a simple functional of W and V (in this case their product), which gives a peak at the correct switch point. The estimate η of the switch point is detected via a threshold.

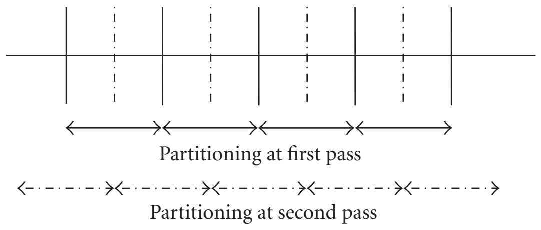


Figure 2: Passes for partitioning: the overall sequence is divided at the first pass into two shorter subsequences for testing. This is repeated in a second pass with the start and end points of the subsequences offset in order to avoid missing a switch point due to chaotic behavior.

The method described so far works well provided the sequence to be partitioned derives from two networks and the switch point does not lie close to the edge of the sequence. If the switch point lies close to the start or end of the sequence, then one of the transition counting matrices will be insufficiently populated, thereby causing the purity function to exhibit chaotic behavior.

If the data sequence is long and there is possibly a large number of switch points, then the sequence can be divided into a series of shorter subsequences that are individually tested by the method described. Owing to the effects of chaotic behavior near subsequence borders, the method is repeated in a second pass in which the sequence is again divided into shorter subsequences but with the start and end points offset (see Figure 2). This ensures that a switch point will not be missed simply because it lies close to the edge of the data subsequence being tested.

The purity function provides a measure of the difference in the relative behavior of two Boolean networks. It is possible that two Boolean networks can be different but still have many common transitions between their states. In this case the purity function will indicate a smaller distinction between the two models. This is particularly true where the two models have common attractors. Moreover, on average, the value of the purity function may vary greatly between subsequences. Hence, we apply the following normalization to obtain a normalized purity value:

$$P_{\text{norm}} = \frac{P - T}{T}, \tag{11}$$

where P is the purity value in the window and T is either the mean or geometric mean of the window values. The normalization removes differences in the ranges and average values of points in different subsequence, thereby making it easier to identify genuine peaks resulting from switches between Boolean networks.

If two constituent Boolean networks are very similar, then it is more difficult to distinguish them and they may be identified as being the same on account of insufficient or noisy data. This kind of problem is inherent to any inference procedure. If two networks are identified during inference, this will affect the switching probability because it will be based on the inferred model, which will have less constituent Boolean networks because some have been identified. In practice, noisy data are typically problematic owing to overfitting, the result being spurious constituent Boolean networks in the inferred model. This overfitting problem has been addressed elsewhere by using Hamming-distance filters to identify close data profiles [9]. By identifying similar networks, the current proposed procedure acts like a lowpass filter and thereby mitigates overfitting. As with any lowpass filter, discrimination capacity is diminished.

4.2. *Estimation of the switching, selection, and perturbation probabilities*

So far we have been concerned with identifying a family of Boolean networks composing a PBN; much longer data sequences are required to estimate the switching, selection, and perturbation probabilities. The switching probability may be estimated simply by dividing the number of switch points found by the total sequence length. The perturbation probability is estimated by identifying those transitions in the sequence not determined by a constituent-network function. For every data point, the next state is predicted using the model that has been found. If the predicted state does not match the actual state, then it is recorded as being caused by perturbation. Switch points are omitted from this process. The perturbation rate is then calculated by dividing the total instances of perturbation by the length of the data sequence.

Regarding the selection probabilities, we assume that a constituent network cannot switch into itself; otherwise there would be no switch. This assumption is consistent with the heuristic that a switch results from the change of a latent variable that in turn results in a change of the network structure. Thus, the selection probabilities are conditional, depending on the current network. The conditional probabilities are of the form q_{AB}, which gives the probability of selecting network B during a switch, given the current network is A, and q_{AB} is estimated by dividing the number of times the data sequence switches from A to B by the number of times it switches out of A.

In all cases, the length N of the sequence necessary to obtain good estimates is key. This issue is related to how often we expect to observe a perturbation, network switch, or network selection during a data sequence. It can be addressed in terms of the relevant network parameters.

We first consider estimation of the perturbation probability P. Note that we have defined P as the probability of making a random state selection, whereas in some papers each variable is given a probability of randomly changing. If the observed sequence has length N and we let X denote the number of perturbations (0 or 1) at a given time point, then the mean of X is p and the estimate, $\widehat{p}$, we are using for p is the sample mean of X for a random sample of size N, the sample being random because perturbations are independent. The expected number of perturbations is Np, which is the mean of the random variable S given by an independent sum of N random variables identically distributed to X. S possesses a binomial distribution with variance $Np(1-p)$.

A measure of goodness of the estimator is given by

$$P(|p-\widehat{p}|<\varepsilon)=P(|Np-S|<N\varepsilon) \tag{12}$$

for $\varepsilon > 0$. Because S possesses a binomial distribution, this probability is directly expressible in terms of the binomial density, which means that the goodness of our estimator is completely characterized. This computation is problematic for large N, but if N is sufficiently large so that the rule-of-thumb $\min\{Np, N(1-p)\} > 5$ is satisfied, then the normal approximation to the binomial distribution can be used.

Chebyshev's inequality provides a lower bound:

$$\begin{aligned} P(|p-\widehat{p}|<\varepsilon) &= 1-P(|Np-S|\geq N\varepsilon) \\ &\geq 1-\frac{p(1-p)}{N\varepsilon^2}. \end{aligned} \tag{13}$$

A good estimate is very likely if N is sufficiently large to make the fraction very small. Although often loose, Chebyshev's inequality provides an asymptotic guarantee of goodness. The salient issue is that the expected number of perturbations (in the denominator) becomes large.

A completely analogous analysis applies to the switching probability q, with q replacing p and $\widehat{q}$ replacing $\widehat{p}$ in (12) and (13), with Nq being the expected number of switches.

To estimate the selection probabilities, let p_{ij} be the probability of selecting network B_j given a switch is called for and the current network is B_i, $\widehat{p}_{ij}$ its estimator, r_i the probability of observing a switch out of network B_i, $\widehat{r}_i$ the estimator of r_i formed by dividing the number of times the PBN is observed switching out of B_i divided by N, s_{ij} the probability of observing a switch from network B_i to network B_j, and $\widehat{s}_{ij}$ the estimator of s_{ij} formed by dividing the number of times the PBN is observed switching out of B_i into B_j by N. The estimator of interest, $\widehat{p}_{ij}$, can be expressed as $\widehat{s}_{ij}/\widehat{r}_i$. The probability of observing a switch out of B_i is given by $qP(B_i)$, where $P(B_i)$ is the probability that the PBN is in B_i, so that the expected number of times such a switch is observed is given by $NqP(B_i)$. There is an obvious issue here: $P(B_i)$ is not a model parameter. We will return to this issue.

