

Systems biology

Generating Boolean networks with a prescribed attractor structure

Ranadip Pal¹, Ivan Ivanov¹, Aniruddha Datta¹, Michael L. Bittner² and Edward R. Dougherty^{1,2,*}¹Department of Electrical Engineering, Texas A&M University, College Station, TX 77843, USA and ²Translational Genomics Research Institute, 400 North Fifth Street, Suite 1600, Phoenix, AZ 85004, USAReceived on March 5, 2005; revised on August 9, 2005; accepted on September 5, 2005
Advance Access publication September 8, 2005

ABSTRACT

Motivation: Dynamical modeling of gene regulation via network models constitutes a key problem for genomics. The long-run characteristics of a dynamical system are critical and their determination is a primary aspect of system analysis. In the other direction, system synthesis involves constructing a network possessing a given set of properties. This constitutes the inverse problem. Generally, the inverse problem is ill-posed, meaning there will be many networks, or perhaps none, possessing the desired properties. Relative to long-run behavior, we may wish to construct networks possessing a desirable steady-state distribution. This paper addresses the long-run inverse problem pertaining to Boolean networks (BNs).

Results: The long-run behavior of a BN is characterized by its attractors. The rest of the state transition diagram is partitioned into level sets, the j -th level set being composed of all states that transition to one of the attractor states in exactly j transitions. We present two algorithms for the attractor inverse problem. The attractors are specified, and the sizes of the predictor sets and the number of levels are constrained. Algorithm complexity and performance are analyzed. The algorithmic solutions have immediate application. Under the assumption that sampling is from the steady state, a basic criterion for checking the validity of a designed network is that there should be concordance between the attractor states of the model and the data states. This criterion can be used to test a design algorithm: randomly select a set of states to be used as data states; generate a BN possessing the selected states as attractors, perhaps with some added requirements such as constraints on the number of predictors and the level structure; apply the design algorithm; and check the concordance between the attractor states of the designed network and the data states.

Availability: The software and supplementary material is available at <http://gsp.tamu.edu/Publications/BNs/bn.htm>

Contact: edward@ee.tamu.edu

INTRODUCTION

The dynamical modeling of gene regulation via network models constitutes a basic problem for genomics. As with any dynamical system, long-run characteristics are critical, and determining these is a primary aspect of system analysis. In the other direction is system synthesis: construct a network possessing a given set of

properties. This inverse *problem* is generally ill-posed, meaning there will be many networks, or perhaps none, having the desired properties. Relative to long-run behavior, one may wish to construct networks possessing a desirable stationary distribution. We consider a long-run inverse problem pertaining to Boolean networks (BNs).

Boolean networks compose a class of discrete models where the expression levels of each gene are assumed to have two possible values: ON or OFF (Kauffman, 1969). Such a model cannot capture the underlying continuous and stochastic biochemical nature of protein production and gene regulation; however, one often encounters genes that are essentially ON or OFF throughout a given biochemical pathway. The switch-like regulatory function of these genes determines their role in regulation, and this activity is well represented by a coarse-grain model like a BN. This, together with the relative simplicity of the dynamical system described by a BN, explains why such networks have attracted significant attention from the research community—for instance (Kauffman, 1993; Huang, 1999; Shmulevich *et al.*, 2002a). In practice, BN need to be designed (inferred) from data, and a significant amount of effort has gone into their design (Akutsu *et al.*, 1999; Ideker *et al.*, 2000; Maki *et al.*, 2001; Shmulevich *et al.*, 2002b; Lähdesmäki *et al.*, 2003). Inference from data constitutes an inverse problem in that a network is constructed relative to some relationship with the sample data.

