The Second International Symposium on Optimization and Systems Biology (OSB'08)
Lijiang, China, October 31– November 3, 2008

A Genetic Algorithm for Optimal Control of Probabilistic Boolean Networks

Wai-Ki Ching[1] Ho-Yin Leung*[1] Nam-Kiu Tsing[1]
Shu-Qin Zhang[2]

[1]Advanced Modeling and Applied Computing Laboratory, Department of Mathematics, The University of Hong Kong, Hong Kong.
Emails:wching@hkusua.hku.hk, obliging@hkusua.hku.hk, nktsing@hku.hk
[2]Faculty of Mathematical Sciences, Fudan University, Shanghai, China.
Email:zhangs@fudan.edu.cn

Abstract We study the problem of finding optimal control policies for Probabilistic Boolean Networks (PBNs). Boolean Networks (BNs) and PBNs are effective tools for modeling genetic regulatory networks. A PBN is a collection of BNs driven by a Markov chain process. It is well-known that the control/intervention of a genetic regulatory network is useful for avoiding undesirable states associated with diseases like cancer. The optimal control problem can be formulated as a probabilistic dynamic programming problem. However, due to the curse of dimensionality, the complexity of the problem is huge. The main objective of this paper is to introduce a Genetic Algorithm (GA) approach for the optimal control problem. Numerical results are given to demonstrate the efficiency of our proposed GA method.

Keywords Boolean Networks; Dynamic Programming; Genetic Algorithm; Intervention; Optimal Control Policy; Probabilistic Boolean Networks.

1 Introduction

To understand and model the mechanism in which the cells execute and control a large number of operations is an important research focus in systems biology. In fact, a lot of mathematical models have been proposed for the above purpose, such as neural networks, differential equations, directed graphs etc [5, 12]. One of the approaches is to model a genetic regulatory network and then infer its structure by real gene expression data. For this purpose, Boolean Network (BN) and its generalization Probabilistic Boolean Network (PBN) have been proposed and received much attention. BN was first introduced by Kauffman [6, 7]. In a BN, each gene is regarded as a vertex of the network and is then quantized into two levels only (expressed (1) or un-expressed (0)) though the idea can be extended to the case of more than two levels. The target gene is predicted by several genes called its input genes through a Boolean function. If the input genes and the Boolean functions are given, then a BN is defined and it can be considered as a deterministic dynamical system.

Since a biological system has its stochastic nature and the microarray data sets used to infer the network structure are usually not accurate because of the experimental noise

in the complex measurement process, a deterministic model may not be appropriate. To cope with the randomness of the network, Shmulevich *et al.* [13, 14] extended the BN to the PBN that can take the advantage of the rule-based properties of BNs. The dynamics of a PBN can be studied in the context of a standard Markov chain. Therefore the theory of Markov chain process [1, 2] can be used to analyze the network. The PBNs in [13, 14] are called instantaneously random PBNs. To stabilize the network, later random gene perturbations were introduced to the network in [15]. Another extension of the instantaneously random PBN is the context-sensitive PBN [10]. The steady-state probability of the network carries important information of the captured network. Fast numerical method [18] and approximation method [3] have also been proposed for the computation of the steady-state probability distribution. It is an important goal of the biologists to design therapy and strategy for the intervention of the dynamics of a biological network, in particular, in the case of diseases like cancer. Genetic intervention has been proposed to facilitate a PBN to evolve to some targeted desirable state, see for instance [9, 11, 15, 16]. Later Ching et al. [4] give a new optimal control formulation that considers the case of hard constraints, i.e., to include a maximum upper bound for the number of controls that can be applied to the PBN. The new formulation can be applied to both perturbed and context-sensitive PBNs. On one hand, during the treatment of one patient, the cost of the operation conducted may be expensive. On the other hand, it may be impractical to have unlimited operations (such as chemotherapy) to a patient.

Here we introduce a genetic algorithm for computing the optimal control strategies for the model proposed in [4]. The paper is organized in the following sequel. In Section two, we give a probabilistic dynamic programming formulation for the optimal finite-horizon control problem proposed in [4]. In Section three, we present our genetic algorithm for solving the optimal control problem. In Section four, numerical examples are given to demonstrate the efficiency of our proposed GA method. Finally, concluding remarks are given to address further research issues in Section five.

