Large-Scale Approximate Intervention Strategies for Probabilistic Boolean Networks as Models of Gene Regulation

Mehmet Tan Reda Alhajj Faruk Polat

***Abstract*— Control of Probabilistic Boolean Networks as models of gene regulation is an important problem; the solution may help researchers in various different areas. But as generally applies to control problems, the size of the state space in gene regulatory networks is too large to be considered for comprehensive solution to the problem; this is evident from the work done in the field, where only very small portions of the whole genome of an organism could be used in control applications. The Factored Markov Decision Problem (FMDP) framework avoids enumerating the whole state space by representing the probability distribution of state transitions using compact models like dynamic bayesian networks. In this paper, we successfully applied FMDP to gene regulatory network control, and proposed a model minimization method that helps finding better approximate policies by using existing FMDP solvers. The results reported on gene expression data demonstrate the applicability and effectiveness of the proposed approach.**

I. Introduction

Devising control strategies for gene regulatory networks (GRNs) is important to avoid undesirable gene activity profiles. A control or intervention strategy for GRN can be defined as a way to interact with the network in terms of some actions to reach some pre-defined objective(s). These interventions (or actions) are usually defined in terms of (in)activation of certain types of genes or proteins; the objective is to reach (or avoid) a set of state(s) (or gene activity profiles). For instance, Gefitinib is a drug used in the treatment of a type of lung cancer and inhibits (inactivates) the epidermal growth factor receptor (EGFR) tyrosine kinase enzyme, which leads to the cease of uncontrolled cell proliferation of malignant cells. Without this inactivation of EGFR, the cells may continue to divide beyond normal limits.

Probabilistic Boolean Networks (PBNs) is one promising method to model gene regulations [22], [23]. Finding intervention strategies has also been studied on PBNs [5], [19], [18]. Under some assumptions, a PBN can be equivalently represented by a corresponding Markov chain [22]; this brings the opportunity to use the Markov Decision Problem (MDP) framework for the control problem [1], [8]. The most well-known issue in solving MDP is the curse of dimensionality. This refers to the exponential growth of the state space in terms of the number of variables to represent the problem. In our case, the variables and states correspond to genes and gene activity profiles, respectively. Even for representing gene expression values in the simplest way by only *ON* and *OFF* (Boolean) values as in the case of PBNs, the size of the state space is proportional to 2^N for N genes.

A Factored MDP (denoted FMDP) [4] is a promising alternative to represent the control problem of GRNs since GRNs are naturally factorized and the genes correspond to the factors. In this framework, the structure existing in the problem is exploited by representing the transition probabilities in terms of factored models like dynamic bayesian networks (see Section III-B). But for some of the cases, factored representations also require exponential space [4].

Our contributions in this paper are two-fold. First, we apply the FMDP framework to PBN control problem. To the best of our knowledge, these are the first results for large-scale control in GRN domain. Second, we propose a pre-processing method for a given PBN based on the influence of genes on others [22] that minimize the PBN for even further scalability in approximately solving control problems. The results presented here are promising steps through genome-wide solutions for the control problem of GRNs.

The rest of the paper is organized as follows. Section II discusses PBNs and the influence concept. Section III formally defines the control problem for PBNs and discusses the existing solutions. Section IV introduces the edge elimination algorithm for approximate solutions. Section V reports experimental results. Section VI is conclusions and future work.

II. Probabilistic Boolean Networks

Boolean Networks (BNs) [13] represent gene expression by using only two levels: *ON* and *OFF*. The expression level for a gene g_i at time step $t+1$ is related to the expression level of k_i other genes at time t by a boolean function, $f^{(i)}(g_{i_1}, ..., g_{i_k})$, where genes g_{i_1} to g_{i_k} are called *parents* of g_i. So, BN is defined by a set of genes $V = (g_1, ..., g_n)$ and a set of boolean functions $F = (f^{(1)}, ..., f^{(n)})$. On the other hand, PBN is defined by a set of genes $V = (g_1, ..., g_n)$ and a set $F = (F_1, ..., F_n)$, where each F_i is a set of functions for g_i, $F_i = \{f_j^{(i)}\}_{j=1,...,l_i}$. Each $f_j^{(i)}$ is one of the possible functions to determine the next state of g_i, and l_i is the number of such functions. The probability of choosing $f_j^{(i)}$ in F_i to predict the next state of g_i is denoted $c_j^{(i)}$.