Let us first consider $\widehat{s}_{ij}$. Define the following events: A^t is a switch at time t, B_i^t is the event of the PBN being in network B_i at time t, and $[B_i \to B_j]^t$ is the event B_i switches to B_j at time t. Then, because the occurrence of a switch is independent of the current network,

$$\begin{aligned} P\left(\left[B_i \longrightarrow B_j\right]^t\right) &= P(A^t)P(B_i^{t-1})P\left(\left[B_i \longrightarrow B_j\right]^t \mid B_i^{t-1}\right) \\ &= qP(B_i^{t-1})p_{ij}. \end{aligned} \tag{14}$$

The probability of interest depends on the time, as does the probability of being in a particular constituent network; however, if we assume the PBN is in the steady state, then the time parameters drop out to yield

$$P\left(\left[B_i \longrightarrow B_j\right]^t\right) = qP(B_i)p_{ij}. \tag{15}$$

Therefore the number of times we expect to see a switch from B_i to B_j is given by $NqP(B_i)p_{ij}$.

Let us now return to the issue of $P(B_i)$ not being a model parameter. In fact, although it is not directly a model parameter, it can be expressed in terms of the model parameters so long as we assume we are in the steady state. Since

$$B_i^t = \left[(A^t)^c \cap B_i^{t-1}\right] \cup \left(A^t \cap \bigcup_{j\neq i} B_j^{t-1} \cap \left[B_j \longrightarrow B_i\right]^t\right) \tag{16}$$

a straightforward probability analysis yields

$$\begin{aligned} P(B_i^t) &= (1-q)P(B_i^{t-1}) \\ &\quad + q\sum_{j\neq i} P(B_j^{t-1})P\left(\left[B_j \longrightarrow B_i\right]^t \mid B_j^{t-1}\right). \end{aligned} \tag{17}$$

Under the steady-state assumption the time parameters may be dropped to yield

$$P(B_i) = \sum_{j\neq i} p_{ji}P(B_j). \tag{18}$$

Hence, the network probabilities are given in terms of the selection probabilities by

$$\mathbf{0} = \begin{pmatrix} -1 & p_{21} & \cdots & p_{m1} \\ p_{12} & -1 & \cdots & p_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ p_{1m} & p_{2,m-1} & \cdots & -1 \end{pmatrix} \begin{pmatrix} P(B_1) \\ P(B_2) \\ \vdots \\ P(B_m) \end{pmatrix}. \tag{19}$$

5. EXPERIMENTAL RESULTS

A variety of experiments have been performed to assess the proposed algorithm. These include experiments on single BNs, PBNs, and real data. Insofar as the switching, selection, and perturbation probabilities are concerned, their estimation has been characterized analytically in the previous section so we will not be concerned with them here.

Thus, we are concerned with the percentages of the predictors and functions recovered from a generated sequence. Letting c_p and t_p be the number of predictors correctly identified and the total number of predictors in the network, respectively, the percentage, π_p, of predictors correctly identified is given by

$$\pi_p = \frac{c_p}{t_p} \times 100. \tag{20}$$

Letting c_f and t_f be the number of function outputs correctly identified and the total number of function outputs in network, respectively, the percentage, π_f, of function outputs correctly identified is given by

$$\pi_f = \frac{c_f}{t_f} \times 100. \tag{21}$$

The functions may be written as truth tables and π_f corresponds to the percentage of lines in all the truth tables recovered from the data which correctly match the lines of the truth tables for the original function.

TABLE 4: Average percentage of predictors and functions recovered from 104 BN sequences consisting of $n = 7$ variables for $k = 2$ and $k = 3$, and $P = .01$.

Sequence length	Model recovery			
	Predictors recovered (%)		Functions recovered (%)	
	$k = 2$	$k = 3$	$k = 2$	$k = 3$
500	46.27	21.85	34.59	12.26
1000	54.33	28.24	45.22	19.98
2000	71.71	29.84	64.28	22.03
4000	98.08	34.87	96.73	28.53
6000	98.11	50.12	97.75	42.53
8000	98.18	50.69	97.87	43.23
10 000	98.80	51.39	98.25	43.74
20 000	100	78.39	98.333	69.29
30 000	100	85.89	99.67	79.66
40 000	100	87.98	99.75	80.25

5.1. Single Boolean networks

When inferring the parameters of single BNs from data sequences by our method, it was found that the predictors and functions underlying the data could be determined very accurately from a limited number of observations. This means that even when only a small number of the total states and possible transitions of the model are observed, the parameters can still be extracted.

These tests have been conducted using a database of 80 sequences generated by single BNs with perturbation. These have been constructed by randomly generating 16 BNs with $n = 7$ variables and connectivity $k = 2$ or $k = 3$, and $P = .01$. The sequence lengths vary in 10 steps from 500 to 40 000, as shown in Table 4. The table shows the percentages of the predictors and functions recovered from a sequence generated by a single BN, that is, a pure sequence with $n = 7$, for $k = 2$ or $k = 3$, expressed as a function of the overall sequence length. The average percentages of predictors and functions recovered from BN sequences with $k = 2$ is much higher than for $k = 3$ in the same sequence length.

5.2. Probabilistic Boolean networks

For the analysis of PBN inference, we have constructed two databases consisting of sequences generated by PBNs with $n = 7$ genes.

(i) Database A: the sequences are generated by 80 randomly generated PBNs and sequence lengths vary in 10 steps from 2000 to 500 000, each with different values of p and q, and two different levels of connectivity k.

(ii) Database B: 200 sequences of length 100 000 are generated from 200 randomly generated PBNs, each having 4 constituent BNs with $k = 3$ predictors. The switching probability q varies in 10 values: .0001, .0002, .0005, .001, .002, .005, .01, .02, .05, 0.1.