A Boolean network (BN) $B = (V, F)$ on n genes is defined by a set of nodes/genes $V = \{x_1, \dots, x_n\}$, $x_i \in \{0,1\}$, $i = 1, \dots, n$, and a vector of Boolean functions, $F = (f_1, \dots, f_n)$, $f_i : \{0,1\}^n \rightarrow \{0,1\}$, $i = 1, \dots, n$. Each node x_i represents the state/expression of the gene x_i , where $x_i = 0$ means that the gene is OFF and $x_i = 1$ means that the gene is ON. The function f_i is the predictor function for gene x_i . Updating the states of all of the genes in B is done synchronously at every time step according to their predictor functions. A subset $W_i \subseteq V$ is called the predictor set for the gene x_i if the restriction $f_i|_{W_i}$ of the predictor function f_i equals f_i . The cardinality of the set W_i is related to the number of edges incident with the vertex x_i in the directed graph $\Gamma = (V, E)$, where an edge $(x_i, x_j) \in E$ indicates that gene x_i is one of the factors determining the value of the gene x_j . $W = (W_1, \dots, W_n)$ is called the predictor set for the BN. A state in B is a vector (x_1, \dots, x_n) of gene values. We assume that the states of B are interpreted as binary numbers and are ordered accordingly. Thus, there are $N = 2^n$ states in a BN on n genes, and they are enumerated as $0, 1, 2, \dots, N-1$. There is a $N \times n$ truth table associated with and equivalent to B , where the rows correspond

*To whom correspondence should be addressed.

to the states in B and the columns contain the corresponding values for the predictor functions. The truth table of B induces a directed graph $\tilde{\Gamma} = (\tilde{V}, \tilde{E})$ with the states in B as the set \tilde{V} of its vertices, and with edges $(s_i, s_j) \in \tilde{E}$ connecting the state s_i with the state s_j if $F(s_i) = s_j$. It is clear that the truth table associated with B determines $\tilde{\Gamma}$ and vice versa. $\tilde{\Gamma}$ is called the state transition diagram of B , and $\tilde{\Gamma}$ is called compatible with W if the truth table induced by $\tilde{\Gamma}$ has W as the predictor set for the BN associated with that truth table. The state transition diagram represents the dynamics of the network.

Given an initial state, the network will eventually enter a set of states through which it will repeatedly cycle forever. Each such set is called an attractor cycle and states within attractor cycles are called attractors. The state transition diagram is partitioned into level sets, where level set l_j is composed of all states that transition to one of the attractor states in exactly j transitions, with attractors composing level set l_0 . Non-attractor states are the transient states of the network. These are partitioned according to the attractor cycles because each transient state begins a sequence of transitions that eventually ends up in a unique attractor cycle. The attractor cycles are mutually disjoint. The partition class corresponding to an attractor cycle is called the basin of the cycle. Given any transient state, it belongs to a unique basin and unique level.

The attractor inverse problem, which involves constructing BNs possessing a given attractor structure, is important to network inference from steady-state data. Most microarray-based gene-expression studies do not involve controlled temporal experimental data; rather, it is assumed that the data result from sampling from the steady state (see Biological Considerations in the Supplementary material). Under the BN model, this means that the data come from the attractors. If one considers a more general Boolean-type model, such as a BN with random perturbations or a Probabilistic Boolean Network (PBN), then the dynamical system represented by the network is an ergodic Markov chain and there exists a steady-state distribution; nevertheless, under mild stability assumptions that reflect biological state stability, most of the steady-state probability mass is concentrated in the attractors and it is expected that most data correspondingly come from the attractors (Brun *et al.*, 2005).

Assuming that we are sampling from the steady state, a key criterion for checking the validity of a designed network is that much of its steady-state mass lies in the observed sample states (Kim *et al.*, 2002). For BNs, this means that there should be concordance between the model's attractor states and the data states. Such a criterion can be used to test a design algorithm (Zhou *et al.*, 2004): randomly select a set of states to be used as data states; generate a BN possessing the selected states as attractors, perhaps with some added requirements such as constraints on connectivity and the level structure; apply the design algorithm; and check the concordance between the attractor states of the designed network and the states used as data. This can be done repeatedly for different data states and constraints. The algorithms provided in this paper can be used to generate the BNs in this scenario, and in fact have been used in this way in Zhou *et al.* (2004) (absent description or analysis of the algorithms). Owing to the concentration of mass in the attractors of PBNs and the fact that a PBN can be viewed as a collection of BNs on the same set of genes and with a certain probability q of switching between different BNs plus a certain probability p of flipping the value of any one of the participating

genes, the aforementioned procedure can be applied to PBNs by generating the constituent BNs.