Given binary quantized gene expression data, deriving a PBN model requires finding F and $c_j^{(i)}$ for all i and j. To do this, a measure of how well a function predicts the value of

M. Tan is with the Department of Computer Engineering, Middle East Technical University, Ankara, Turkey, mtan@ceng.metu.edu.tr, he is visiting scholar at the Department of Computer Science, University of Calgary, Calgary, Alberta, Canada; mtan@ucalgary.ca, his research is partially supported by The Scientific and Technological Research Council of Turkey

R. Alhajj is with the Department of Computer Science, University of Calgary, Calgary, Alberta, Canada, he is also affiliated with the Department of Computer Science, Global University, Beirut, Lebanon, alhajj@ucalgary.ca

F. Polat is with the Department of Computer Engineering, Middle East Technical University, Ankara, Turkey, polat@ceng.metu.edu.tr

a gene is needed. Coefficient Of Determination (COD) [7] is one such measure. COD compares the prediction performance of a function with the best constant estimator in the absence of other information. Assume that we are given the parents P_i of g_i and a function $f_j^{(i)}(P_i)$ to predict g_i. The COD θ_j^i of $f_j^{(i)}$ is defined as follows: $\theta_j^i = \frac{\varepsilon_i - \varepsilon(g_i, f_j^{(i)}(P_i))}{\varepsilon_i}$, where ε_i is the error of the best constant estimate of g_i and $\varepsilon(g_i, f_j^{(i)}(P_i))$ is a probabilistic error measure [23]. Given θ_j^i values, it is straightforward to define $c_j^{(i)}$ [23]: $c_j^{(i)} = \frac{\theta_j^i}{\sum_{m=1}^{l_i} \theta_m^i}$.

For a given set of parent genes P_i, $f_j^{(i)}$ can be derived using various methods. In this paper, we use best-fit extension paradigm of Lähdesmäki *et al.* [16]. Briefly, this method outputs $f_j^{(i)}(P_i)$ that best predicts g_i, where the error measure is the number of mis-predicted values of g_i. We use publicly available Matlab implementation of Best-Fit Extension in PBN-Toolbox[1].

A. Influence of genes

Given gene g_i and its parent genes P_i, Shmulevich *et al.* [23], formalized *Influence* to measure the effect of a parent on g_i. In other words, influence of g_i on g_j is the probability that the next state value of g_j will change when we change the value of g_i at the current time step. To formally define the influence of g_i on g_j, denoted $I_i(g_j)$, first we have to define the influence of a gene wrt a boolean function f, which is the probability that the output of f will change if we change g_i. Assume that f is defined on the set of input genes $P = (g_1, ..., g_k)$. The influence $I_j(f)$ of g_j on f is defined as;

$$I_j(f) = Pr\{f(g_1, ..g_{j-1}, 0, g_{j+1}, .., g_k) \oplus f(g_1, ..g_{j-1}, 1, g_{j+1}, .., g_k)\} \quad (1)$$

where $\oplus$ stands for exclusive *OR*. Eqn 1 depicts the probability that f will output a different value if g_j is toggled while the other input variables stay the same. Also note that $I_j(f) = 0$ if $g_j \notin P$.

Given V, F and $c_j^{(i)}$ of a PBN, $I_i(g_j)$ is defined as follows [23]: $I_i(g_j) = \sum_{k=1}^{l_j} I_i(f_k^{(j)}) c_k^{(j)}$, which is the weighted sum of all influences of g_i on the set F_j.

III. PBN Control

Given a PBN, in addition to simulating the model, intervening the evolution of the PBN for the objective of reaching or avoiding some predefined set of states from the current state is an important problem as well. But this should be done as effective as possible since it incurs some cost. This defines the control problem in PBNs as: *given a set of control actions and their costs, find a sequence of actions for the given PBN to achieve the predefined objective(s)*. Markov Decision Processes [20] form one of the most widely used frameworks to formulate control problems in the operations research and artificial intelligence communities.

[1]available at: http://personal.systemsbiology.net/ilya/PBN/PBN.htm

A. Markov Decision Problems

A Markov Decision Process is formally defined as a quadruple (S, A, T, R), where S is the set of states, A is the set of actions, T is the transition probability function such that $T(s, a, s')$ denotes the probability of the next state being s' given the current state s and action a, and R is the reward function that represents the objective of the control process. A MDP is a Markov Decision Process associated with a performance criterion. The performance criterion we adapt in this paper is the infinite horizon discounted reward criterion. So the objective is to maximize the total discounted reward: $\sum_t \beta_t R_t(s, a)$, where $R_t(s, a)$ is the immediate reward of performing action a in state s at time t, and $\beta \in (0, 1)$ is the discount factor. In this paper, we assume that R_t and β are independent of t; so we omit t after this point.