The key issue for PBNs is how the inference algorithm works relative to the identification of switch points via the purity function. If the data sequence is successfully partitioned into pure sequences, each generated by a constituent BN, then the BN results show that the predictors and functions can be accurately determined from a limited number of observations. Hence, our main concern with PBNs is apprehending the effects of the switching probability q, perturbation probability p, connectivity k, and sequence length. For instance, if there is a low switching probability, say $q = .001$, then the resulting pure subsequences may be several hundred data points long. So while each BN may be characterized from a few hundred data points, it may be necessary to observe a very long sequence simply to encounter all of the constituent BNs.

When analyzing long sequences there are two strategies that can be applied after the data have been partitioned into pure subsequences.

(1) Select one subsequence for each BN and analyze that only.

(2) Collate all subsequences generated by the same BN and analyze each set.

Using the first strategy, the accuracy of the recovery of the predictors and functions tends to go down as the switching probability goes up because the lengths of the subsequences get shorter as the switching probability increases. Using the second strategy, the recovery rate is almost independent of the switching probability because the same number of data points from each BN is encountered. They are just cut up into smaller subsequences. Past a certain threshold, when the switching probability is very high the subsequences are so short that they are hard to classify.

Figure 3 shows a graph of predictor recovery as a function of switching probability for the two strategies using database B. Both strategies give poor recovery for low switching probability because not all of the BNs are seen. Strategy 2 is more effective in recovering the underlying model parameters over a wider range of switching values. For higher values of q, the results from strategy 1 decline as the subsequences get shorter. The results for strategy 2 eventually decline as the sequences become so short that they cannot be effectively classified.

These observations are borne out by the results in Figure 4, which show the percentage of predictors recovered using strategy 2 from a PBN-generated sequence with 4 BNs consisting of $n = 7$ variables with $k = 3$, $P = .01$, and switching probabilities $q = .001$ and $q = .005$ for various length sequences using database A. It can be seen that for low sequence lengths and low probability, only 21% of the predictors are recovered because only one BN has been observed. As sequence length increases, the percentage of predictors recovered increases and at all times the higher switching probability does best, with the gap closing for very long sequence lengths.

More comparisons are given in Figures 5 and 6, which compare the percentage predictor recovery for two different connectivity values and for two different perturbation values, respectively. They both result from strategy 2 applied to database A. It can be seen that it is easier to recover predictors for smaller values of k and larger values of p.

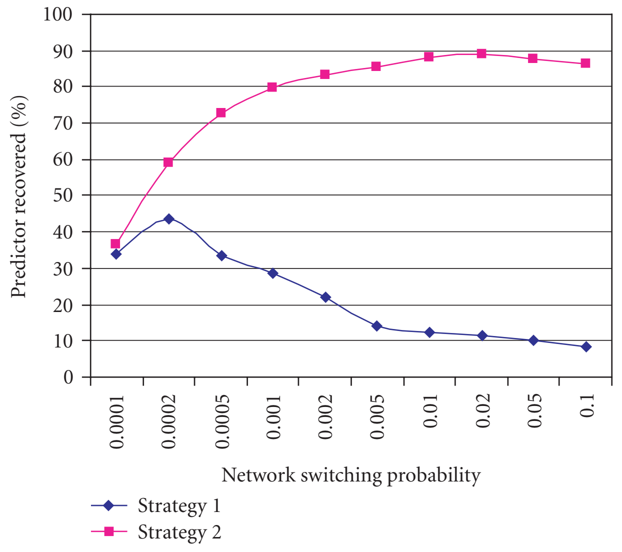


FIGURE 3: The percentage of predictors recovered from fixed length PBN sequences (of 100 000 sample points). The sequence is generated from 4 BNs, with $n = 7$ variables and $k = 3$ predictors, and $P = .01$.

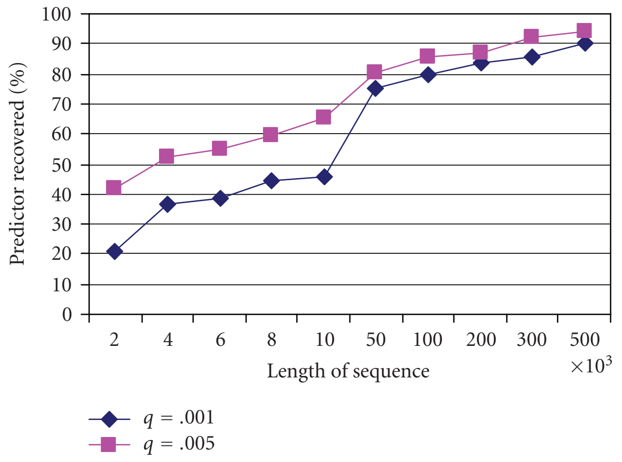


FIGURE 4: The percentage of predictors recovered using strategy 2 from a sequence generated from a PBN with 4 BNs consisting of $n = 7$ variables with $k = 3$, $P = .01$ and switching probabilities $q = .001$ and $q = .005$ for various length sequences.

A fuller picture of the recovery of predictors and functions from a PBN sequence of varying length, varying k, and varying switching probability is given in Table 5 for database A, where $P = .01$ and there are three different switching probabilities: $q = .001, .005, .03$. As expected, it is easier to recover predictors for low values of k. Also over this range the percentage recovery of both functions and predictors increases with increasing switching probability.

TABLE 5: The percentage of predictors recovered by strategy 2 as a function of various length sequences from sequences generated by experimental design A with at $P = .01$, switching probabilities, $q = .001, .005, .03$, and for $k = 2$ and $k = 3$.

Sequence length	$q = .001$				$q = .005$				$q = .03$			
	Predictor recovered (%)		Functions recovered (%)		Predictor recovered (%)		Functions recovered (%)		Predictor recovered (%)		Functions recovered (%)	
	$k = 2$	$k = 3$	$k = 2$	$k = 3$	$k = 2$	$k = 3$	$k = 2$	$k = 3$	$k = 2$	$k = 3$	$k = 2$	$k = 3$
2000	22.07	20.94	20.15	12.95	50.74	41.79	37.27	25.44	65.25	48.84	53.52	34.01
4000	36.90	36.31	33.13	23.89	55.43	52.54	42.49	37.06	74.88	56.08	66.31	42.72
6000	53.59	38.80	43.23	26.79	76.08	54.92	66.74	42.02	75.69	64.33	67.20	51.97
8000	54.75	44.54	47.15	29.42	77.02	59.77	67.48	45.07	76.22	67.86	67.72	55.10
10 000	58.69	45.63	53.57	36.29	79.10	65.37	69.47	51.94	86.36	73.82	80.92	61.84
50 000	91.50	75.03	88.22	65.29	94.58	80.07	92.59	71.55	96.70	86.64	94.71	78.32
100 000	97.28	79.68	95.43	71.19	97.97	85.51	96.47	78.34	98.47	90.71	96.68	85.06
200 000	97.69	83.65	96.39	76.23	98.68	86.76	97.75	80.24	99.27	94.02	98.03	90.79
300 000	97.98	85.62	96.82	79.00	99.00	92.37	98.19	88.28	99.40	95.50	98.97	92.50
500 000	99.40	89.88	98.67	84.85	99.68	93.90	99.18	90.30	99.83	96.69	99.25	94.21

