Finding Optimal Control Policy by Using Dynamic Programming in Conjunction with State Reduction

Xi CHEN
AMAC Laboratory, Department of Mathematics,
The University of Hong Kong
Hong Kong, China
Email: dlkcissy@hku.hk

Wai-Ki CHING
AMAC Laboratory, Department of Mathematics,
The University of Hong Kong
Hong Kong, China
Email: wching@hku.hk

***Abstract*—In this paper we study the problem of finding optimal control policy for probabilistic Boolean networks (PBNs). Previous works have been done by using dynamic programming-based (DP) method. However, due to the high computational complexity of PBNs, DP method is computationally inefficient for large networks. Inspired by the state reduction strategies studied in [10], we consider using dynamic programming in conjunction with state reduction approach to reduce the computational cost of DP method. Numerical examples are given to demonstrate the efficiency of our proposed method.**

I. Introduction

An important goal for studying the behavior of genes is to develop control policy for the potential application to medical therapy. While many models have been proposed for modeling gene regulatory networks, Boolean Networks (BNs) [9] and its extension Probabilistic Boolean Networks (PBNs) [12] have received much attention. For reviews on BNs and PBNs, interested readers can consult [2], [4], [11]. In fact, many methods in control theory are available for the intervention of PBNs. Datta et al. [7] proposed an external intervention method based on optimal control theory. In their work, genes are classified as internal nodes and external nodes (control nodes). One can intervene the values of internal nodes in some desirable manner by controlling the values of certain external nodes. By defining the control cost for each control input and terminal cost for each state, the problem is to find a sequence of control inputs that leads the network into desirable states at the terminal step with minimum average cost. The classical techniques of dynamic programming is then employed to solve the optimization problem. Later, Chen et al. [5] consider an external intervention problem based on optimal control theory and dynamic programming. Given the terminal cost of each state, the objective is to derive the network into the state with the maximum cost being minimized by applying external controls. The problem is important in the view of medical therapy because patients/organisms would like to minimize the damage even for the worst case. They proved that both minimizing the maximum cost and minimizing the average cost are Σ_2^p-hard. A dynamic programming-based algorithm is then proposed for finding a control sequence that minimize the maximum cost in control of PBN. The above dynamic programming-based methods have high computational complexity. One has to deal with the matrices with size increasing exponentially with the number of nodes in PBN. Hence one possible way out is to consider network reduction approach.

Recently several reduction methods have been proposed. In [8], a CoD-based reduction algorithm is introduced. Coefficient of Determination (CoD) helps to evaluate the influence of a candidate node for deletion on the target node and find the optimal candidate node for deletion. The proposed algorithm can well preserve the attractor structure and long-run dynamics of the original network. Qian et al. [10] proposed a reduction method by considering deleting states directly. Instead of deleting nodes, they choose to delete the out-most states which have less influence on the network. Here we consider a transition probability-based reduction strategy. The strategy is easier to be carried out as we do not need the stationary distribution information of PBNs beforehand.

We consider the problem of minimizing the maximum cost in control of PBN and we use transition probability-based reduction strategy to reduce the network complexity of PBN. We show that the result of finding optimal control sequence on the reduced network is the same as the one on the original network. Then we apply dynamic programming-based algorithm on the reduced network. Since the computational complexity of dynamic programming-based algorithm on the original network is $O(2^n)$ (depending on the number of network states) for fixed number of control nodes m and fixed number of steps M, using state reduction may reduce the computation complexity to $O(|R|)$, where R is the set of states after reduction.

The remainder of the paper is structured as follows. In Section 2, we briefly review basic definitions of PBNs. Section 3 gives a short introduction to dynamic programming-based algorithm proposed in [5]. In Section 4, a brief review on transition probability-based reduction strategy [10] will be given. Then we apply dynamic programming-based algorithm on the reduced network. Section 5 gives numerical experiments to compare the results of the proposed method with results given by the algorithm in [5]. Finally, concluding remarks are given to discuss further research issues.