Solution to an MDP is called a policy, π; it is a mapping from states S to actions A. Every π defines a value function V^π from S to real numbers. $V^\pi(s)$ is the total discounted future reward of choosing an action a according to π in state s, and following π thereafter. V^π can be found iteratively:

$$V_{k+1}^\pi(s) = R(s, \pi(s)) + \beta \sum_{s'} T(s, \pi(s), s') V_k^\pi(s') \quad (2)$$

where iteratively applying Eqn 2 is called policy evaluation.

Optimal policy π^* is the best policy in terms of the given performance criterion. In our case, it is the policy that achieves maximum possible infinite horizon discounted future reward. Value function corresponding to π^* is the optimal value function, V^*, which can also be found iteratively using the following Bellman update:

$$V_{k+1}(s) = max_a [R(s, a) + \beta \sum_{s'} T(s, a, s') V_k(s')] \quad (3)$$

Given all components of an MDP, Eqn 3 converges to the unique V^* as $k \to \infty$. From V^*, π^* can be found as:

$$\pi^*(s) = argmax_a [R(s, a) + \beta \sum_{s'} T(s, a, s') V^*(s')] \quad (4)$$

With arbitrary initialization to V_0, the algorithm that uses Eqn 3 to find V^* is called value iteration [2]. One simple stopping criterion for value iteration is:

$$||V_{k+1} - V_k|| \leq \frac{\epsilon(1 - \beta)}{2\beta} \quad (5)$$

where $||X|| = max\{|x| : x \in X\}$ denotes maximum norm. Eqn 5 ensures V_{k+1} is within $\epsilon/2$ of V^* for any state [20].

Another well-known algorithm for solving an MDP is the policy iteration algorithm [20]. Instead of starting with arbitrary V, policy iteration starts with an arbitrary policy π, and finds V^π using Eqn 2. Then for all states s, it searches for an action a that satisfies the following equation:

$$V^\pi(s) < R(s, a) + \beta \sum_{s'} T(s, a, s') V^\pi(s') \quad (6)$$

If found, it updates $\pi(s)$=a, and repeats the policy evaluation and update steps until convergence criterion is met.

B. Factored MDPs

A factored MDP (FMDP) [3] is a representation language for MDPs to exploit the structure of the control problem. In most problems, T can be represented in terms of a set of state variables, where in our case these variables correspond to genes.

As representing T for a MDP requires exponential space in the number of variables, FMDP proposes to represent T for each specific action in the form of a dynamic bayesian network (DBN) [6]. A DBN is composed of variables $G = (g_1, g_2, ..., g_n, g'_1, g'_2, ...g'_n)$, where the variables with a prime denote the random variables at the next time step. So, a DBN represents the relationships between random variables in the current and next time steps. We denote the set of primed variables by X' and non-primed by X, where $G = X \cup X'$. Each variable g'_i has a set of parents P_i, where the value of g'_i depends only on P_i. In this paper, we assume that $P_i \subset X$, and the variables in X do not have any parents, i.e., there are no synchronous dependencies between variables, all dependencies are between the variables at time step t and the variables at time step $t+1$. This is a common assumption for modeling GRNs using a DBN.

A DBN associates to each g'_i and its parents P_i a conditional probability distribution (CPD). A discrete CPD is usually represented as a table. But some space can be gained if CPDs are represented by decision trees in case they have the same values for different instantiations of the parents [4].

In addition to CPDs, the structure in V and π can also be exploited to represent them by decision trees. Both value trees and policy trees have internal nodes labeled with the variables themselves and edges labeled with the values (instantiations) of the variables. Leaf nodes of a value tree have values of the states corresponding to all states that have the same instantiations of the variables in the path from the root to the leaf. The same way, leaf nodes of a policy tree have the actions corresponding to the states that have the same instantiations of the variables in the path from the root to the leaf. The reader is referred to [4] for details.

Solving FMDP requires modifying these value and policy trees at each iteration. Decision-Theoretic Regression [4] is one of the methods to modify decision tree representations of value and policy trees; each iteration results in a new value or policy tree that is closer to the decision tree representation for V^*. Structured value and policy iteration are two algorithms that use decision-theoretic regression to solve FMDPs [4]. We use the publicly available FMDP solver, SPUDD[2] ("Stochastic Planning using Decision Diagrams") [10]. Instead of using decision trees, SPUDD uses algebraic decision diagrams (ADD) [21]. The SPUDD package also includes an approximate FMDP solver, APRICODD ("Approximate Policy Construction using Decision Diagrams") [24].