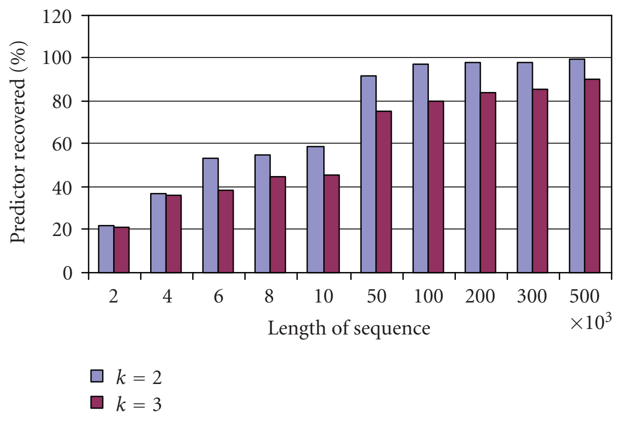


FIGURE 5: The percentage of predictors recovered using strategy 2 and experimental design A as a function of sequence length for connectivities $k = 2$ and $k = 3$.

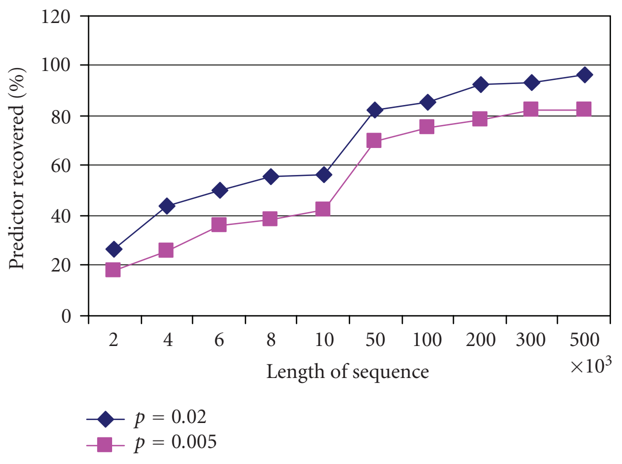


FIGURE 6: The percentage of predictors recovered using strategy 2 and experimental design A as a function of sequence length for perturbation probabilities $P = .02$ and $P = .005$.

We have seen the marginal effects of the switching and perturbation probabilities, but what about their combined effects? To understand this interaction, and to do so taking into account both the number of genes and the sequence length, we have conducted a series of experiments using randomly generated PBNs composed of either $n = 7$ or $n = 10$ genes, and possessing different switching and perturbation values. The result is a set of surfaces giving the percentages of predictors recovered as a function of p and q.

The PBNs have been generated according to the following protocol.

(1) Randomly generate 80 BNs with $n = 7$ variables and connectivity $k = 3$ (each variable has at most 3 predictors, the number for each variable being randomly selected). Randomly order the BNs as A1, A2, ..., A80.

(2) Consider the following perturbation and switching probabilities: $P = .005$, $P = .01$, $P = .015$, $P = .02$, $q = .001$, $q = .005$, $q = .01$, $q = .02$, $q = .03$.

(3) For each p, q, do the following: (1) construct a PBN from A1, A2, A3, A4 with selection probabilities 0.1, 0.2, 0.3, 0.4, respectively; (2) construct a PBN from A5, A6, A7, A8 with selection probabilities 0.1, 0.2, 0.3, 0.4, respectively; (3) continue until the BNs are used up.

(4) Apply the inference algorithm to all PBNs using data sequences of length $N = 4000, 6000, 8000, 10\,000, 50\,000$.

(5) Repeat the same procedure from (1)–(4) using 10 variables.

Figures 7 and 8 show fitted surfaces for $n = 7$ and $n = 10$, respectively. We can make several observations in the parameter region considered: (a) as expected, the surface heights increase with increasing sequence length; (b) as expected, the surface heights are lower for more genes, meaning that longer sequences are needed for more genes; (c) the surfaces tend to increase in height for both p and q, but if q is too large, then recovery percentages begin to decline. The trends are the same for both numbers of genes, but recovery requires increasingly long sequences for larger numbers of genes.

