

# Supplementary Material for the paper Generating Boolean Networks With a Prescribed Attractor Structure

Ranadip Pal , Ivan Ivanov , Aniruddha Datta ,Michael L. Bittner , and Edward R. Dougherty.

## 1 Biological Considerations

The use of steady state data as a starting point for functional and relational inferences in biology is commonplace. Most of what is known in biology in terms of the association of genes with particular functions or phenotypes, and in terms of the connections between genes that collaborate to produce particular functions or phenotypes, has been inferred from steady state data. One of the most productive fields of inference, biochemical genetics, has used massive mutational screens to establish associations between particular genes and particular metabolic processes producing subunits, such as synthesis of an amino acid or between particular genes and more complex processes, such as the replication of a genome. The strategy used in these projects has usually been to produce a large set of cells, each having approximately one mutation (randomly placed) in its genome and then allowing the mutated cells to grow under conditions that shelter the organism from the defect of the mutation. In metabolite studies, the mutants would be propagated on a rich media that supplies many different essential metabolites, therefore allowing cells harboring mutations inactivating the synthesis of these metabolites to grow. The defect is then revealed by transferring arrays of representatives of the pool of mutated cells to a minimal media and noting which cell types died. The mutants could then be further tested by transfer of a representative array of the mutated cells to media containing more and more highly specified sets of supplemental metabolites until one could specify that a particular mutant is associated with a particular chemical conversion. Similar strategies with somewhat more involved technical approaches have been equally useful in examining multicellular, diploid organisms, allowing a surprisingly large amount of knowledge to be gained about such complex processes as gene involvement in development simply by scoring the final phenotypic state of the developed organism bearing the mutated regulatory gene.

The current ability to examine the kinds of alterations in gene structure, copy number, expression level, protein level and protein modification as cellular phenotypes associated with diseases such as cancer provides a new opportunity to exploit the same kinds of strategies that have been previously used, but on a much larger, multivariate scale. The basic strategy remains the same, associate particular phenotypes with particular alterations. Using any single measurement of cell state, one would expect to identify a set of genes that are consistently altered, in a way detectable by the measurement, in samples arising from a particular molecular pathology. This raises the question of how many variants of a pathologic process are likely to be represented in a series of samples taken from a disease such as cancer arising in a given tissue. While it is well known that there are many ways to assemble altered regulatory networks that would lead to activation of a cellular process such as chronic proliferation, it is also known that cancers arising from a particular tissue of origin, such as breast, tend mainly to utilize a small subset of them, and that this tendency may be shaped by the particular cellular subtype within the tissue of origin (Nielsen *et al.*). It stands to reason that the acquisition of altered regulatory networks that will provide activation or repression of particular cellular functions will be strongly influenced by the number of alterations required to achieve the functional goal, and thus that the majority of altered regulations observed for a given disease type will be represent only a few classes, each with distinctive components.

Therefore, in the case of expression profiling, the expectation is that some set of genes will behave with varying degrees of consistency at an observable rate in samples where that particular molecular pathology is operating, and that this pattern of consistency will not be observed in samples where that process is not operating. By examining the capabilities of the genes in the set to predict each other's behavior, a number of sparse models of the network of molecular interactions supporting the phenotype being studied can be produced. These coarse network representations clearly do not represent detailed views of the molecular mechanisms yet they do represent something of great value to the biologists studying the problem, a focused list of some genes that are likely to be involved in producing or maintaining a particular phenotype in a given set of samples, and a ranking of the likely degree of coupling between these members. Given knowledge

of genes in such representations of the network, biologists will be able to rapidly assess whether the genes predicted to be involved are likely candidates, given the phenotype under consideration. The hypothesized network will also provide a coarse model that can rapidly be tested with typical methods of perturbing the expression level of critical genes to test the connection of the proposed network to the phenotype. This process represents a much more rapid way of identifying connections between genes and between genes and phenotype than the single mutation approach to functional identification on normal tissue, since one identifies sets of genes involved in a phenotype. As the network is already a partially refined mechanistic hypothesis, it also allows for more rapid testing and validation of its explanatory power. By way of example, a less refined version of the method presented here has allowed the identification and rapid validation of the signaling molecule, Wnt5a, as an elicitor of a very complex invasive phenotype in melanoma (Bittner *et al.*, Kim *et al.*, Weeraratna *et al.*)