In terms of GRNs, given PBN model derived from some kind of biological data, actions, and the objective defined in terms of the reward genes, the PBN control problem can be solved by the following steps: 1) Convert PBN to DBN; 2) For each action $a \in A$, construct DBN_a that represents probability distribution $T(s, a, s')$ for all s, s'; 3) Given reward function R and discount factor β, define FMDP M; 4) Solve M using SPUDD.

[2]available at : http://www.computing.dundee.ac.uk/staff/jessehoey/spudd/

C. PBN to DBN conversion

The probability distribution that a PBN represents can be equally represented by a corresponding DBN [15]. Let cpd_{P_i,i_v} denote the entry in a discrete CPD of a DBN corresponding to the probability that g'_i will take the value v at time step $t+1$, given an instantiation vector P_i of its parents at time t. Also, let $f_j^{(i)}(P_i) == v$ denote whether the output of $f_j^{(i)}$ for the input P_i is v or not, i.e., 1 if v, 0 otherwise. Given parameters $F_i = \{f_1^{(i)}, f_2^{(i)}, ..., f_{l_i}^{(i)}\}$ and $c_j^{(i)}$ for g_i in a PBN, cpd_{P_i,i_v} can be computed as:

$$cpd_{P_i,i_v} = \sum_{j=1}^{l_i} (f_j^{(i)}(P_i) == v) c_j^{(i)} \tag{7}$$

Performing these steps for each $g \in G$ of the PBN, a DBN representing the same probability distribution is constructed.

To find DBN_a for each a, some simple updates on CPTs have to be performed. Since the actions are defined as interventions (toggling the action gene), to get the CPTs for $DBN_{a_{g_i}}$, for all CPTs in the DBN that have g_i as a parent, rows that differ only in the value of g_i are switched.

IV. Edge elimination from factored representations

Although factored representations help in solving some of the problems, they still suffer from the curse of dimensionality in the worst case. Fortunately, in most of these cases we can still reach a reasonable approximate solution by pruning and/or approximating the value tree. Most of the approximate methods prune the constructed trees during the process of solving FMDP. Another possibility in finding an approximate result is to prune the transition model before solving the problem. In this section, we elaborate on such a method, but before that we introduce the concept of *edge influence*.

A. Edge Influence

Lets start by introducing the basic concepts required to understand edge influence as in Definition 1. Given a PBN, influence of a gene g_i on gene g_j, $I_i(g_j)$, can be interpreted as a measure of the strength of the link between the two genes. But, $I_i(g_j)$ will be zero if g_i is not among the parents of g_j. However, this does not mean g_i has no influence on g_j.

Given a node g_i as the root, an "unrolled" PBN is constructed (as a tree) by expanding each node g at level t with the parents of g at level $t-1$ in the given PBN. Nodes are expanded unless the unique path from the leaf node to g_i includes a cycle.

Recall that each $I_i(g_j)$ corresponds to the effect of the value of g_i at time step t on the value of g_j at time step $t+1$, and assume g_k is one of the parents of g_i. When we unroll the PBN one time step, we will observe the path

$g_k \rightarrow g_i \rightarrow g_j$. We know that, based on Markovian property, $I_k(g_i)$ is independent of $I_i(g_j)$. Therefore, to compute the influence of g_k on g_j after two time steps, we simply have to multiply $I_k(g_i)$ and $I_i(g_j)$.

Formally, consider a simple path (a path that does not have any cycles) $p = g_{i_1}, g_{i_2}, ..., g_{i_k}$ between g_{i_1} and g_{i_k} in an unrolled PBN; if we label each edge $E_{g_{i_l} g_{i_{l+1}}}$ on the path with $I_{i_l}(g_{i_{l+1}})$, we can define the influence of g_{i_1} on g_{i_k} corresponding to path p, denoted $I^{(p)}_{i_1}(g_{i_k})$, as:

$$I^{(p)}_{i_1}(g_{i_k}) = \prod_{l=k}^{2} I_{i_{l-1}}(g_{i_l}).$$

Note that there can be more than one simple path from one gene to another in an unrolled PBN. Let P be the set of all simple paths from g_i to g_j in an unrolled PBN. We define *all-path influence* of g_i on g_j, denoted $I^{(*)}_i(g_j)$, as:

$$I^{(*)}_i(g_j) = \begin{cases} \sum_{p \in P} I^{(p)}_i(g_j) & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \quad (8)$$

Definition 1 (Edge Influence): Given three genes g_i, g_j and g_k, Edge Influence (EI) of the edge between g_i and g_j on g_k is defined as: $EI_{i,j}(g_k) = I_i(g_j) I^{(*)}_j(g_k)$.
EI can also be computed on a set of genes, denoted EIS: $EIS_{i,j}(S) = \sum_{g_k \in S} EI_{i,j}(g_k)$, which is simply the sum of influences of an edge on all genes in the given set.