The inverse problem, attractors to network, is a one-to-many mapping, and there may be a multitude of networks possessing a given attractor structure. In the other direction, if the problem is constrained, say by the number of predictors permitted, there may be no solution. Generally speaking, steady-state behavior restricts the dynamical behavior, but does not determine it. In particular, for Boolean networks it does not determine the basin structure. Thus, while we might obtain good inference regarding the attractors, we may obtain poor inference relative to their probabilities relative to random initializations (or to random perturbations in more general networks). This is because if the basin of an attractor is small, it is less likely to catch a random initialization than if it is large. When sampling from the steady state, the attractors with small basins are less likely to appear (and may not appear at all), whereas those with large basins may appear numerous times. A key advantage of checking a design algorithm with generated synthetic networks is that the levels and basins of a synthetic network are known and therefore one can better evaluate algorithm performance.

In this paper we present two algorithms for solving the attractor inverse problem. The problem can be formulated in the following manner. Given a set V consisting of n nodes (genes), a family of n subsets W_1, W_2, \dots, W_n of V with cardinalities not less than m and not larger than M , $0 < m \leq M$, a set A containing k states, and two positive integers $l \leq L$, accordingly construct a BN with node set V , having predictor set $W = (W_1, W_2, \dots, W_n)$, possessing attractor cycles A_1, A_2, \dots, A_r , where $A = \cup_{j=1}^r A_j$, and containing between l and L level sets. The requirement on the predictor set means that the state transition diagram of the designed network must be compatible with W . There may exist none or many compatible networks. The algorithms are typically initiated by specifying a minimum and maximum number of predictors for each gene, randomly selecting W_1, W_2, \dots, W_n subject to the specified maximum and minimum, and randomly selecting attractor cycles of given sizes. In this way, one can utilize the algorithms to search for BNs constrained by the connectivity of the network. For instance, if $|W_i| \leq M$ for $i = 1, 2, \dots, n$, then the algorithms find networks with connectivity bounded by M —if any exist with the required attractor structure.

In the case of a BN with only singleton attractors, the problem can be reformulated as a search problem in the following way: In the space of all k -forests $\tilde{\Gamma}$ for a BN on n genes, and with the number of their level sets ranging between l and L , find at least one which is compatible with a given (randomly generated) predictor set W , where the cardinality of each W_i ranges between m and M .

SYSTEMS AND METHODS

Under the assumption that we are sampling from the steady state, biological state stability means that most of the steady-state probability mass is concentrated in the attractors and that real-world attractors are most likely to be singleton attractor cycles consisting of a single state. A state transition diagram constitutes a single-rooted tree if it possesses exactly one singleton attractor (a single-state attractor cycle): the network reaches its attractor via the tree. If it possesses k singleton attractors, then it is composed of k single-rooted trees and is called a k -forest: the network reaches an attractor state via one of the single-rooted trees. In this section we examine the size of the search state under the assumption of singleton attractors. To simplify the formulas, it is assumed that $m = 1$ and $l = 1$. One can easily make the necessary changes in the general case.

There are $A = A_{n,M} = \sum_{i=1}^M \binom{n}{i}$ possible predictor sets W_i for each gene x_i , $i = 1, \dots, n$. Thus, there are A^n possible predictor sets W to select from when searching for a compatible state transition diagram $\tilde{\Gamma}$. The different choices for $\tilde{\Gamma}$ depend on the number of attractor states and on the level set structure of the state transition diagram. There are $\binom{N}{k}$ possible ways of selecting the k singleton attractors for $\tilde{\Gamma}$. The remaining $N_k = N - k$ non-attractor states will form the level sets of $\tilde{\Gamma}$. There are $n^{(l_{i+1})n^{(l_i)}}$ ways for connecting any two successive level sets l_i and l_{i+1} with $n^{(l_i)}$ and $n^{(l_{i+1})}$ states in them, respectively. Therefore, the number of possible ways to structure the level sets for $\tilde{\Gamma}$ with no more than L level sets is

$$\begin{aligned} \Lambda &= \Lambda_{L,k,n} \\ &= \sum_{i=1}^L \sum_{\substack{n(l_1) \dots n(l_i) \\ n(l_1) + \dots + n(l_i) = N_k}} \frac{N_k!}{n(l_1)! \dots n(l_i)!} k^{n(l_1)} n^{(l_2)} \dots n^{(l_{i-1})} n^{(l_i)}, \end{aligned} \quad (1)$$

where \sum_i is a summation over all of the different choices of the positive integers $n(l_1), \dots, n(l_i)$ such that $\sum_{j=1}^i n(l_j) = N_k$. Combining this with the choices for selecting attractor states, and with the choices for selecting predictor sets, yields the size of the search space,

$$S = S_{n,m,k} = A^n \binom{N}{k} \Lambda. \quad (2)$$

To appreciate the size of the search space, consider an example of a very small BN, where $n = 4$, $m = 4$, $k = 4$, and the state transition diagram has exactly 4 levels. The computations show that $S \geq 10^{17}$.