## 2 Proof of the Theorem

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

**Proof:** In the proof we can assume without loss of generality that the vertices of each  $k$ -forest are labeled using the integers from  $\{1, 2, \dots, N\}$ . First we prove that there is a bijection between the collection  $\mathcal{F}_k$  of all  $k$ -forests,  $k$ -a fixed nonnegative integer less than  $N$ , and the collection  $\mathcal{B}_k$  of triples  $(\omega_k, A_k, r)$ ,  $r \in A_k$ , where the set  $A_k \in \mathcal{A}_k$ -the collection of all  $k$  element subsets of  $\{1, \dots, N\}$ , and  $\omega_k \in \Omega_{N-k}$ -the collection of all sequences of length  $N - k - 1$  integers formed using the integers  $i \in \{1, \dots, N\}$ . Define the mapping

$$\lambda : \mathcal{F}_k \rightarrow \Omega_{N-k} \times \mathcal{A}_k \times \{1, \dots, N\}$$

(A) In particular, given a  $k$ -forest  $F$ , the first component of  $\lambda(F)$  is generated recursively in the following way:

Set  $i = 1$

1. Search for the leaf in  $F$  with the smallest label  $v_i$ .
2. Remove the edge  $(v_i, v)$  from  $F$ .
3. Set the  $i$ -th element of  $\omega_k$  to  $v$  i.e. set  $\omega_{k,i} = v$ .
4. Set  $i = i + 1$ .
5. If  $i < N - k$  start from 1 again, otherwise set the third component of  $\lambda(F) = r$  where  $(v_r, r)$  is the only remaining edge in  $F$ .

In the above procedure a root of a tree forming  $F$  is not considered to be a leaf after removal of the tree stemming from it. The second component of  $\lambda(F)$  simply lists all of the roots of  $F$ . It is obvious that after repeating the above procedure  $N - k - 1$  times we will end up with only the roots of  $k - 1$  of the trees forming  $F$  plus the remaining leaf  $v_r$  connected to  $r$ . Notice that the linear order of the set  $\{1, \dots, N\}$  implies that the mapping  $\lambda$  is well defined.

(B) Next we start with a triple  $(\omega_k, A_k, r)$ ,  $r \in A_k$ , and show that there is a  $k$ -forest  $F$ , such that  $\lambda(F) = (\omega_k, A_k, r)$ . Indeed, one can set the  $k$  roots of  $F$  to be the elements of  $A_k$ , and after that one can apply the following procedure generating the rest of  $F$ : Set  $j = 1$ . Form the set  $B_k = \{1, \dots, N\} \setminus A_k$ . Create  $k$  roots for  $F$  using the elements in  $A_k$ .

- (a) Find the smallest element  $i \in B_k$  that is not equal to  $\omega_{k,l}$ ,  $l = j, j + 1, \dots, N - k - 1$ .
- (b) Form an edge  $(i, \omega_{k,j})$  in  $F$ .
- (c) Set  $B_k = B_k \setminus i$  and set  $j = j + 1$
- (d) If  $j > N - k - 1$  connect the only element of  $B_k$  to  $r$  and then stop, otherwise start from (a) again.

Since at every step  $j$  we remove from  $B_k$  elements not present in  $\omega_k$  starting from the  $j$ -th position on, none of the elements of  $B_k$  can participate in a cycle. Therefore, the resulting graph is a  $k$ -forest with its roots the elements in  $A_k$ .