Considering nodes with same label as different nodes, the unrolled PBN corresponds to a tree. By considering them as the same node and aggregating them, we find a graph; then, path search in a tree turns into path search in a graph. Computing all paths between two nodes in a graph is a hard problem. It is NP-complete as it includes the solution of the longest path problem which is known to be NP-complete [12]. Also, the size of the unrolled PBN tree can grow exponentially large depending on the structure of PBN and the number of genes. So, it is better to compute approximate values for EI. One possible method is to prune the unrolled PBN tree. Also notice that the unrolled PBN tree for a given gene only includes relevant genes and edges. All nodes in a tree have an influence on the given gene, so the parts of the PBN that are not related to the solution of the control problem are not expanded and the EI values for those edges are not computed.

1) Approximate computation of EI: Limiting the size of the unrolled PBN up to a certain level can give good results. But a better method is to prune the unrolled PBN if $I^{(*)}_i(g_j)$ is less than a certain threshold T. As $I_i(g_j)$ is actually a probability value, $I^{(p)}_i(g_j)$ for any i, j and p monotonically decreases with each new level in the unrolled PBN. So, when we consider that the current all path influence values below T are not significant then we may stop expanding a node i further down in case $EI_{i,j}(g_k) \leq T$, where g_k is the root.

After this approximation, we are ready to construct an approximate algorithm for computing EI values for a given gene g_j which will be the root of the unrolled PBN tree. The complete process is given in Algorithm 1. It is a recursive algorithm that actually does a limited depth first traversal of the unrolled PBN tree, and does not expand node i for sufficiently small values of $I^{(*)}_i(g_j)$.

Algorithm 1 $computeEI(g, p, I^{(p)}_g(g_t), T, g_t, pbn, EI)$

Input: gene g, path $p = g_t, p_{i_1}, .., p_{i_k}$, path influence $I^{(p)}_g(g_t)$, target gene g_t, PBN pbn, initial values of EI
Output: $\forall g_i, g_j$ reachable from g_t, $EI_{(g_i,g_j)}(g_t)$
  if $g \in p$ **then**
    $EI_{g,p_{i_k}}(g_t) = EI_{g,p_{i_k}}(g_t) + I_g(p_{i_k}) * I^{(p)}_{p_{i_k}}(g_t)$
  else
    if $EI_{g,p_{i_k}}(g_t) > T$ **then**
      for every $p_g \in parents(g)$ in pbn **do**
        $EI = computeEI(p_g, \{p, g\}, I_g(p_{i_k}) * I^{(p)}_{p_{i_k}}(g_t), T, g_t, pbn, EI)$
      end for
    end if
  end if
  return EI

B. Edge elimination for approximate solutions of FMDPs

According to Definition 1, the EI value is a measure of how a certain gene is effected by the changes in values of other genes. In FMDP, the solution includes genes that have some effect on the reward genes. So, genes that have no effect on the reward genes at any time in the future can be eliminated from FMDP. In a previous study, we proposed a method to choose and eliminate unimportant genes from an MDP [26]. However, based on the study described in this paper, we realized that instead of eliminating a gene completely, removing some of the unimportant edges from a DBN in FMDP may produce better results.

Given the set of reward genes Γ of FMDP, $EIS_{i,j}(\Gamma)$ denotes how each relevant edge in the FMDP influences the set of reward genes. This influence can be very low and negligible for some of the edges. This means that edges with low EIS values can be eliminated from consideration. So, given a threshold δ, the edges with the smallest EIS values whose total EIS does not exceed δ are removed. After an edge is eliminated, we can marginalize out the eliminated edge from DBN. But based on the conducted experiments, we realized that maximum likelihood learning of the parameters of the whole DBN based on data sampled from the original DBN gives better results.

Algorithm 2 *reduceFMDP*$(EIS(\Gamma), \delta, M, D)$

Input: $EIS(\Gamma)$,δ,FMDP M,D
Output: FMDP $\hat{M}$
  $\hat{M} = M$
  Let S be the sorted set of edges $E_{i,j}$ where $EIS_{i,j}(\Gamma) \neq 0$
  Take the first k edges,S_k, from S such that $\sum_{E_{i,j} \in S_k} EIS_{i,j}(\Gamma) < \delta$
  for all $E_{i,j} \in S_k$ **do**
    Remove edge $E_{i,j}$ from DBNs for all actions in $\hat{M}$
    Learn maximum likelihood parameters of new DBN from data D
  end for
  return $\hat{M}$

The process in Algorithm 2 reduces a given FMDP M, to another possibly *sparser* FMDP $\hat{M}$ by applying the procedure described above. Let π^* and $\hat{\pi}^*$ denote the optimal

policies for M and $\hat{M}$, respectively; $\hat{\pi}^*$ will depend on fewer number of variables than π^* because of the absent edges. This means that value trees or policy trees will require less computational resources to store and modify.