The following theorem extends a well-known result (Deo, 1974), about 1-trees, or single-rooted trees. Its proof is given in the Supplementary material, along with examples to illustrate the procedure used in the derivation.

THEOREM 1. *The cardinality of the collection of all forests on N vertices is $(N + 1)^{N-1}$.*

No restrictions are imposed on the structure of the level sets. Consequently the theorem provides us with an upper bound for the term $\binom{N}{k} \Lambda$ appearing in Equation (2). While this upper bound is by no means tight, it is much tighter in comparison to the number of all possible directed graphs on N vertices, N^N . One can easily see that the ratio $(N + 1)^{N-1}/N^N$ is asymptotically equal to e/N . Since $N = 2^n$, the probability mass decreases exponentially relative to the number of genes n . This shows that a brute force search for an acceptable BN by randomly filling in a BN truth table has very little chance of success. Therefore, if one wants to efficiently generate a BN with the desired characteristics, one has to incorporate information from the state transition diagram, as well as the information about the predictor set of the network, into the algorithm.

We present two algorithms to generate BNs given attractor and connectivity information. The first algorithm works directly with the truth table, incorporating at the same time the information about the attractor set, as well as the information about the predictor set of the BN. There is no control over the level set structure, and the transition diagram generated by the algorithm has to be checked for the presence of cycles. We present the algorithms for the case of singleton attractors and provide the adaptation for multiple-state cyclic attractors in the Supplementary material.

Algorithm 1

Step 1: Randomly generate a set of k attractor states. If Step 1 has been repeated more than a pre-specified number of times terminate the algorithm.

Step 2: Randomly pick up a predictor set W , where each W_i has not less than m and not more than M elements. If Step 2 has been repeated more than a pre-specified number of times go back to Step 1.

Step 3: Check if the selected attractor set is compatible with W , i.e. only the attractor set of the state transition diagram is checked for compatibility

against W . If the attractor set is not compatible with W go back to Step 2, otherwise continue to Step 4.

Step 4: Fill in the entries of the truth table that correspond to the attractors generated in Step 1. Using the predictor set W randomly fill in the remaining entries of the truth table. If Step 4 has been repeated more than a pre-specified number of times go back to Step 2.

Step 5: Search for cycles of any length in the state transition diagram $\tilde{\Gamma}$ that is associated with the truth table generated in Step 4. If a cycle is found go back to Step 4, otherwise continue to Step 6.

Step 6: If $\tilde{\Gamma}$ has less than l or more than L level sets go back to Step 4, otherwise continue to Step 7.

Step 7: Save the generated BN and terminate the algorithm.

The second algorithm employs a state transition diagram $\tilde{\Gamma}$ that satisfies the design goals about attractor structure and level-set structure, and checks if the truth table associated with $\tilde{\Gamma}$ has a predictor set W satisfying the design goals.

Algorithm 2

Step 1: Randomly generate a state transition diagram $\tilde{\Gamma}$ that satisfies the design goals about the attractor structure and level set structure. If Step 1 has been repeated more than a pre-specified number of times terminate the algorithm.

Step 2: Fill in the truth table using $\tilde{\Gamma}$.

Step 3: If there is at least one W_i in the predictor set W given by the truth table that has less than m or more than M elements go back to Step 1, otherwise continue to Step 4.

Step 4: Save the generated BN and terminate the algorithm.

IMPLEMENTATION

The algorithms have been implemented in Matlab and C, the latter being far more efficient. Moreover, while this paper has focused on binary-valued networks, there is nothing inherently binary in the algorithms and they can be directly applied to more finely quantized networks, albeit, at the cost of much larger solution spaces.