- (C) Finally, given two different  $k$ -forests  $F_1$  and  $F_2$  we claim that  $\lambda(F_1) \neq \lambda(F_2)$ , where the equality between two points in the space  $\Omega_{N_k} \times \mathcal{A}_k \times \{1, \dots, N\}$  is defined in the obvious way. Clearly  $\lambda(F_1) \neq \lambda(F_2)$  if  $F_1$  and  $F_2$  have different sets of roots. If both  $k$ -forests have the same set of roots, then they must differ in at least one edge. If now, we assume to the contrary that  $\lambda(F_1) = \lambda(F_2)$  that means that the only way  $F_1$  can differ from  $F_2$  is if there is at least one edge in  $F_1$  not present in  $F_2$  or vice versa. At the same time the equality of the components of  $\lambda(F_1)$  and  $\lambda(F_2)$  means that the procedure in part (A) removes consecutively exactly the same edges from  $F_1$  and from  $F_2$  which in its turn implies that the two  $k$ -forests have the same set of edges, which contradicts our assumption about  $F_1$  and  $F_2$  being different from each other.

(A), (B) and (C) together show that the mapping  $\lambda$  is, indeed, a bijection from  $\mathcal{F}_k$  to  $\mathcal{B}_k$ . Thus, the problem of counting all of the  $k$ -forests,  $k = 1, \dots, N$ , on  $N$  vertices, reduces to counting the elements of  $\cup_{k=1}^N \mathcal{B}_k$ . One should notice that the mapping  $\lambda$  is not defined for  $k = N$  but this is the trivial case where  $\mathcal{B}_k$  has just one member, namely the  $N$ -forest where each vertex in the graph happens to be a root for one of the  $N$  trees composing the forest. There are  $\binom{N}{k}$  ways of selecting an element of  $A_k \in \mathcal{A}$ ,  $k$  ways of selecting  $r \in A_k$ , and  $N^{n-k-1}$  of sequences in  $\Omega_{N_k}$ . Therefore, for each fixed  $k$ , the cardinality of  $\mathcal{B}_k$  is  $\binom{N}{k} k N^{N-k-1}$ , and since the sets  $\mathcal{B}_k$ ,  $k = 1, \dots, N$  are pair wise disjoint, the cardinality of  $\cup_{k=1}^N \mathcal{B}_k$  is

$$\sum_{k=1}^N \binom{N}{k} k N^{N-k-1} = (N+1)^{N-1} \quad \bullet$$

The following two examples illustrate the procedures described in part (A) and part (B) of the proof of *Theorem 1*.

**Example 2.1:** Suppose we are given the state transition diagram in Figure 1. If one applies the procedure from part (A), then one will produce the triple  $(\omega_2, A_2, 2)$ , where  $\omega_2 = 4, 6, 1, 4, 1, 2, 4, 7, 9$ ,  $A_2 = \{2, 7\}$ , and where the edges  $(3, 4)$ ,  $(5, 6)$ ,  $(6, 1)$ ,  $(8, 4)$ ,  $(10, 1)$ ,  $(11, 4)$ ,  $(4, 7)$ , and  $(12, 9)$  have been consecutively removed from the 2-forest in Figure 1.

**Example 2.2:** Suppose we are given the triple  $(\omega_3, A_3, 1)$  where  $\omega_3 = 3, 10, 12, 2, 5, 5, 2, 5$ , and where  $A_3 = \{1, 5, 9\}$ . If one applies the procedure from part (B), then the following edges will be generated in the order they are listed:  $(4, 3)$ ,  $(3, 10)$ ,  $(6, 12)$ ,  $(7, 2)$ ,  $(8, 5)$ ,  $(10, 5)$ ,  $(11, 2)$ ,  $(2, 5)$ , and  $(12, 1)$ . Thus, the 3-forest in Figure 2 will be generated.