V. Experiments

The method proposed in this paper is a pre-processing procedure to obtain a compact FMDP to find an approximate solution for large state spaces. It can be used to eliminate "unimportant" edges prior to using any FMDP solver. All experiments have been performed using a computer with 2.4 GHz Core(2) CPU and 3GB of RAM running Linux.

TABLE I

Mutated T-cell activation model

Product	Predictors
$CD45$	$Input$
$CD8$	$Input$
$TCRlig$	$Input$
Ca	$IP3$
$Calcin$	Ca
$cCbl$	1
$CREB$	Rsk
DAG	$PLCg(act)$
ERK	MEK
Fos	ERK
Fyn	$(Lck \wedge CD45) \vee (TCRbind \wedge CD45)$
$Gads$	LAT
$Grb2Sos$	LAT
$lKKbeta$	$PKCth$
$IP3$	$PLCg(act)$
JNK	SEK
Jun	JNK
LAT	1
Lck	$\overline{PAGCsk} \wedge CD8 \wedge CD45$
lkB	$\overline{lKKbeta}$
ltk	$SLP76$
MEK	Raf
$PAGCsk$	$Fyn \vee \overline{TCRbind}$
$PKCth$	DAG
$PLCg(act)$	$(SLP76 \wedge PLCg(bind)) \wedge (ltk \vee Rlk)$
$PLCg(bind)$	LAT
Raf	Ras
Ras	$Grb2Sos \vee RasGRPI$
$RasGRPI$	$PKCth \wedge DAG$
Rlk	Lck
Rsk	ERK
SEK	$PKCth$
$SLP76$	$Gads$
$TCRbind$	$TCRlig \wedge \overline{cCbl}$
$TCRphos$	$Fyn \vee (TCRbind \wedge Lck)$
$AP1$	$Jun \wedge Fos$
CRE	$CREB$
$NFAT$	$Calcin$
$NFkB$	$\overline{lkB}$

To demonstrate applicability, we performed an experiment similar to Faryabi *et al* [8]. They solved a constrained MDP for the control of mammalian cell-cycle. The boolean model for the signalling pathway of the mammalian cell-cycle includes 9 genes. A similar boolean pathway for the activation of transcription factors (TFs) that activate T-cells is given in [14]; this model has 40 genes. So, solving this problem with MDP formalism requires very large resources as the size of the state space is 2^{40}.

Chronic lymphocytic leukemia (CLL) is a type of cancer that is caused by the uncontrolled proliferation of immunologically immature lymphocytes. ZAP70 is an important gene in the signaling pathway of T-cell activation [11]. High ZAP-70 expression is thought to be the indicator of T-cell activation and prognosis and overall survival for CLL [9], [17]. Similarly, in T-cell activation model of Klamt *et al.* [14], if ZAP-70 is overexpressed (it is always *ON*), then the TFs that lead to proliferation of T-cells become always active (*ON*). So, in the light of these findings, we introduced a ZAP-70 overexpression mutation to the model given in [14]. ZAP-70 is, therefore, always active (*ON*) in our mutated model given in Table I as logical formulae, similar to the way the mammalian cell-cycle model is given in [8]. This mutation (according to our model) leads to unlimited T-cell proliferation; a cancerous state.

The first three genes in Table I are the input variables as given in [14], and the last four are the output TFs which activate T-cells. The interpretation of the table is simple; the next state of the genes in "Product" column is determined by the present states and logical formulae of the genes in "Predictors" column. The model can be formulated as a PBN by considering different instantiations of input genes. A PBN can also be represented as a set of boolean networks and a switching probability between these, which selects the boolean network that will be used to determine next gene activity profile [18]. The different values of input genes actually lead to different boolean networks which can be "merged" into a single PBN with equal switching probabilities [8].

Having defined the PBN model of T-cell activation, the control problem here is defined as finding an intervention strategy that avoids the activation of output TFs. So, the states that we try to avoid are those where AP1, CRE, NFAT and NFkB are all active (*ON*) together. This way, the unlimited proliferation of T-cells may be stopped. Given one of the genes as the control gene and a "noop" action, we tried to find the best control gene and intervention strategy using the proposed methods.