A performance comparison (see Supplementary material) shows the better efficiency of Algorithm 1. The reason is that the state transition diagrams generated by Algorithm 1 have a very small probability mass in the space of all k -forests, $k = 1, \dots, N$ on N vertices. When each gene predictor set W_i is required to have exactly m elements, the number of possible state transition diagrams generated by Algorithm 1 is $\binom{n}{m}^n N^{2^m}$. Using Theorem 1, one can obtain an estimate of the probability mass of the state transition diagrams generated by Algorithm 1 within the space of all k -forests, $k = 1, \dots, N$: $\binom{n}{m}^n N^{2^m} / (N + 1)^{N-1}$. For the case $n = 6$, $m = 5$, this ratio is $\sim 1.7911 \times 10^{-52}$.

The following example provides a walk-through illustration to show how Algorithm 1 works in the particular case of 3 genes. An example to illustrate algorithm 2 is given in the Supplementary material.

EXAMPLE 3.1. (ALGORITHM 1) Step 1: Suppose that $k = 2$, $m = 1$, $M = 2$, $l = 1$ and $L = 5$. Next, suppose that the states 000 and 011 are generated by Step 1.

Step 2: Suppose W is generated where $W_1 = \{x_2, x_3\}$, $W_2 = \{x_1, x_3\}$, $W_3 = \{x_1, x_2\}$.

Step 3: Table 1 shows that the attractors generated in Step 1 are compatible with W . The remaining entries a_1, \dots, a_6 in the truth table are filled in randomly in the next step of the algorithm. One can

Table 1. Truth table for example 3.1

$x_1 \ x_2 \ x_3$	f_1	f_2	f_3
0 0 0	0	0	0
0 0 1	a_1	1	0
0 1 0	a_2	0	1
0 1 1	0	1	1
1 0 0	0	a_3	a_4
1 0 1	a_1	a_5	a_4
1 1 0	a_2	a_3	a_6
1 1 1	0	a_5	a_6

notice certain patterns in the entries in each one of the three columns of the table. These reflect the structure of the predictor set W , and reduce the number of the possible ways to randomly fill in the missing entries during the next step of the algorithm.

On the one hand, if the attractors generated in Step 1 were 000 and 100, then for the predictor function of the first gene x_1 , we must have $f_1(0,0) = 0$, while from the second attractor, we get $f_1(0,0) = 1$, which is a contradiction. Therefore that attractor set is not compatible with the set W generated in Step 2.

Step 4: Here we randomly fill in the remaining entries of the truth table. Suppose that this produces $a_1 = 0, a_2 = 0, a_3 = 1, a_4 = 0, a_5 = 0, a_6 = 1$. This selection produces the following transitions in the state transition diagram: $0 \rightarrow 0; 1 \rightarrow 2; 2 \rightarrow 1; 3 \rightarrow 3; 4 \rightarrow 2; 5 \rightarrow 0; 6 \rightarrow 3; 7 \rightarrow 1$, where we have used the decimal representation of the states. It is clear that during Step 5 of the algorithm the cycle $1 \rightarrow 2; 2 \rightarrow 1$ will be discovered, which will cause the BN generated by the present truth table to be discarded, and we will be returned to Step 4.

On the other hand if we had $a_1 = 0, a_2 = 1, a_3 = 1, a_4 = 0, a_5 = 0, a_6 = 1$ produced in Step 4, then the transitions in the state transition diagram would be $0 \rightarrow 0; 1 \rightarrow 2; 2 \rightarrow 5; 3 \rightarrow 3; 4 \rightarrow 2; 5 \rightarrow 0; 6 \rightarrow 7; 7 \rightarrow 1$. Since the only cycles here are those within the attractor set, Step 5 of the algorithm will take us to Step 6. Step 6 will detect that there are 5 level sets, and this will take us to Step 7.

DISCUSSION

The algorithms produce many distinct networks satisfying a given set of constraints. The presence of multiple solutions allows for optimization procedures when designing PBNs from microarray data. In this section, we describe a procedure for designing a PBN from microarray data. The sizes of the basins are used to select BNs from a group of generated networks and to combine them in a PBN whose steady-state distribution closely matches the observed frequency distribution of the data. The assumption that these data correspond to the steady state of the underlying gene regulatory system provides a reference point of how closely the dynamics of a generated PBN approximate the data. The designed PBN should have these data points as attractors—and no other attractors because there is no reason in the data for having other attractors. We focus on singleton attractors.