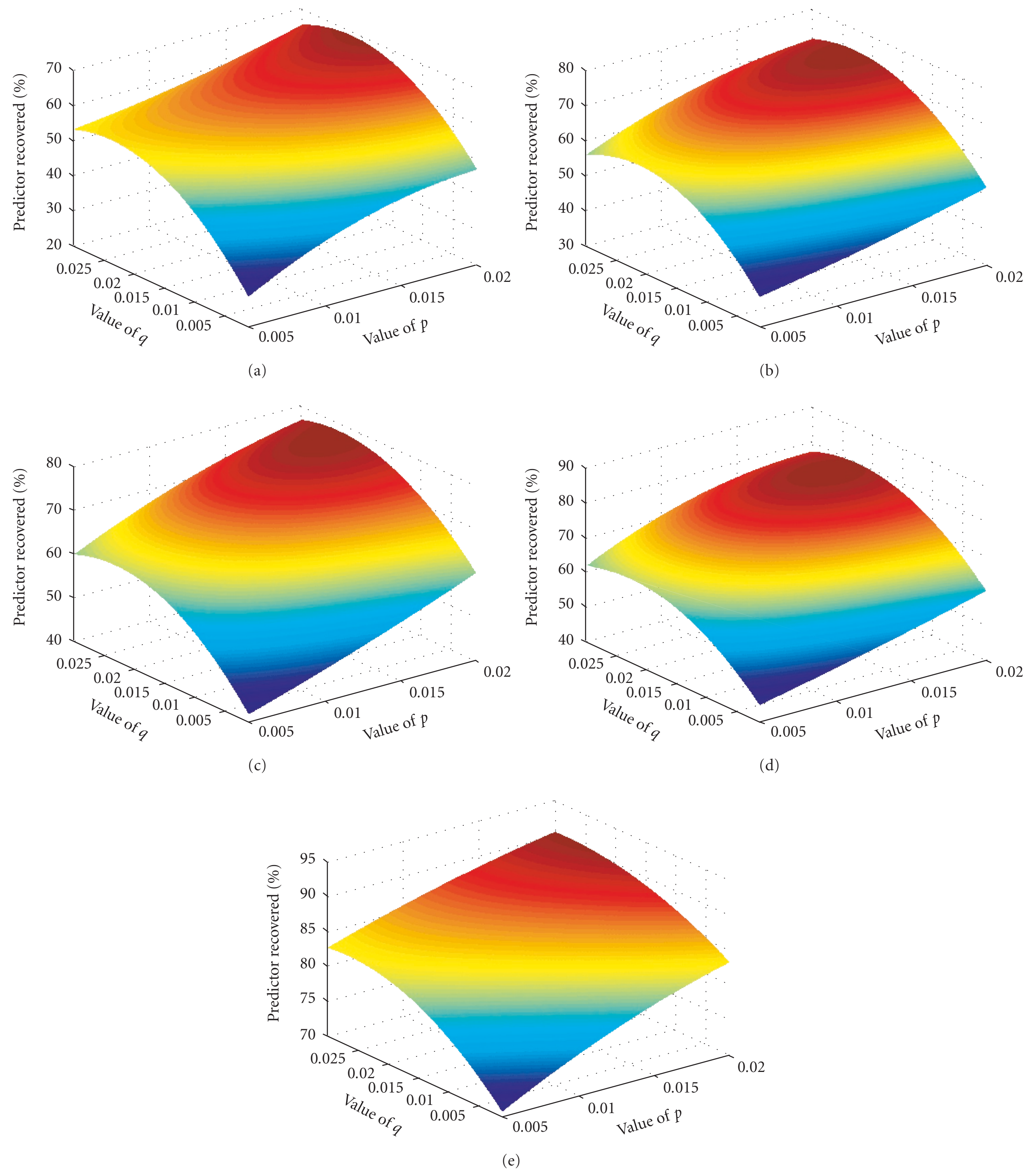
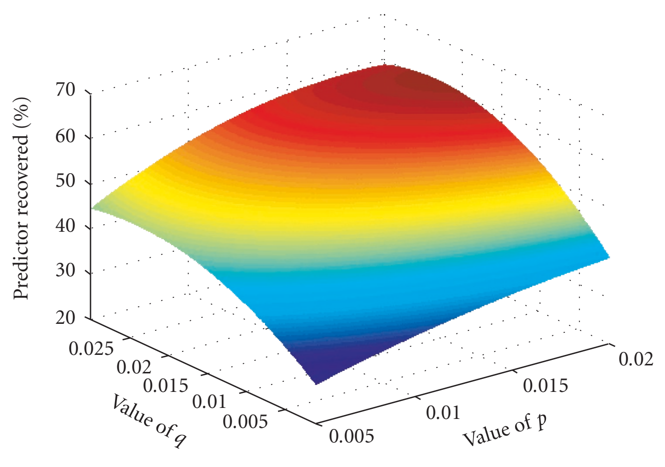


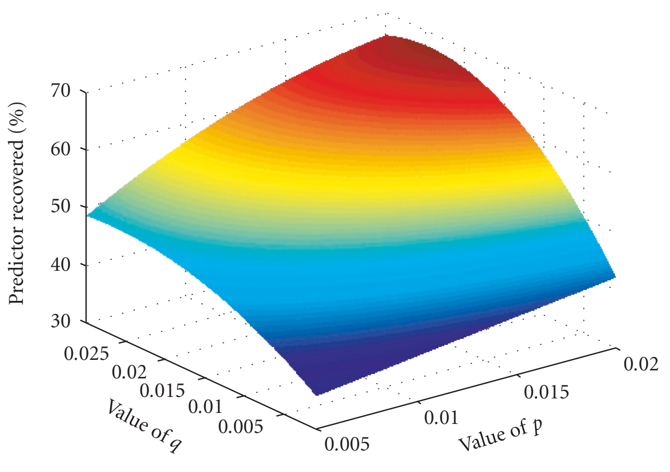
Figure 7: Predictor recovery as a function of switching and perturbation probabilities for $n = 7$ genes: (a) $N = 4000$, (b) $N = 6000$, (c) $N = 8000$, (d) $N = 10\,000$, (e) $N = 50\,000$.

6. A SUBSAMPLING STRATEGY

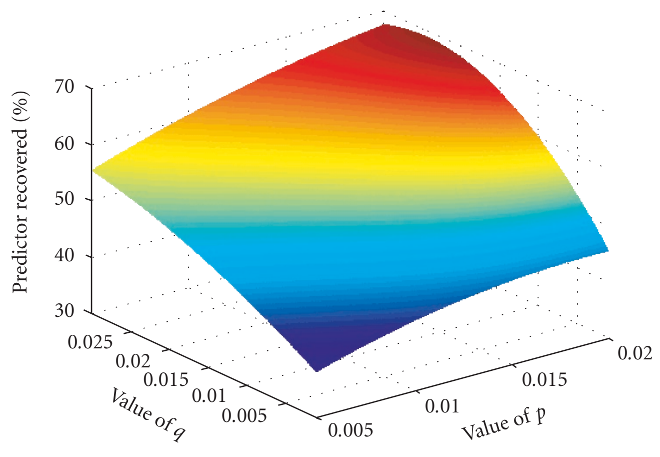
It is usually only necessary to observe a few thousand sample points in order to determine the underlying predictors and functions of a single BN. Moreover, it is usually only necessary to observe a few hundred sample points to classify a BN as being BN1, BN2, and so forth. However, in analyzing a PBN-generated sequence with low switching probability, say



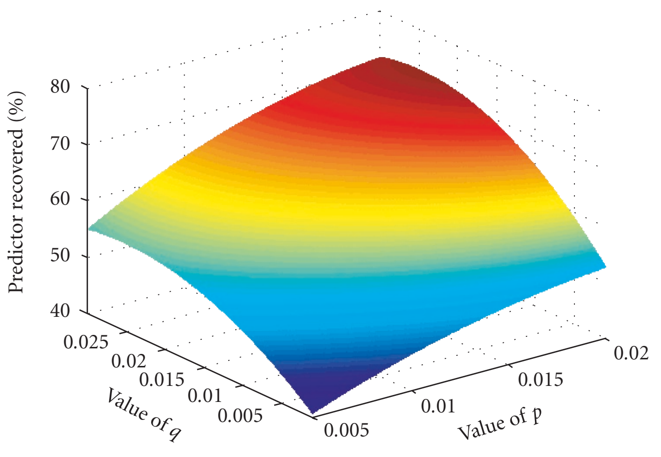
(a)