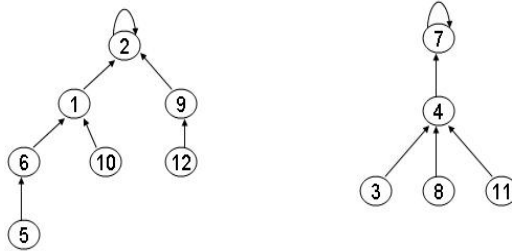


Figure 1: State Transition Diagram for Example 2.1

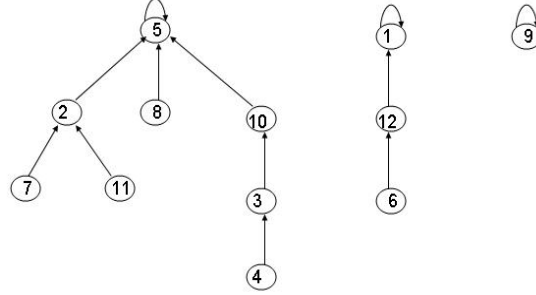


Figure 2: State Transition Diagram for Example 2.2

### 3 Extension of the Algorithms to include attractor cycles

In this section, we present the extension of the two algorithms given in the paper to include attractor cycles.

#### Algorithm 1 extension

STEP1: Randomly generate a set of  $k$  attractor states and their connections.\* If the connections generate an attractor cycle with length  $>$  Max Cycle Length, then repeat STEP1. If STEP1 has been repeated more than a pre-specified number of times, then terminate the algorithm.

STEP2: Randomly pick up a predictor set  $W$ , where each  $W_i$  has not less than  $m$  and not more than  $M$  elements. If STEP2 has been repeated more than a pre-specified number of times go back to STEP1.

STEP3: Check if the selected attractor set is compatible with  $W$ , i.e. the attractor set transitions \*\* of the state transition diagram are checked for compatibility against  $W$ . If the attractor set is not compatible with  $W$ , then go back to STEP2; otherwise continue to STEP4.

STEP4: Fill in the entries of the truth table that correspond to the attractor set transitions generated in STEP1. Using the predictor set  $W$ , randomly fill in the remaining entries of the truth table. If STEP4 has been repeated more than a pre-specified number of times go back to STEP2.

STEP5: Search for cycles of length  $>$  Max Cycle Length in the state transition diagram  $\tilde{\Gamma}$  that is associated with the truth table generated in STEP4. If a cycle is found, then go back to STEP4; otherwise continue to STEP6.

STEP6: If  $\tilde{\Gamma}$  has less than  $l$  or more than  $L$  level sets, go back to STEP4; otherwise continue to STEP7.

STEP7: Save the generated BN and terminate the algorithm.

\* If Max Cycle Length is given to be 1, then we connect each attractor to itself. Otherwise we include random connections between attractors. Let the random attractors selected be  $a_1, a_2, a_3$  and  $a_4$ . If Max Cycle Length is 1, then the connections are  $a_1 \rightarrow a_1, a_2 \rightarrow a_2, a_3 \rightarrow a_3, a_4 \rightarrow a_4$ . Otherwise, there are 4 attractors and hence we choose a random permutation of numbers 1 to 4. Say the random permutation is 1, 3, 2 and 4. Then the connections between the attractors  $a_1, a_2, a_3$  and  $a_4$  will be  $a_1 \rightarrow a_1, a_2 \rightarrow a_3, a_3 \rightarrow a_2, a_4 \rightarrow a_4$  (Figure 3). If the random permutation is 4, 3, 1 and 2, then the transitions between the attractors will be  $a_1 \rightarrow a_4, a_2 \rightarrow a_3, a_3 \rightarrow a_1, a_4 \rightarrow a_2$ .

\*\* For the first random permutation 1, 3, 2, 4, the transitions are  $a_1 \rightarrow a_1, a_2 \rightarrow a_3, a_3 \rightarrow a_2, a_4 \rightarrow a_4$ ; for the random permutation 4, 3, 1, 2 the transitions are  $a_1 \rightarrow a_4, a_2 \rightarrow a_3, a_3 \rightarrow a_1, a_4 \rightarrow a_2$ .