2011 IEEE International Conference on Systems Biology (ISB)
978-1-4577-1666-9/11/$26.00 ©2011 IEEE

II. A Brief Review on BNs and PBNs

A BN consists of a set of n nodes (genes) as follows:

$$\{v_1, v_2, \ldots, v_n\}, \quad v_i \in \{0, 1\},$$

and a set of Boolean functions denoted by

$$\{f_1, f_2, \ldots, f_n\}.$$

Each $v_i(t)$ is defined as the state of node i at time t. The rules of regulatory interactions among nodes is then represented by the Boolean functions:

$$v_i(t+1) = f_i(v_{i1}, v_{i2}, \ldots, v_{ik})$$

where $\{v_{i1}, v_{i2}, \ldots, v_{ik}\}$ are input nodes of f_i, and they are called parent nodes of node v_i. We define

$$IN(v_i) = \{v_{i1}, v_{i2}, \ldots, v_{ik}\}.$$

The number of parent nodes to v_i is called the *in-degree* of v_i. The largest in-degree of $\{v_1, v_2, \ldots, v_n\}$ is called the *maximum in-degree* of BN and is denoted by K.

Since BN is a deterministic model, considering the high complexity nature of gene regulatory networks, a stochastic model is more preferable. Thus PBN is then introduced. A PBN can be regarded as an extension of BN to a probabilistic setting. In a PBN, each node v_i has a set of Boolean functions:

$$\{f_1^{(i)}, f_2^{(i)}, \ldots, f_{l(i)}^{(i)}\}. \tag{1}$$

The state of v_i at time $t+1$ is predicted by one of Boolean functions in (1) with selection probabilities $c_j^{(i)}$. Here

$$\sum_{j=1}^{l(i)} c_j^{(i)} = 1, \quad c_j^{(i)} \geq 0 \text{ for } j = 1, 2, \ldots, l(i).$$

Moreover, a PBN can be regarded as a finite collection of BNs over a fixed set of nodes, where each BN has a fixed set of Boolean functions $\mathbf{f_j} = \{f_{j_1}^{(1)}, f_{j_2}^{(2)}, \ldots, f_{j_n}^{(n)}\}$. The BN having Boolean function set $\mathbf{f_j}$ $(j = 1, 2, \ldots, N)$ is called the jth BN. At each time step t, one of BNs is selected with probability

$$q_j = c_{j_1}^{(1)} c_{j_2}^{(2)} \cdots c_{j_n}^{(n)}, \quad j = 1, 2, \ldots, N,$$

and the states of $\{v_1(t+1), v_2(t+1), \ldots, v_n(t+1)\}$ is predicted by the Boolean function set $\mathbf{f_j}$. Then we introduce the decimal representation of states. Suppose the current state is $\{v_1(t), v_2(t), \ldots, v_n(t)\}$, we define

$$w(t) = 1 + \sum_{i=1}^{n} 2^{n-i} v_i(t).$$

Since $\{v_1(t), v_2(t), \ldots, v_n(t)\}$ ranges from $\{0, 0, \ldots, 0\}$ to $\{1, 1, \ldots, 1\}$, $w(t)$ has a range $[1, 2^n]$.

The dynamics of a PBN can be studied by using Markov chain theory, see for instance [6]. The one-step transition can be described by the transition matrix A where each entry A_{ij} is given by

$$A_{ij} = \sum_{k \in \mathbb{I}} q_k, \quad i, j = 1, 2, \ldots, 2^n. \tag{2}$$

Here $i = w(t+1)$ and $j = w(t)$ and $\mathbb{I}$ is set of BNs that the network can enter state i from state j. We remark that A is a column stochastic matrix.

III. Review on Dynamic Programming

In this section to facilitate the discussion, we first introduce several definitions, we then introduce the dynamic programming-based algorithm. Suppose a PBN has a set of internal nodes $\{v_1, v_2, \ldots, v_n\}$ which is the same as node set defined in Section 2, and a set of external nodes (control nodes) $\{v_{n+1}, v_{n+2}, \ldots, v_{n+m}\}$. At time $t+1$, the states of $v_i, i = 1, 2, \ldots, n$ is predicted by

$$v_i(t+1) = f_j^{(i)}(v_{i1}, v_{i2}, \ldots, v_{ik}),$$

where v_{ik} can be either an internal node or an external node. This provides a possible way for intervening the states of internal nodes by controlling the values of external nodes.