The problem is too large to solve exactly; so for an approximate solution, we used APRICODD with the size parameter set to 75. To be able to evaluate the policies, we performed simulations. We applied each policy in a simulation starting from a random initial state for 10,000 steps and counted the number of undesired states (the states where all output TFs are ON) and number of interventions during the simulation. The results are reported in Table II; note that not all of the genes exist as control genes, only those that lead to good policies in terms of the simulation results. Without any intervention, the system stays in undesirable states in 9,754 steps out of 10,000. The last value of each column in Table II is the time elapsed to find the control policy for that control gene and δ. Times for $\delta \neq 0$ also include the time needed to sample from the original distribution and to learn parameters for the new model.

The durations for finding the control policies in Table II are dominated by the time for sampling and parameter learning. Solving the FMDP for $\delta \neq 0$ take very short times (around 0.02 secs each). Another notable property of the output is that although the durations for $\delta = 0$ take varying amount of time, durations for $\delta \neq 0$ are almost equal.

Among all other control genes, ERK and MEK are the most effective ones in terms of both the number of undesired states and the number of interventions. Number of interventions here is important as it represents the cost associated with the control policy. ERK and MEK effectively stop activation of four output TFs with relatively low cost. The policy for ERK as the control gene is given in Figure 1. It

depends on five genes; when all of them are ON, it intervenes ERK. $Raf/MEK/ERK$ pathway has been shown to be important in the development of leukemia [25]. It is also interesting to note that Raf in this pathway is not as effective as others; a point that needs further investigation.

TABLE II

T-CELL ACTIVATION CONTROL RESULTS, NUMBER OF UNDESIRED STATES, NUMBER OF INTERVENTIONS AND TIME IN SECONDS

Control gene	$\delta = 0$	$\delta = 0.05$	$\delta = 0.1$	$\delta = 0.2$
Ca	22, 9644, 94.5	17, 9606, 41.7	17, 9669, 41.6	19, 9662, 41.7
$Calcin$	5, 9642, 173.3	9, 9601, 41.7	10, 9595, 41.6	10, 9553, 41.6
$CREB$	8, 9596, 161.6	8, 9571, 41.7	5, 9647, 41.7	10, 9618, 41.6
DAG	3248, 3238, 229.9	3243, 3262, 41.7	3259, 3245, 41.7	3246, 3260, 41.7
ERK	18, 4870, 271.5	22, 4874, 41.7	25, 4859, 41.7	19, 4869, 41.7
Fos	6, 9592, 170.1	7, 9572, 41.7	7, 9602, 41.6	12, 9578, 41.7
$lKKbeta$	15, 9673, 91.2	18, 9644, 41.6	24, 9608, 41.6	27, 9599, 41.9
JNK	19, 9687, 91.7	15, 9640, 41.8	17, 9658, 41.8	15, 9639, 41.7
Jun	8, 9614, 161.3	10, 9660, 41.8	14, 9610, 41.7	16, 9586, 41.7
lkB	5, 9650, 166.6	11, 9591, 41.7	11, 9599, 41.7	8, 9580, 41.6
MEK	23, 4896, 124.2	29, 4890, 41.6	30, 4900, 41.7	20, 4874, 41.7
$PKCth$	36, 4875, 138.3	30, 4874, 41.7	40, 4874, 41.7	26, 4886, 41.6
$PLCg(act)$	2461, 2440, 219.3	2455, 2445, 41.7	2464, 2450, 41.7	2456, 2449, 41.7
Raf	3253, 3252, 212.5	3238, 3254, 41.7	3263, 3248, 41.7	3265, 3270, 41.7
Rsk	20, 9655, 88.8	18, 9658, 41.70	2, 9653, 41.7	23, 9620, 41.8

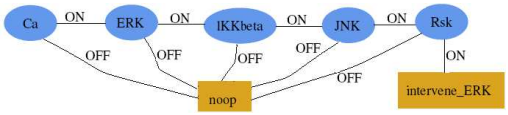

Fig. 1. Policy obtained when ERK is the control gene and $\delta = 0.2$

VI. CONCLUSIONS AND FUTURE WORK

Devising intervention strategies for PBNs is an important and hard problem. In this paper, we demonstrated an approach to solve the PBN control problem in the context of FMDP, which is a well-known framework in the machine learning community. The first contribution of this paper is applying the FMDPs method to solve the PBN control problem for the first time. This may lead to better scalability in representation and solution of the control problems in bioinformatics. We also proposed a method to approximately solve the control problem by eliminating some of the edges in the model. The results presented show the applicability and success of the proposed algorithm.