The design procedure begins by selecting a random number N_1 between 2 and 5, and then randomly selecting N_1 distinct states as singleton attractors from the original data according to the data frequency distribution. Repeating this procedure 10 times yields 10 sets, A_1, A_2, \dots, A_{10} , of singleton attractors, with A_i possessing N_i attractors, $2 \leq N_i \leq 5$ and $i = 1, 2, \dots, 10$. After that, Algorithm 1

is employed to generate 100 BNs, $\mathcal{B}_{i1}, \mathcal{B}_{i2}, \dots, \mathcal{B}_{i100}$, for each of the 10 attractor sets. The generated networks have state transition diagrams satisfying two additional constraints. First, the number of their level sets range from 2 to 10. This constraint manifests the understanding that in the underlying gene regulatory network the steady-state/fixed points of the system are not achieved with too few or too many consecutive transitions. The second constraint is that all gene predictor sets have between 1 and 3 genes, the number being randomly set.

The BNs generated for each one of the 10 attractor sets are then used as a sample space for the selection of a PBN whose steady-state distribution matches closely the frequency distribution of the data in the mean-square error (MSE) sense. One BN from each group of 100 BNs is randomly selected, and the basin size of each singleton attractor is calculated. These numbers are used as estimates of the steady-state probabilities of the corresponding attractors (keeping in mind that very little time is spent in transient states and that random perturbations and switching randomly put the network in different basins). For example, if the BN \mathcal{B}_{ij} has attractor states $\mathbf{a}_1^{ij}, \mathbf{a}_2^{ij}, \dots, \mathbf{a}_{N_{ij}}^{ij}$ with corresponding basin sizes $S_1^{ij}, S_2^{ij}, \dots, S_{N_{ij}}^{ij}$, respectively, then our estimate of the steady-state probability for the attractor \mathbf{a}_i^{ij} is given by $\pi_{ij}(\mathbf{a}_i^{ij}) = S_i^{ij} / \sum_{k=1}^{N_{ij}} S_k^{ij}$. One can take the average of these estimates of the steady-state probabilities of the singleton attractor \mathbf{a}_i^{ij} over the 10 BNs comprising the PBN as an estimate of the steady-state probability of that attractor in the PBN. If a particular singleton attractor is not present in a constituent BN, then its contribution to the steady-state probability is set to 0.

Continuing in this fashion, one obtains an estimate of the steady-state probabilities of each one of the data states used in the generation of a PBN. Let us denote these states by $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_m$ and their corresponding estimated steady-state probabilities by $\pi_1, \pi_2, \dots, \pi_m$. The procedure calculates the MSE between $\pi_1, \pi_2, \dots, \pi_m$ and f_1, f_2, \dots, f_m , and where f_i is the relative frequency of \mathbf{b}_i in the sample. The designed PBN is selected as the one that minimizes the MSE among a randomly selected subset of 10 000 PBNs from the set of all possible PBNs that can be generated using the BNs produced by Algorithm 1 for the selected attractor sets. We settle on 10 000 PBNs because an exhaustive search is prohibitive, there being a total of 100^{10} possible PBNs.

We have applied the preceding PBN design procedure using gene-expression profiles from a study of 31 malignant melanoma samples in which messenger RNA was isolated directly from melanoma biopsies, and fluorescent cDNA from the message was prepared and hybridized to a microarray containing probes for 8150 cDNAs (representing 6971 unique genes) (Bittner *et al.*, 2000). The 7 genes WNT5A, pirin, S100P, RET1, MART1, HADHB and STC2 used here for the model were chosen from a set of 587 genes from the dataset that have been subjected to an analysis of their ability to cross predict each other's state in a multivariate setting (Kim *et al.*, 2002). The gene-expression profiles were binarized to arrive at 31 binary vectors with 7 columns corresponding to the selected 7 genes. The frequency distribution of the 18 distinct binary data vectors is shown in Figure 1. The assumption that these data correspond to the steady state of the underlying gene regulatory system implies that those and only those 18 distinct data vectors should appear as attractors in the generated PBN. This condition is guaranteed by the design procedure.