2 The Optimal Finite-Horizon Control Problem

We briefly present the discrete optimal control problem studied in [4]. Starting with an initial state probability distribution $\mathbf{v}_0$ the PBN will evolve according to two possible transition probability matrices P_0 and P_1. If there is no external control applied to the network, the PBN evolves according to the fixed transition probability matrix P_0. When a control is applied to the network, the PBN will evolve according to another known transition probability matrix P_1 (here we assume that P_1 has more favorable steady states). However, the network will return back to P_0 again when no more control is applied to the network. We note that there can be more than one type of control to choose in each time step. But for simplicity of discussion, here we only assume that there is only one possible control. We suppose that the maximum number of controls that can be applied to the network during the finite investigation period T (finite-horizon) is K where $K \leq T$. The objective here is to find an optimal control policy such that the state of the network is close to a target state vector $\mathbf{z}$. The vector $\mathbf{z}$ can be an unit vector (a desirable state) or a probability distribution (a weighted average of desirable states). We first define the following state probability distribution vectors $\mathbf{v}(i_k i_{k-1} \dots i_1) = P_{i_k} \cdots P_{i_1} \mathbf{v}_0$ to represent all the possible network state probability distribution vectors at time k. Here $i_1, \dots, i_k \in$

$\{0,1\}$ and $\sum_{j=1}^{k} i_j \leq K$ and $i_k i_{k-1} \dots i_1$ is a Boolean string of size k. We then define $U(k) = \{\mathbf{v}(i_k i_{k-1} \dots i_1) : i_1, \dots, i_k \in \{0,1\}$ and $\sum_{j=1}^{k} i_j \leq K\}$ to be the set containing all the possible state probability vectors at time k. We note that one can conduct a forward calculation to compute all the state vectors in the sets $U(1), U(2), \dots, U(T)$ recursively. Here the main computational cost comes from the matrix-vector multiplication and the cost for one matrix-vector multiplication is $O(2^{2n})$ where n is the number of genes in the network. Since in a PBN, the transition probability matrix is sparse, the computational complexity should be less than $O(T2^{2n})$. However, when both K and T are large the size of the problem can be huge. Therefore one has to consider heuristic method such as Genetic Algorithm (GA).

There are at least two possible formulations for our optimal control problem [4]. The first one is to minimize the terminal distance with the target vector $\mathbf{z}$, i.e.,

$$\min_{\mathbf{v}(i_T i_{T-1} \dots i_1) \in U(T)} ||\mathbf{v}(i_T i_{T-1} \dots i_1) - \mathbf{z}||_2. \tag{1}$$

The second one is to minimize the overall average of the distances of the state vectors $\mathbf{v}(i_t \dots i_1)$ $(t = 1, 2, \dots, T)$ to the target vector $\mathbf{z}$, i.e.,

$$\min_{\mathbf{v}(i_T i_{T-1} \dots i_1) \in U(T)} \frac{1}{T} \sum_{t=1}^{T} ||\mathbf{v}(i_t \dots i_1) - \mathbf{z}||_2. \tag{2}$$

A dynamic programming formulation for solving the above problem can be found in [4].

3 The Genetic Algorithm

In this section, we introduce briefly the idea of the proposed Genetic Algorithm (GA) [8]. A policy vector is a vector $\mathbf{p}$ of length T. Its i-th position encodes the policy used in the i period. In the first step, we generate a random population of size N which consists of the policies vectors. The cost of each policy vector is evaluated which is subsequently turned into the probability that it would be picked for the next generation. The calculation of probabilities will be discussed shortly in this section. In the second step, we pick 2 policies from current generation with replacement according to the probabilities calculated above. Crossover occurs at a random position with chance p_c. For example (policies are bold faced for illustration purpose), $\mathbf{p_1} = (00001)$, $\mathbf{p_2} = (11100)$ Crossover at position 2 results in $\mathbf{p_1'} = (\mathbf{00}100)$, $\mathbf{p_2'} = (11\mathbf{001})$ Then each position of the policies mutates with probability p_m. For example, the policy $\mathbf{p_1'}$ is mutated at position 4: $\mathbf{p_1''} = (001\mathbf{1}0)$ We pick 2 policies at a time and then perform crossover or mutation whenever necessary until there are N or $N+1$ policies whichever is even. If N is odd, one policy vector is randomly removed from the generation. We then calculate the costs and the probabilities of each policy vector. Go back to the second step if the stopping criteria are not met. The algorithm terminates otherwise.