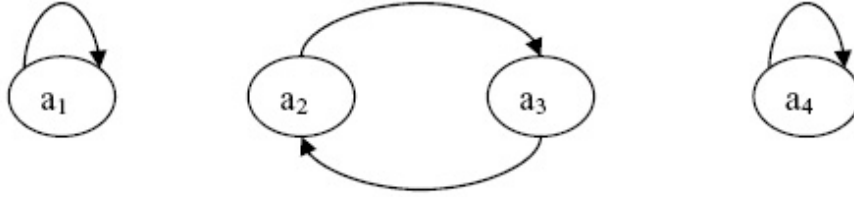


Figure 3: Connections among Attractors

### Algorithm 2 extension

STEP1: Randomly generate a state transition diagram  $\tilde{\Gamma}$  that satisfies the design goals about the attractor structure \*\*\* and level set structure. If STEP1 has been repeated more than a pre-specified number of times, then terminate the algorithm.

STEP2: Fill in the truth table using  $\tilde{\Gamma}$ .

STEP3: 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  $M$ , then elements go back to STEP1; otherwise continue to STEP4.

STEP4: Save the generated BN and terminate the algorithm.

\*\*\* When attractor cycles of length  $\leq$  Max Cycle Length are allowed, then connections between attractors are permitted and only those state transition diagram are selected for the subsequent steps whose cycle lengths are  $\leq$  Max Cycle Length.

## 4 Performance Analysis

Several simulations were carried out in Matlab to evaluate the performance of the two algorithms. Table 1 shows the performance of Algorithm 1 for the case of  $n = 6$ ,  $2 \leq k \leq 4$ ,  $m = 1$ ,  $l = 1$ , and  $L = 2^6 - 1$ . The number of maximum repetitions of STEP1, STEP2, and STEP4 were set to 10, 20 and 500m respectively. The total execution time for this simulation was 13123.875 seconds or roughly 3.5 hrs on a 2.4 GHz P4 Intel Xeon Processor.

n	M	BNs saved at STEP7	BNs searched in STEP5
6	1	1267	7670
6	2	1375	10160
6	3	2396	19124
6	4	1399	27590
6	5	1960	35060
6	6	1704	37550

Table 1: Simulations for Algorithm1

Table 2 shows the performance of Algorithm 1 for the case of  $n = 10$ ,  $1 \leq k \leq 6$ ,  $m = 1$ ,  $l = 1$ , and  $L = 2^{10} - 1$ . The number of maximum repetitions of STEP1, STEP2, and STEP4 were set to 10, 15 and 1000, respectively. The execution time for this simulation, was 58842.5 seconds or around 16 hours on an identical machine.

The significant increase in the run time for the case of 10 genes can be attributed to two major factors: first, the number of the cyclic state transition diagrams increases as the number of genes increases, e.g. Table 2; and second, the low probability mass of  $k$ -forests in the space of all directed graphs.

n	M	BNs saved at STEP7	BNs searched in STEP5
10	9	80	30090

Table 2: Simulation for Algorithm1

Table 3 shows the performance of Algorithm 2 for the case  $n = 3, 1 \leq k \leq 2, m = 1, M = 2, l = 2,$  and  $L = 5$ . There were 1000 BNs generated in STEP1 and the simulation time was 4.01 seconds. One can notice the low frequency of successfully generated BNs even for such a small number of genes. The simulation for the case  $n = 6, 2 \leq k \leq 4, m = 1, M = 5, l = 4,$  and  $L = 15$  confirms that observation: the algorithm did not generate any BN during the first  $3 \times 10^6$  iterations. It took 78329.2 seconds or approximately 21hrs to run this many iterations.

n	M	BNs saved at STEP4
3	1	5
3	2	43

Table 3: Simulations for Algorithm 2

The reason for such a huge difference in the performance of the two algorithms is the fact 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. One can easily see that 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,  $k = 1, \dots, N$ ;

$$\frac{\binom{n}{m}^n N^{2^m}}{(N+1)^{N-1}}$$

For the case  $n = 6, m = 5$  this ratio is approximately  $1.7911 \times 10^{-52}$

The high run time for the algorithms has compelled us to write them in a lower level language and to optimize the code as much as possible. In the following discussion we provide the performance analysis of Algorithm 1 using C code. The next two tables gives the performance of Algorithm 1 using the same parameters as in the Matlab Code.