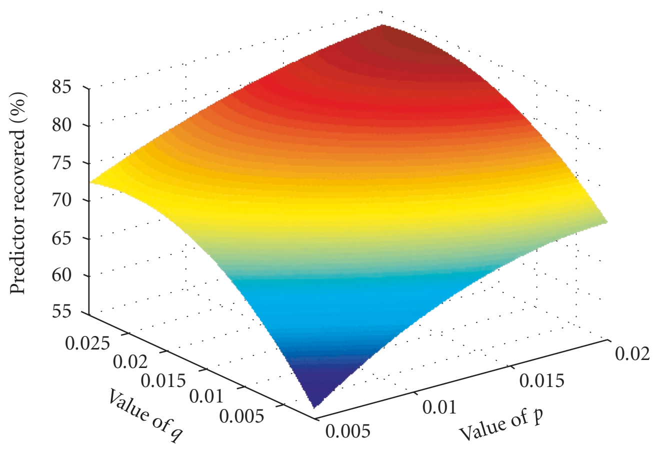
(b)



(c)



(d)



(e)

Figure 8: Predictor recovery as a function of switching and perturbation probabilities for $n = 10$ genes: (a) $N = 4000$, (b) $N = 6000$, (c) $N = 8000$, (d) $N = 10\,000$, (e) $N = 50\,000$.

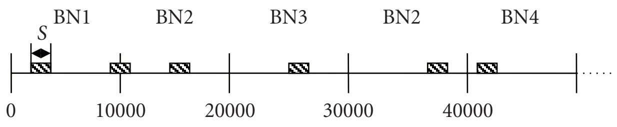


FIGURE 9: Subsampling strategy.

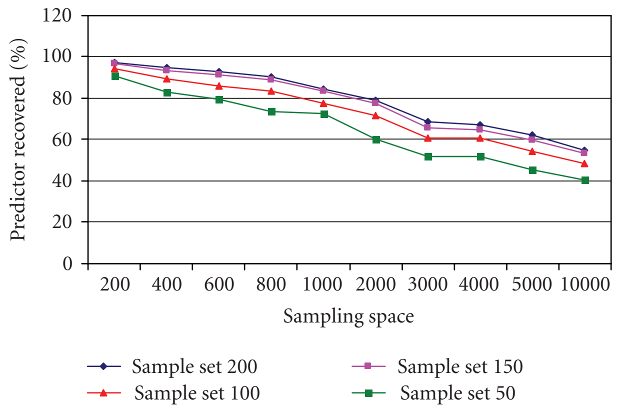


FIGURE 10: Predictor recovery percentages using various subsampling regimes.

$q = .001$, it is necessary on average to observe 1,000 points before a switch to the second BN occurs. This requires huge data lengths, not for deriving the parameters (predictors and functions) of the underlying model, but in order for a switch to occur in order to observe another BN.

This motivates consideration of subsampling. Rather than analyzing the full sequence, we analyze a small subsequence of data points, skip a large run of points, analyze another sample, skip more points, and so forth. If the sample is sufficiently long to classify it correctly, then the samples from the same BN may be collated to produce good parameter estimates. The subsampling strategy is illustrated in Figure 9. It is for use with data possessing a low switching probability. It is only necessary to see a sequence containing a small number of sample points of each BN in order to identify the BN. The length of the sampled subsequences is fixed at some value S.

To test the subsampling strategy, a set of 20 data sequences, each consisting of 100 000 samples points, was generated from a PBN consisting of 4 BNs, $n = 7$ variables, $k = 2$, $P = .01$, and $q = .001$ in database A. We define a sampling space to consist of a sampling window and non-sampling interval, so that the length of a sampling space is given by $L = S + I$, where I is the length of the non-sampling interval. We have considered sampling spaces of lengths $L = 200, 400, 600, 800, 1000, 2000, 3000, 4000, 5000$, and 10 000 and sampling windows (subsequences) of lengths $S = 50, 100, 150$, and 200. When $S = L$, there is no subsampling. The results are shown in Figure 10, which shows the percentage of predictors recovered. The recovery percentage by processing all 100 000 points in the full sequence is 97.28%.

Subsampling represents an effort at complexity reduction and is commonly used in engineering applications to gain speed and reduce cost. From a larger perspective, the entire investigation of gene regulatory networks needs to take complexity reduction into consideration because in the natural state the networks are extremely complex. The issue is whether goals can be accomplished better using fine- or coarse-grained analysis [28]. For instance, a stochastic differential equation model might provide a more complete description in principle, but a low-quantized discrete network might give better results owing to reduced inference requirements or computational complexity. Indeed, in this paper we have seen the inference difficulty that occurs by taking into account the stochasticity caused by latent variables on a coarse binary model. Not only does complexity reduction motivate the use of models possessing smaller numbers of critical parameters and relations, for instance, by network reduction [29] suppressing functional relations in favor of a straight transitional probabilistic model [30], it also motivates suboptimal inference, as in the case of the subsampling discussed herein or in the application of suboptimal intervention strategies to network models, such as PBNs [31].

7. REAL-DATA NETWORK EXPERIMENT

To test the inference technique on real data, we have considered an experiment based on a model affected by latent variables, these being a key reason for PBN modeling. Latent variables are variables outside the model whose behavior causes the model to appear random—switch between constituent networks.

The real-gene PBN is derived from the drosophila segment polarity genes for which a Boolean network has been derived that consists of 8 genes: wg_1, wg_2, wg_3, wg_4, PTC_1, PTC_2, PTC_3, and PTC_4 [26]. The genes are controlled by the following equations:

$$\begin{aligned}
wg_1 &= wg_1 \text{ and not } wg_2 \text{ and not } wg_4,\\
wg_2 &= wg_2 \text{ and not } wg_1 \text{ and not } wg_3,\\
wg_3 &= wg_1 \text{ or } wg_3,\\
wg_4 &= wg_2 \text{ or } wg_4,\\
PTC_1 &= (\text{not } wg_2 \text{ and not } wg_4) \text{ or}\\
&\qquad (PTC_1 \text{ and not } wg_1 \text{ and not } wg_3),\\
PTC_2 &= (\text{not } wg_1 \text{ and not } wg_3) \text{ or}\\
&\qquad (PTC_2 \text{ and not } wg_2 \text{ and not } wg_4),\\
PTC_3 &= 1, \qquad PTC_4 = 1.
\end{aligned} \tag{22}$$

Now let wg_4 and PTC_4 be hidden variables (not observable). Since PTC_4 has a constant value, its being a hidden variable has no effect on the network. However, if we let

Table 6: Percentages of the predictors and functions recovered from the segment polarity PBN.