Genetic algorithms are good at retaining patterns having good objective value. These elite descendants would produce quality offsprings. Hence better solutions are obtained when the population evolves. To ensure better accuracy, the whole process (Steps 1 and 2) is repeated conditionally. Here we call the "completion of one generation evolution" to be one iteration. When the difference between the best solution in the previous iterations

Table 1: The numerical results when $T = 12$ (objective function (1) is used)

K		1	3	5	7
Computational time in seconds	DP	0.3	5.1	22.5	40.4
	GA	(0.7, 0.1)	(4.5, 1.1)	(10.2, 2.4)	(27.9, 10.2)
Optimal value	DP	0.1021	0.0714	0.0632	0.0632
	GA	(0.1028, 0.0029)	(0.0728, 0.0018)	(0.0640, 0.0013)	(0.0632, 0.0001)

and the current iteration is within a tolerance of, say in our case, $\varepsilon = 0.0001$ for 5 times consecutively, we assume that little progress can be made by doing more iterations. We then stop repeating the process and obtain the best solution as the output. Within one iteration, there are stopping criteria too. When the following criteria are both satisfied, current iteration would stop and proceed to the next:

(a) The standard deviation of the costs divided by the mean of the costs is less than 0.3, and
(b) the minimum cost of the current generation divided by the minimum among the previous iterations is bigger than 1.2.

When criterion (a) is satisfied, the costs have small variations and hence the iteration is almost converged. When criterion (b) is satisfied, the current minimum is considerably larger than the previous best minimum. With both, it implies that the current generation is almost converged and there is quite a distance between the current best and the previous best solution, hence very little progress is likely to be made if we carry on.

In the encoding of the policies, a policy vector is a 0-1 vector $\mathbf{p}$ of length T, where T is the time horizon of the control $\mathbf{p} = (i_1 i_2 i_3 \dots i_T), \quad i_k \in \{0,1\}$. The maximum number K of controls applied to the system is fixed, so some of the policy vectors are invalid (number of controls exceeds K). We cannot remove or modify all the invalid policies in the genetic algorithm otherwise it is very likely that the policies possess certain pattern only, making the evolution slow. To overcome this problem, we allow the existence of such invalid policies in the population but only their first K controls will be considered, i.e. if $(i_1 i_2 \dots i_T)$ is the policy vector then it means $(i_1 i_2 \dots i_U 00 \dots 0)$ where

$$U = \begin{cases} \min\left\{ k : \sum_{n=1}^{k} i_n = K \right\} & \text{if } \sum_{n=1}^{T} i_n \geq K; \\ U = T. \quad \text{otherwise.} \end{cases}$$

For example, policy vector (**101** 01) actually means policy (**101** 00) when $K = 2$. Likewise, the policy vector (**101** 11) encodes the same information.

In our algorithm, we take the size of population to be $\frac{1}{2}KT$. Here we assume that the size of population to be vary directly with K and T. The factor $1/2$ is picked according to our numerical experiences. To simplify our algorithm, the size is chosen to be the biggest even number smaller than or equal to $\frac{1}{2}KT$. To facilitate the convergence of the algorithm, the initial population is first generated by the policies having number of controls exactly equal to K. For example, (10100) and (00011) are possible to be generated but (11100) is not. Finally to transform the costs to probabilities, we use a simple scheme as follows: Since we are aiming at a minimum cost, we take reciprocal of cost, and then normalize the reciprocals to yield probabilities. Since the cost function has high complexity, the

Table 2: The numerical results when $T = 16$ (objective function (1) is used)

K		1	3	5	7
Computational time in seconds	DP	0.5	16.9	143.8	186.3
	GA	(2.8, 0.5)	(20.4, 5.9)	(60.5, 19.2)	(144.0, 31.8)
Optimal value	DP	0.1023	0.0712	0.0634	0.0631
	GA	(0.1030, 0.0028)	(0.0728, 0.0025)	(0.0646, 0.0015)	(0.0632, 0.0001)

Table 3: The numerical results when $T = 20$ (objective function (1) is used)

K		1	3	5	7
Computational time in seconds	DP	0.7	41.3	601.2	3447.2
	GA	(10.4, 1.6)	(46.3, 12.3)	(122.2, 41.2)	(186.5, 75.9)
Optimal value	DP	0.1024	0.0712	0.0634	0.0632
	GA	(0.1031, 0.0028)	(0.0731, 0.0019)	(0.0664, 0.0029)	(0.0633, 0.0003)

calculated costs are stored in the memory as a table. Each time when we compute the cost, we need to check if there is such entry in the table first. If it has not been calculated, we will compute it and store it in the table. Otherwise the value is read from the table without computation.