Table 4 shows the performance of Algorithm 1 for the case of  $n = 6, 2 \leq k \leq 4, m = 1, l = 2,$  and  $L = 20$ . The number of maximum repetitions of STEP1, STEP2, and STEP4 were set to 10, 20 and 500, respectively. The total execution time for this simulation was 99 seconds on a 2.4 GHz P4 Intel Xeon Processor.

n	M	BNs searched at STEP5	BNs in STEP6	BNs saved at STEP7
6	1	5500	1000	1000
6	2	13000	2503	1609
6	3	20500	2783	2531
6	4	31500	2012	1868
6	5	44500	2618	2524
6	6	34000	1995	1971

Table 4: Simulations for Algorithm1

Table 5 shows the performance of Algorithm 1 for the case of  $n = 10, 1 \leq k \leq 6, m = 1, l = 2,$  and  $L = 35$ . The number of maximum repetitions of STEP1, STEP2, and STEP4 were set to 10, 15 and 1000, respectively. The execution time for this simulation was 779 seconds on an identical machine.

n	M	BNs searched at STEP5	BNs in STEP6	BNs saved at STEP7
10	9	60000	295	231

Table 5: Simulation for Algorithm1

To illustrate the performance of the C code when cyclic attractors are permitted, we considered 6 genes with parameters  $n = 6$ ,  $1 \leq k \leq 6$ ,  $m = 1, M = 3$ ,  $l = 2$ , and  $L = 12$ , and maximum length of attractor cycle being 2 (a reasonable number because in practice, if cycles are permitted, we assume them to be short). The number of maximum repetitions of STEP1, STEP2, and STEP4 has been set to 10, 15 and 1000, respectively. The results are provided in Table 6. The execution time was 47.0 seconds. One of the Boolean Networks saved at STEP 7 is shown in Figure 4

n	M	BNs searched at STEP5	BNs in STEP6	BNs saved at STEP7
6	3	43000	8602	7817

Table 6: Simulation for enhanced Algorithm1

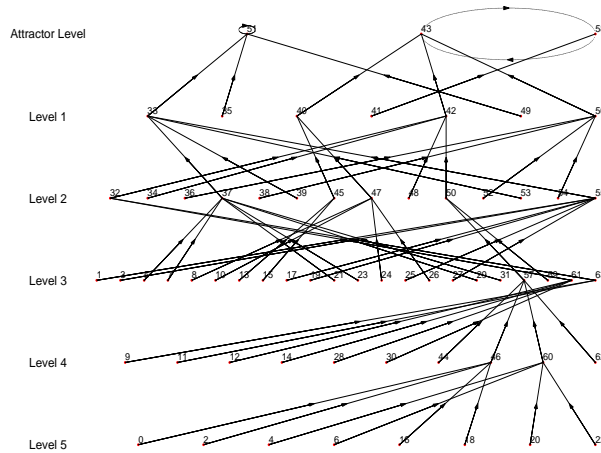


Figure 4: BN with maximum length of attractor cycle=2.

The efficiency of the C Code has made the problem tractable for larger numbers of genes. For example, when we run the code for 12 genes with parameters  $1 \leq k \leq 10$ ,  $m = 1, M = 4$ ,  $l = 2$ , and  $L = 40$ , and maximum length of attractor cycle permitted being 2, the execution time is 317 seconds, and it generates 600 Boolean networks with the specified constraints.