To simplify our presentation, we adopt the decimal representation of states and define

$$z_t = 1 + \sum_{i=1}^{n} 2^{n-i} v_i(t)$$

as the state of network. Then we define control input as

$$u_t = 1 + \sum_{i=1}^{m} 2^{m-i} v_{n+i}(t).$$

We are interested in the following problem: **Minimizing the maximum cost in control of PBN**.

Given the terminal cost $C(z_M)$ for each state $z_M \in \{1, 2, \ldots, 2^n\}$ at terminal time step M, find a sequence of control input $u_0, u_1, \ldots, u_M$ such that starting from the given initial state the network will enter into the state with minimized maximum cost at time step M. In [5], a dynamic programming-based method is proposed for the above problem:

Step 0: Set $t = M$;
$J(z_M, h_M) = C(z_M)$ for all $h_M = \{0, \ldots, M\}$.
Step 1: $t := t - 1$.
Step 2: For any $z_t \in \{1, \ldots, 2^n\}$ and $h_t \in \{0, \ldots, M\}$, compute

$$J(z_t, h_t) = \min_{u_t \in \{1, \ldots, 2^m\}} \begin{cases} \max\limits_{z_{t+1} \in F(z_t, u_t)} J(z_{t+1}, h_t), \text{ if } u_t = u(z_{t+1}, h_t), \\ \max\limits_{z_{t+1} \in F(z_t, u_t)} J(z_{t+1}, h_t - 1), \text{ otherwise.} \end{cases}$$

and

$$u(z_t, h_t) = \text{argmin}_{u_t \in \{1, \ldots, 2^m\}} \begin{cases} \max\limits_{z_{t+1} \in F(z_t, u_t)} J(z_{t+1}, h_t), \text{ if } u_t = u(z_{t+1}, h_t), \\ \max\limits_{z_{t+1} \in F(z_t, u_t)} J(z_{t+1}, h_t - 1), \text{ otherwise.} \end{cases}$$

Step 3: If $t > 0$, go back to **Step 1**; Otherwise, stop.

978-1-4577-1666-9/11/$26.00 ©2011 IEEE

IV. Dynamic Programming-based Algorithm on the Reduced Network

Due to the high network complexity of a PBN, if the algorithm in Section 3 is applied, one has to deal with matrices of huge size which increases exponentially with the number of internal nodes. Hence network reduction is an important issue to be addressed in this situation. In [10], a transition probability-based state reduction strategy is proposed. In a PBN, we consider all attractor states and initial state as critical states, and they are preserved during state reduction. A state i can be deleted if the following equation is satisfied:

$$\sum_{j=1}^{2^n} A_{ij} < \xi \tag{3}$$

where $\xi > 0$ is a parameter which needs to be predetermined. The value of ξ depends on perturbation probability and it is usually very small. When Equation (3) is satisfied, the network will never enter state i from other states unless perturbation happens. When we consider PBNs without perturbation, Equation (3) can be rewritten as

$$\sum_{j=1}^{2^n} A_{ij} = 0. \tag{4}$$

Which means that the network will never enter state i from other states. Hence, deleting state i will not influence the steady-state distribution of the network. Since the computational complexity of dynamic programming is $O(2^n)$ for fixed number of control nodes m and fixed number of steps M, using state reduction may reduce the computation complexity to $O(|R|)$, where R is the set of states after reduction. It is straightforward to see that we have the following theorem.

Theorem 1: the result of dynamic programming-based algorithm on the reduced network will be the same as the one on the original network.