4 Experimental Results

In this section, we apply the optimal control to a twelve-gene network [3]. We assume that there are two Boolean functions $f_1^{(i)}$ and $f_2^{(i)}$ associated with each gene i. All the Boolean functions and their variables are generated randomly as in [3]. Due to the huge size of the resulting transition probability matrix, in the numerical experiments we applied the matrix approximation method in [3] and also the matrix-vector multiplication method in [18] to speed up the computational time. We assume that the control when applied to the network will activate gene 1, i.e., gene 1 is expressed. Then the transition probability matrix when a control is applied is given by $P_1 = \begin{pmatrix} 0 & 0 \\ I & I \end{pmatrix}$ where $\mathbf{0}$ and I are the 2^{11}-by-2^{11} zero matrix and the identity matrix respectively.

In the numerical experiment, we assume that the initial state vector of the network is the uniform distribution vector $\mathbf{v}_0 = \frac{1}{2^{12}}(1,1,\ldots,1)^T$. The target vector is $\mathbf{z} = \frac{1}{2^{12}}(\mathbf{0},\mathbf{1})^T$ where $\mathbf{0}$ and $\mathbf{1}$ are the 1×2^{11} zero vector and the 1×2^{11} vector of all ones respectively.

The following tables compare both the computational time and the optimal objective values obtained in the two methods: Dynamic Programming (DP) [4] and Genetic Algorithm (GA). For GA, we perform the calculations 50 times to get the average μ and the variance σ^2 and the results are presented in an ordered pair (μ,σ) format. We consider the total time T to be $12, 16, 20$ and we try several different maximum number of controls $K = 1,3,5,7$. We remark that the DP approach will give the optimal policy. However,

Table 4: The numerical results when $T = 12$ (objective function (2) is used)

K		1	3	5	7
Computational time in seconds	DP	0.3	5.2	23.0	41.1
	GA	(1.0, 0.1)	(6.3, 1.1)	(8.0, 2.5)	(9.3, 2.4)
Optimal value	DP	0.1098	0.0875	0.0655	0.0444
	GA	(0.1098, 0.0000)	(0.0875, 0.0003)	(0.0657, 0.0009)	(0.0444, 0.0000)

Table 5: The numerical results when $T = 16$ (objective function (2) is used)

K		1	3	5	7
Computational time in seconds	DP	0.5	16.8	145.0	481.9
	GA	(1.8, 0.1)	(18.0, 1.7)	(31.3, 9.8)	(21.6, 5.5)
Optimal value	DP	0.1160	0.0992	0.0824	0.0656
	GA	(0.1160, 0.0000)	(0.0993, 0.0004)	(0.0824, 0.0000)	(0.0657, 0.0004)

Table 6: The numerical results when $T = 20$ (objective function (2) is used)

K		1	3	5	7
Computational time in seconds	DP	0.7	42.0	604.9	3487.8
	GA	(6.9, 0.2)	(42.5, 2.5)	(117.5, 25.7)	(78.7, 31.4)
Optimal value	DP	0.1198	0.1063	0.0928	0.0793
	GA	(0.1198, 0.0000)	(0.1064, 0.0004)	(0.0929, 0.0002)	(0.0795, 0.0005)

due to the curse of dimensionality, it is not efficient for large T and K. The computational time of the DP approach will increase with respect to K and T but not in the case of GA, see Tables 5 and 6 for instance. For larger values of K and T, the GA approach is much faster than the DP approach with exact or very close optimal objective values. We see that for smaller values of T, e.g. $T = 12$, GA approach does not has advantage over the DP approach and can be even worse for small value of K such as $K = 1,3$. However, when both T and K are getting large, e.g. $T = 20$ and $K = 5,7$, GA has much better performance in terms of the computational time.