## 5 Examples

Given below is a walk-through example to show how algorithm 2 work in the particular case of 3 genes:

### Example 4.2 (Algorithm 2)

Suppose that  $k = 2$ ,  $m = 1$ ,  $M = 2$ ,  $l = 1$  and  $L = 3$ .

Next, suppose that the transition diagram shown in Figure 5 is randomly generated in STEP1. The truth table resulting from STEP2 is shown in Table 7. STEP3: It is clear from this truth table that  $W_1 = \{x_1, x_2, x_3\}$ , and since it has more than  $M = 2$  elements the algorithm returns to STEP1.

On the other hand if the transition diagram shown in Figure 6 was generated in STEP1, then STEP2 would produce the truth table shown in Table 8.

Now each  $W_i$ ;  $i = 1, 2, 3$  has no more than  $m = 2$  elements, and the algorithm successfully terminates producing a BN with the truth table shown in Table 4 and state transition diagram from Figure 6.

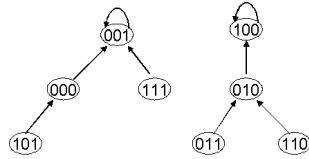


Figure 5: First State Transition Diagram for Example 4.2

Gene Values	$f_1$	$f_2$	$f_3$
0 0 0	0	0	1
0 0 1	0	0	1
0 1 0	1	0	0
0 1 1	0	1	0
1 0 0	1	0	0
1 0 1	0	0	0
1 1 0	0	1	0
1 1 1	0	0	1

Table 7: First Truth Table for Example 4.2

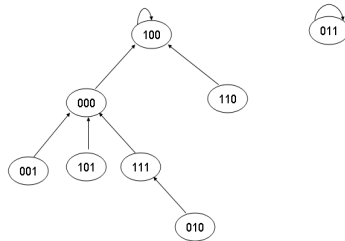


Figure 6: Second State Transition Diagram for Example 4.2

## Simulation Tools

A free PBN toolbox is available on the web at <http://www2.mdanderson.org/app/ilya/PBN/PBN.htm>. Its description is given by: "This toolbox is written in MATLAB and can be used to work with Boolean Networks and Probabilistic Boolean Networks. It includes functions for simulating the network dynamics, computing network statistics (numbers and sizes of attractors, basins, transient lengths, Derrida curves, percolation on 2-D lattices, influence matrices), computing state transition matrices and obtaining stationary distributions,

Gene Values	$f_1$	$f_2$	$f_3$
0 0 0	1	0	0
0 0 1	0	0	0
0 1 0	1	1	1
0 1 1	0	1	1
1 0 0	1	0	0
1 0 1	0	0	0
1 1 0	1	0	0
1 1 1	0	0	0

Table 8: Second Truth Table for Example 4.2

inferring networks from data, generating random networks and functions, visualization and printing, intervention, and membership testing of Boolean functions.”

Matlab and C codes to generate BNs using Algorithms given in the paper is available at the website <http://gsp.tamu.edu/Publications/BNs/bn.htm>.

## 6 REFERENCES

Bittner, M., *et al.* (2000)., ”Molecular classification of cutaneous malignant melanoma by gene expression profiling”, *Nature*, 2000. 406(6795): p. 536-40.

Kim, S., H. Li, Y. Chen, N. Cao, E.R. Dougherty, M.L. Bittner, and E.B. Suh (2002) ”Can Markov chain Models mimic biological regulation?”, *Biological Systems*, 10, p. 337-357.

Nielsen, T.O., *et al.* (2004), ”Immunohistochemical and clinical characterization of the basal-like subtype of invasive breast carcinoma,” *Clinical Cancer Research*, 10(16): p. 5367-74.

Weeraratna, A.T., *et al.*(2002)., ”Wnt5a signaling directly affects cell motility and invasion of metastatic melanoma”, *Cancer Cell*, 1: p. 279-88.