Proof: It is straightforward to see that, starting from the initial state, the network will never enter into transient states for deletion, and therefore the network will never stop at those states at the terminal time step. That means the deleted states will not be included in the optimal route, and the cost of deleted states will not be counted. Hence deleting these transient states will not influence the result of DP method on the reduced network. ■

Based on transition probability-based strategy, one can iteratively delete those transient states until all the remaining states are critical states. For each step, we need to update the transition matrix for the reduced network by deleting the corresponding row and column from the transition matrix. After reduction, we can get a reduced network with set of states R and $|R|$ by $|R|$ transition matrix B. Then we can apply dynamic programming-based algorithm on the reduced network.

V. Numerical Experiments

In this section, we give some numerical examples to compare the results of dynamic programming-based algorithm on the reduced network with the one on the original network.

A. A 6-Gene Example

We first consider a 6-node example. We consider the cases of $m = 1, 2$, $N = 2, 4, 8$ and $K = 2, 3$. The Boolean function set of PBN are randomly generated. We let $M = 20$ and $C(z_M) = z_M$. When $m = 1$, there are 5 internal nodes and 1 control node. The original network size is 2^5. When $m = 2$, there are 4 internal nodes and 2 control nodes. The original network size is 2^4. Table 1 gives the numerical results of this example. The second column gives the network size before and after reduction. The third column gives minimized maximum cost obtained by using the dynamic programming-based algorithm on the original and reduced network. The last column records the CPU time of running the program for dynamic programming-based algorithm before and after reduction.

B. A 12-Gene Example

We then consider a 12-node example. We consider the cases of $m = 1, 2$, $N = 2, 4, 8$ and $K = 2, 3$. Again the Boolean function set of PBN are randomly generated. We let $M = 40, C(z_M) = z_M$. When $m = 1$, there are 11 internal nodes and 1 control node. The original network size is 2^{11}. When $m = 2$, there are 10 internal nodes and 2 control nodes. The original network size is 2^{10}. Table 2 gives the corresponding results. We see that our proposed reduction method is both efficient and effective.

VI. Conclusion

From the experiment results, one can see that applying dynamic programming-based algorithm on the reduced network can reduce the computational complexity and save time. The performance of the algorithm on the reduced network depends on the parameters of n, m, N and K. When the number of nodes is large and $K = 2$, the algorithm on the reduced network performs much better than the one on the original network. Future research issues will pay attention to statistic analysis of the distribution of zero rows in transition matrix in terms of n. Moreover, we will keep exploring ways of reducing computational complexity of intervention strategies.

Acknowledgment

Research support in part HKU strategic theme grant on computational sciences, National Natural Science Foundation of China Grant No. 10971075 and Guangdong Provincial Natural Science Grant No. 9151063101000021.

References

[1] T. Akutsu, M. Hayashida and T. Tumura, *Integer Programming-Based Methods for Attractor Detection and Control of Boolean Networks*, in Proc. the Combined 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference, (2009) 5610-5617.

[2] T. Akutsu, M. Hayashida and T. Tamura, *Algorithms for Inference, Analysis and Control of Boolean Networks*, Proc. 3rd International Conference on Algebraic Biology (AB 2008), Lecture Notes in Computer Science, 5147 (2008) 1-15.

978-1-4577-1666-9/11/$26.00 ©2011 IEEE

	Size		Cost		CPU Time (sec.)	
	Original	Reduced	Original	Reduced	Original	Reduced
$m=1$ $N=2$ $K=2$	32	24	17	17	0.127563	0.094261
$m=1$ $N=2$ $K=3$	32	26	5	5	0.125479	0.099712
$m=1$ $N=4$ $K=2$	32	26	21	21	0.128609	0.106157
$m=1$ $N=4$ $K=3$	32	27	19	19	0.129054	0.108553
$m=1$ $N=8$ $K=2$	32	30	25	25	0.135445	0.127391
$m=1$ $N=8$ $K=3$	32	32	29	29	0.135501	0.135597
$m=2$ $N=2$ $K=2$	16	11	9	9	0.106091	0.069203
$m=2$ $N=2$ $K=3$	16	10	3	3	0.099579	0.05853
$m=2$ $N=4$ $K=2$	16	14	9	9	0.105124	0.099579
$m=2$ $N=4$ $K=3$	16	16	6	6	0.103975	0.099719
$m=2$ $N=8$ $K=2$	16	16	14	14	0.106254	0.105018
$m=2$ $N=8$ $K=3$	16	16	12	12	0.104854	0.105317

TABLE I
A 6-NODE EXAMPLE.