5 Concluding Remarks

In this paper, we introduce a Genetic Algorithm (GA) for solving the optimal control policy for a PBN. Numerical results indicated that the GA is very efficient for solving the captured problem and the accuracy is also very high. The followings are our future research issues. (i) We will apply our proposed algorithm to more real networks. (ii) We will further improve the GA and extend it handle to the case of more than two control policies.

Acknowledges

Research support in part by HKRGC Grant 7017/07P and HKU CRCG Grants and HKU strategic theme grant on computational sciences.

References

[1] Ching, W. Fung, E., Ng, M. and Akutsu T. (2005). On construction of stochastic genetic networks based on gene expression sequences. International Journal of Neural Systems, 15, 297-310.

[2] Ching, W. and Ng, M. (2006). Markov Chains : models, algorithms and applications. International Series on Operations Research and Management Science, Springer, New York.

[3] Ching, W. Zhang, S., Ng, M. and Akutsu T. (2007). An approximation method for solving the steady-state probability distribution of probabilistic Boolean networks. Bioinformatics, 23 (2007) 1511-1518.

[4] Ching, W., Zhang, S., Jiao, Y., Akutsu, T. and Wong, A. Optimal finite-horizon control for probabilistic Boolean networks with hard constraints, The International Symposium on Optimization and Systems Biology (OSB 2007), Lecture Notes in Operations Research 7, Series Editors: Ding-Zhu Du and Xiang-Sun Zhang, (2007) 21-28.

[5] de Jong, H. (2002). Modeling and simulation of genetic regulatory systems: A Literature Review. J. Comput. Biol., 9, 69-103.

[6] Kauffman, S.A. (1969). Metabolic stability and epigenesis in randomly constructed genetic nets. Theoretical Biology, 22, 437-67.

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

[8] M. Mitchell (1996) An introduction to genetic algorithms. MIT Press. Cambridge, Mass.

[9] Ng, M., Zhang, S., Ching, W. and Akutsu, T. (2006). A control model for Markovian genetic regulatory network. Transactions on Computational Systems Biology, Springer, 4070, 36-48.

[10] Pal, R., Datta, A., Bittner, M. L., and Dougherty, E. R. (2005). Intervention in context-sensitive probabilistic Boolean networks. Bioinformatics, 21 (7), 1211-18.

[11] Pal, R., Datta, A., and Dougherty, E. R. (2006). Optimal Infinite-Horizon Control for Probabilistic Boolean networks. IEEE Tran. Signal Processing, 54 (6), 2375-87.

[12] Smolen, P., Baxter, D. and Byrne, J. (2000). Mathematical modeling of gene network. Neuron, 26, 567-80.

[13] Shmulevich,I., Dougherty, E.R., Kim, S., Zhang, W. (2002). Probabilistic Boolean Networks: A rule-based uncertainty model for gene regulatory networks. Bioinformatics, 18, 261-74.

[14] Shmulevich, I., Dougherty, E., Kim, S. and Zhang, W. (2002). From Boolean to probabilistic Boolean networks as models of genetic regulatory networks. Proceedings of the IEEE, 90, 1778-92.

[15] Shmulevich, I., Dougherty, E.R., Zhang, W. (2002). Gene perturbation and intervention in probabilistic Boolean networks. Bioinformatics, 18, 1319-31.

[16] Shmulevich, I., Dougherty, E.R., Zhang, W. (2002). Control of stationary behavior in probabilistic Boolean networks by means of structural intervention. Biological Systems, 10, 431-46.

[17] Shmulevich, I., Gluhovsky, I., Hashimoto, R.F., Dougherty, E.R. and Zhang W. (2003). Steady-state analysis of genetic regulatory networks modeled by probabilistic Boolean networks. Comparative and Functional Genomics, 4, 601-08.

[18] Zhang, S., Ching, W., Ng, M. and Akutsu, T. (2007). Simulation study in probabilistic Boolean network models for genetic regulatory networks. Journal of Data Mining and Bioinformatics, 1, 217-40.

[19] Zhang, S., Hayashida, M., Akutsu, T., Ching, W. and Ng, M. (2007). Algorithms for finding small attractors in Boolean networks. EURASIP Journal on Bioinformatics and Systems Biology, 20180 (13 pages).