Length	$q = .001$		$q = .005$		$q = .02$	
	Predictor recovered (%)	Function recovered (%)	Predictor recovered (%)	Function recovered (%)	Predictor recovered (%)	Function recovered (%)
2000	51.94	36.83	56.95	39.81	68.74	49.43
4000	58.39	38.49	59.86	44.53	70.26	52.86
6000	65.78	50.77	80.42	65.40	85.60	68.11
8000	72.74	59.23	83.97	69.47	86.82	70.28
10 000	76.03	63.98	88.10	74.31	92.80	77.83
20 000	87.81	76.86	95.68	81.60	96.98	83.48
30 000	97.35	84.61	97.65	88.28	99.17	88.82
40 000	98.64	85.74	99.19	90.03	99.66	91.05
50 000	99.59	90.18	99.59	90.35	99.79	91.94
100 000	99.69	90.85	99.87	91.19	100	93.97

$wg_4 = 0$ or 1, we will arrive at a 6-gene PBN consisting of two BNs. When $wg_4 = 0$, we have the following BN:

$$\begin{aligned} wg_1 &= wg_1 \text{ and not } wg_2, \\ wg_2 &= wg_2 \text{ and not } wg_1 \text{ and not } wg_3, \\ wg_3 &= wg_1 \text{ or } wg_3, \\ PTC_1 &= (\text{not } wg_2) \text{ or } (PTC_1 \text{ and not } wg_1 \text{ and not } wg_3), \\ PTC_2 &= (\text{not } wg_1 \text{ and not } wg_3) \text{ or } (PTC_2 \text{ and not } wg_2), \\ PTC_3 &= 1. \end{aligned} \tag{23}$$

When $wg_4 = 1$, we have the following BN:

$$\begin{aligned} wg_1 &= wg_1 \text{ and not } wg_2 \text{ and } 0, \\ wg_2 &= wg_2 \text{ and not } wg_1 \text{ and not } wg_3, \\ wg_3 &= wg_1 \text{ or } wg_3, \\ PTC_1 &= (\text{not } wg_2 \text{ and } 0) \text{ or} \\ &\qquad (PTC_1 \text{ and not } wg_1 \text{ and not } wg_3), \\ PTC_2 &= (\text{not } wg_1 \text{ and not } wg_3) \text{ or} \\ &\qquad (PTC_2 \text{ and not } wg_2 \text{ and } 0), \\ PTC_3 &= 1. \end{aligned} \tag{24}$$

Together, these compose a 6-gene PBN. Note that in the second BN we do not simplify the functions for wg_1, PTC_1, and PTC_3, so that they have the same predictors as in the first BN.

There are 6 genes considered here: wg_1, wg_2, wg_3, PTC_1, PTC_2, and PTC_3. The maximum number of predictor genes is $k = 4$. The two constituent networks are regulated by the same predictor sets. Based on this real-gene regulatory network, synthetic sequences have been generated and the inference procedure is applied to these sequences. 600 sequences with 10 different lengths (between 2000 and 100 000) have been generated with various lengths, $P = .01$, and three switching probabilities, $q = .001, .005, .02$. Table 6 shows the average percentages of the predictors and functions recovered.

8. CONCLUSION

Capturing the full dynamic behavior of probabilistic Boolean networks, whether they are binary or multivalued, will require the use of temporal data, and a goodly amount of it. This should not be surprising given the complexity of the model and the number of parameters, both transitional and static, that must be estimated. This paper proposed an algorithm that works well, but shows the data requirement. It also demonstrates that the data requirement is much smaller if one does not wish to infer the switching, perturbation, and selection probabilities, and that constituent-network connectivity can be discovered with decent accuracy for relatively small time-course sequences. The switching and perturbation probabilities are key factors, since if they are very small, then large amounts of time are needed to escape attractors; on the other hand, if they are large, estimation accuracy is hurt. Were we to restrict our goal to functional descriptions of state transitions when in attractor cycles, then the necessary amount of data would be enormously reduced; however, our goal in this paper is to capture as much of the PBN structure as possible, including transient regulation.

Among the implications of the issues raised in this paper, there is a clear message regarding the tradeoff between fine- and coarse-grain models. Even if we consider a binary PBN, which is considered to be a coarse-grain model, and a small number of genes, the added complexity of accounting for function switches owing to latent variables significantly increases the data requirement. This is the kind of complexity problem indicative of what one must confront when using solely data-driven learning algorithms. Further study should include mitigation of data requirements by prior knowledge, such as transcriptional knowledge of connectivity or regulatory functions for some genes involved in the network. It is also important to consider the reduction in complexity resulting from prior constraints on the network generating the data. These might include: connectivity, attractor structure, the effect of canalizing functions, and regulatory bias. In the other direction, one can consider complicating factors such as missing data and inference when data measurements cannot be placed into direct relation with the synchronous temporal dynamics of the model.

ACKNOWLEDGMENTS

We appreciate the National Science Foundation (CCF-0514644 and BES-0536679) and the National Cancer Institute (R01 CA-104620) for partly supporting this research. We would also like to thank Edward Suh, Jianping Hua, and James Lowey of the Translational Genomics Research Institute for providing high-performance computing support.

REFERENCES

[1] E. R. Dougherty, A. Datta, and C. Sima, "Research issues in genomic signal processing," *IEEE Signal Processing Magazine*, vol. 22, no. 6, pp. 46–68, 2005.

[2] T. Akutsu, S. Miyano, and S. Kuhara, "Identification of genetic networks from a small number of gene expression patterns under the Boolean network model," in *Proceedings of the 4th Pacific Symposium on Biocomputing (PSB '99)*, pp. 17–28, Mauna Lani, Hawaii, USA, January 1999.

[3] H. Lähdesmäki, I. Shmulevich, and O. Yli-Harja, "On learning gene regulatory networks under the Boolean network model," *Machine Learning*, vol. 52, no. 1-2, pp. 147–167, 2003.