[3] T. Akutsu, M. Hayashida, W. Ching and M. Ng, *Control of Boolean Networks: Hardness Results and Algorithms for Tree Structured Networks*, Journal of Theoretical Biology, 244 (2007) 670-679.

[4] S. Bornholdt, *Boolean Network Models of Cellular Regulation: Prospects and Limitations*, J. R. Soc. Interface, 5 (2008) 85-94.

[5] X. Chen, T. Akutsu, T. Tamura and W. Ching, *Finding Optimal Control Policy in Probabilistic Boolean Networks with Hard Constraints by Using Integer Programming and Dynamic Programming*, to appear in International Journal of Data Mining and Bioinformatics, 2011.

[6] W. Ching, S. Zhang, M. Ng and T. Akutsu, *An Approximation Method for Solving the Steady-state Probability Distribution of Probabilistic Boolean Networks*, Bioinformatics, 23 (2007) 1511-1518.

[7] A. Datta, R. Pal, A. Choudhary and E.R. Dougherty, *Control Approaches for Probabilistic Gene Regulatory Networks*, IEEE Signal Processing Magazine, 24 (2007) 54-63.

[8] N. Ghaffari, I. Ivanov, X. Qian and E.R. Dougherty, *A CoD-based Reduction Algorithm for Designing Stationary Control Policies on Boolean Networks* Bioinformatics, 26 (2010) 1556-1563.

[9] S. Kauffman, *Metabolic Stability and Epigenesis in Randomly Constructed Genetic Nets*, Journal of Theoretical Biology, 22 (1969) 437-467.

[10] X. Qian, N. Ghaffari, I. Ivanov and E.R. Dougherty, *State Reduction for Network Intervention in Probabilistic Boolean Networks* Bioinformatics, 26 (2010) 3098-3104.

[11] I. Shmulevich, E. Dougherty, S. Kim and W. Zhang, *Probabilistic Boolean Networks: A Rule-based Uncertainty Model for Gene Regulatory Networks*, Bioinformatics, 18 (2002) 261-274.

[12] I. Shmulevich, *Probabilistic Boolean Networks : the Modeling and Control of Gene Regulatory Networks*, Philadelphia : Society for Industrial and Applied Mathematics, (2010).

978-1-4577-1666-9/11/$26.00 ©2011 IEEE

	Size		Cost		CPU Time (sec.)	
	Original	Reduced	Original	Reduced	Original	Reduced
$m = 1$ $N = 2$ $K = 2$	2048	426	757	757	63.839384	4.069782
$m = 1$ $N = 2$ $K = 3$	2048	700	1258	1258	256.685488	36.892619
$m = 1$ $N = 4$ $K = 2$	2048	502	1830	1830	272.268905	23.274458
$m = 1$ $N = 4$ $K = 3$	2048	1462	1591	1591	264.38678	143.477244
$m = 1$ $N = 8$ $K = 2$	2048	1103	2036	2036	279.579186	88.793874
$m = 1$ $N = 8$ $K = 3$	2048	1801	1987	1987	272.351139	243.788024
$m = 2$ $N = 2$ $K = 2$	1024	350	607	607	153.849845	24.354139
$m = 2$ $N = 2$ $K = 3$	1024	444	179	179	148.797692	36.103486
$m = 2$ $N = 4$ $K = 2$	1024	415	342	342	140.623945	29.489216
$m = 2$ $N = 4$ $K = 3$	1024	801	328	328	150.20795	107.27338
$m = 2$ $N = 8$ $K = 2$	1024	759	736	736	146.975119	87.758299
$m = 2$ $N = 8$ $K = 3$	1024	937	756	756	172.926666	146.290486

TABLE II
A 12-NODE EXAMPLE.

978-1-4577-1666-9/11/$26.00 ©2011 IEEE