Figure 1 shows the portion of histogram (the data states only) of the steady-state distribution (after a long run) of the designed PBN,

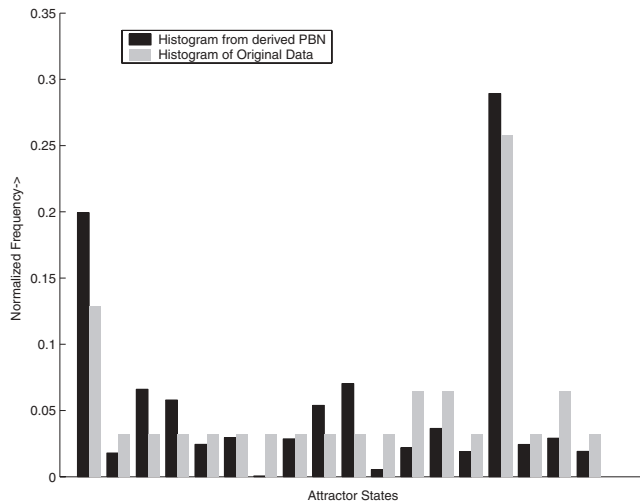


Fig. 1. Histogram for original and generated PBN showing only attractor states.

with $q = 0.001$ and $p = 0.001$, and of the frequency distribution of the data states. The steady-state distribution of the PBN closely matches (in the MSE sense) the frequency distribution observed in the data.

In conclusion, let us remark that from a general perspective we have applied to BN design the pattern-recognition principle of constraining the solution space when making inferences from limited data. This has been accomplished by making assumptions on the dynamical structure of the network, assumptions that can be made in accordance with biological understanding. Since the algorithms generate networks in the constrained solution space, they can be used to provide synthetic networks to test proposed inference algorithms, for both BNs and PBNs.

SUPPLEMENTARY DATA

For supplementary data please refer to *Bioinformatics* online.

ACKNOWLEDGEMENTS

This research was supported by the National Science Foundation (ECS0355227 and CCF0514644) and the National Cancer Institute (CA90301). We would like to extend our appreciation to Ronaldo Hashimoto of the University of Sao Paulo for his careful reading of the manuscript and helpful insights.

Conflict of Interest: none declared.

REFERENCES

- Akutsu, T. *et al.* (1999) Identification of genetic networks from a small number of gene expression patterns under the Boolean network model. *Pac. Symp. Biocomp.*, **4**, 17–28.
- Bittner, M. *et al.* (2000) Molecular classification of cutaneous malignant melanoma by gene expression profiling. *Nature*, **406**, 536–540.
- Brun, M. *et al.* (2005) Steady-state probabilities for attractors in probabilistic Boolean networks. *Signal Processing*, **85**, 1993–2013.
- Deo, N. (1974) *Graph Theory with Applications to Engineering and Computer Science*. Englewood Cliffs, Prentice-Hall, NJ.
- Huang, S. (1999) Gene expression profiling, genetic networks, and cellular states: an integrating concept for tumorigenesis and drug discovery. *Mol. Med.*, **77**, 469–480.
- Ideker, T. E. *et al.* (2000) Discovery of regulatory interactions through perturbation: inference and experimental design. *Pac. Symp. Biocomput.*, **5**, 305–316.
- Kauffman, S. A. (1969) Metabolic stability and epigenesis in randomly constructed genetic nets. *Theor. Biol.*, **22**, 437–467.
- Kauffman, S. A. (1993) *The Origins Of Order: Self-organization and Selection in Evolution*. Oxford University Press, NY.
- Kim, S. *et al.* (2002) Can Markov chain Models mimic biological regulation? *Biol. Syst.*, **10**, 337–357.
- Lähdesmäki, H. *et al.* (2003) On learning gene regulatory networks under the Boolean model. *Machine Learning*, **52**, 147–167.
- Maki, Y. *et al.* (2001) Development of a system for the inference of large scale genetic networks. *Pac. Symp. Biocomput.*, **6**, 446–458.
- Shmulevich, I. *et al.* (2002a) From Boolean to probabilistic Boolean networks as models of genetic regulatory networks. *Proc. IEEE*, **90**, 1778–1792.
- Shmulevich, I. *et al.* (2002b) Inference of genetic regulatory networks via best-fit extensions. In Zhang, W. and Shmulevich, I. (eds), *Computational and Statistical Approaches to Genomics*. Kluwer Academic Publishers, Boston, MA.
- Zhou, X. *et al.* (2004) A Bayesian connectivity-based approach to constructing probabilistic gene regulatory networks. *Bioinformatics*, **20**, 2918–2927.