[4] S. Liang, S. Fuhrman, and R. Somogyi, "REVEAL, a general reverse engineering algorithm for inference of genetic network architectures," in *Proceedings of the 3rd Pacific Symposium on Biocomputing (PSB '98)*, pp. 18–29, Maui, Hawaii, USA, January 1998.

[5] R. Pal, I. Ivanov, A. Datta, M. L. Bittner, and E. R. Dougherty, "Generating Boolean networks with a prescribed attractor structure," *Bioinformatics*, vol. 21, no. 21, pp. 4021–4025, 2005.

[6] X. Zhou, X. Wang, and E. R. Dougherty, "Construction of genomic networks using mutual-information clustering and reversible-jump Markov-chain-Monte-Carlo predictor design," *Signal Processing*, vol. 83, no. 4, pp. 745–761, 2003.

[7] R. F. Hashimoto, S. Kim, I. Shmulevich, W. Zhang, M. L. Bittner, and E. R. Dougherty, "Growing genetic regulatory networks from seed genes," *Bioinformatics*, vol. 20, no. 8, pp. 1241–1247, 2004.

[8] X. Zhou, X. Wang, R. Pal, I. Ivanov, M. L. Bittner, and E. R. Dougherty, "A Bayesian connectivity-based approach to constructing probabilistic gene regulatory networks," *Bioinformatics*, vol. 20, no. 17, pp. 2918–2927, 2004.

[9] E. R. Dougherty and Y. Xiao, "Design of probabilistic Boolean networks under the requirement of contextual data consistency," *IEEE Transactions on Signal Processing*, vol. 54, no. 9, pp. 3603–3613, 2006.

[10] D. Pe'er, A. Regev, G. Elidan, and N. Friedman, "Inferring subnetworks from perturbed expression profiles," *Bioinformatics*, vol. 17, supplement 1, pp. S215–S224, 2001.

[11] D. Husmeier, "Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks," *Bioinformatics*, vol. 19, no. 17, pp. 2271–2282, 2003.

[12] J. M. Peña, J. Björkegren, and J. Tegnér, "Growing Bayesian network models of gene networks from seed genes," *Bioinformatics*, vol. 21, supplement 2, pp. ii224–ii229, 2005.

[13] H. Lähdesmäki, S. Hautaniemi, I. Shmulevich, and O. Yli-Harja, "Relationships between probabilistic Boolean networks and dynamic Bayesian networks as models of gene regulatory networks," *Signal Processing*, vol. 86, no. 4, pp. 814–834, 2006.

[14] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *Journal of Theoretical Biology*, vol. 22, no. 3, pp. 437–467, 1969.

[15] S. A. Kauffman, "Homeostasis and differentiation in random genetic control networks," *Nature*, vol. 224, no. 5215, pp. 177–178, 1969.

[16] S. A. Kauffman, *The Origins of Order: Self-Organization and Selection in Evolution*, Oxford University Press, New York, NY, USA, 1993.

[17] S. Huang, "Gene expression profiling, genetic networks, and cellular states: an integrating concept for tumorigenesis and drug discovery," *Journal of Molecular Medicine*, vol. 77, no. 6, pp. 469–480, 1999.

[18] I. Shmulevich, E. R. Dougherty, S. Kim, and W. Zhang, "Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks," *Bioinformatics*, vol. 18, no. 2, pp. 261–274, 2002.

[19] S. Kim, H. Li, E. R. Dougherty, et al., "Can Markov chain models mimic biological regulation?" *Journal of Biological Systems*, vol. 10, no. 4, pp. 337–357, 2002.

[20] I. Shmulevich, E. R. Dougherty, and W. Zhang, "From Boolean to probabilistic Boolean networks as models of genetic regulatory networks," *Proceedings of the IEEE*, vol. 90, no. 11, pp. 1778–1792, 2002.

[21] I. Shmulevich and E. R. Dougherty, "Modeling genetic regulatory networks with probabilistic Boolean networks," in *Genomic Signal Processing and Statistics*, E. R. Dougherty, I. Shmulevich, J. Chen, and Z. J. Wang, Eds., EURASIP Book Series on Signal Processing and Communication, pp. 241–279, Hindawi, New York, NY, USA, 2005.

[22] A. Datta, A. Choudhary, M. L. Bittner, and E. R. Dougherty, "External control in Markovian genetic regulatory networks," *Machine Learning*, vol. 52, no. 1-2, pp. 169–191, 2003.

[23] R. Pal, A. Datta, M. L. Bittner, and E. R. Dougherty, "Intervention in context-sensitive probabilistic Boolean networks," *Bioinformatics*, vol. 21, no. 7, pp. 1211–1218, 2005.

[24] R. Pal, A. Datta, and E. R. Dougherty, "Optimal infinite-horizon control for probabilistic Boolean networks," *IEEE Transactions on Signal Processing*, vol. 54, no. 6, part 2, pp. 2375–2387, 2006.

[25] A. Datta, R. Pal, and E. R. Dougherty, "Intervention in probabilistic gene regulatory networks," *Current Bioinformatics*, vol. 1, no. 2, pp. 167–184, 2006.

[26] R. Albert and H. G. Othmer, "The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in *Drosophila melanogaster*," *Journal of Theoretical Biology*, vol. 223, no. 1, pp. 1–18, 2003.

[27] G. Langholz, A. Kandel, and J. L. Mott, *Foundations of Digital Logic Design*, World Scientific, River Edge, NJ, USA, 1998.

[28] I. Ivanov and E. R. Dougherty, "Modeling genetic regulatory networks: continuous or discrete?" *Journal of Biological Systems*, vol. 14, no. 2, pp. 219–229, 2006.

[29] I. Ivanov and E. R. Dougherty, "Reduction mappings between probabilistic Boolean networks," *EURASIP Journal on Applied Signal Processing*, vol. 2004, no. 1, pp. 125–131, 2004.

[30] W.-K. Ching, M. K. Ng, E. S. Fung, and T. Akutsu, "On construction of stochastic genetic networks based on gene expression sequences," *International Journal of Neural Systems*, vol. 15, no. 4, pp. 297–310, 2005.

[31] M. K. Ng, S.-Q. Zhang, W.-K. Ching, and T. Akutsu, "A control model for Markovian genetic regulatory networks," in *Transactions on Computational Systems Biology V*, Lecture Notes in Computer Science, pp. 36–48, 2006.