In biology and medicine, in addition to solving the control problem, it is almost equally important to find an applicable or simple policy. A complex policy is very hard to apply in the absence of more "intelligent" drugs. Based on this argument, it is necessary to find a policy as simple as possible. In this sense, our proposed method of simplifying the model by edge elimination can help finding approximate but good and simple policies. Finally, the current implementation requires the value of the threshold δ be given by an expert; however, as one of our targets is to minimize user involvement as much as possible, we are currently investigating an automated way for finding δ given an FMDP M; the preliminary results are encouraging and even promise a possible improvement in the overall performance.

REFERENCES

[1] O. Abul, R. Alhajj, and F. Polat. Markov decision processes based optimal control policies for probabilistic boolean networks. In *Proceedings of IEEE BIBE*, pages 337–344, 2004.

[2] R.E. Bellman. *Dynamic Programming*. Princeton University Press,Princeton, New Jersey, 1957.

[3] C. Boutilier, R. Dearden, and M. Goldszmidt. Exploiting structure in policy construction. In *Proceedings of IJCAI*, pages 1104–1111, 1995.

[4] Craig Boutilier, Richard Dearden, and Moises Goldzmidt. Stochastic Dynamic Programming with Factored Representations. *Artificial Intelligence*, 121:49–107, 2000.

[5] A. Datta, A. Choudhary, M.L. Bittner, and E.R. Dougherty. External Control in Markovian Genetic Regulatory Networks. *Machine Learning*, 52:169–191, 2003.

[6] T. Dean and K. Kanazawa. A model for reasoning about persistance and causation. *Computational Intelligence*, 5(3):142–150, 1989.

[7] E.R. Dougherty, S. Kim, and Y. Chen. Coefficient of determination in nonlinear signal processing. *Signal Processing*, 80:2219–2235, 2000.

[8] B. Faryabi, G. Vahedi, J.F. Chamberland, A. Datta, and E. R. Dougherty. Optimal Constrained Stationary Intervention in Gene Regulatory Networks. *EURASIP Journal on Bioinformatics and Systems Biology*, To appear.

[9] Y. Herishanu, S. Kay, and O. Rogowski *et al.* T-cell ZAP-70 overexpression in chronic lymphomatic leukemia (CLL) correlates with CLL ZAP-70 levels, clinical stage and disease progression. *Leukemia*, 19:1289–1291, 2005.

[10] J. Hoey, R. St-Aubin, A. Hu, and C. Boutilier. SPUDD: Stochastic Planning using Decision Diagrams. In *Proceedings of UAI*, 1999.

[11] Y. Huang and RL Wange. T cell receptor signaling: beyond complex complexes. *Journal of Biological Chemistry*, 279(28), 2004.

[12] D.R. Karger, R.Motwani, and G.D.S. Ramkumar. On approximating the longest path in a graph. *Algorithmica*, 18:82–98, 1997.

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

[14] S. Klamt, J. Saez-Rodriguez, J. A. Lindquist, L. Simeoni, and Ernst D. Gilles. A methodology for the structural and functional analysis of signaling and regulatory networks. *BMC Bioinformatics*, 7(56), 2006.

[15] H. Lähdesmäki, S. Hautaniemi, I. Shmulevich, and O. Yli-Harja. Relationships between probabilistic Boolean networks and dynamic Bayesian networks as models of gene regulatory networks. *Signal Processing*, 86(4):814–834, 2006.

[16] H. Lähdesmäki, I. Shmulevich, and O. Yli-Harja. On Learning Gene Regulatory Networks Under the Boolean Network Model. *Machine Learning*, 52(1-2):147–167, 2003.

[17] JA Orchard, RE Ibbotson, Z. Davis, A. Wiestner, A. Rosenwald, and PW Thomas *et al.* ZAP-70 expression and prognosis in chronic lymphomatic leukemia. *Lancet*, 363:105–111, 2004.

[18] R. Pal, A. Datta, and E. R. Dougherty. Robust Intervention in Probabilistic Boolean Networks. *IEEE Transactions on Signal Processing*, 56(3):1280–1294, 2008.

[19] R. Pal, A. Datta, and E.R. Dougherty. Optimal Infinite-Horizon control for Probabilistic Boolean Networks. *IEEE Transactions on Signal Processing*, 54(6):2375–2387, 2006.

[20] ML Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, New York, USA, 1994.

[21] R.I. Bahar, E.A. Frohm, and C.M. Gaona *et al.* Algebraic Decision Diagrams and Their Applications. In *IEEE /ACM International Conference on CAD*, pages 188–191. ACM/IEEE, 1993.

[22] I. Shmulevich, E. R. Dougherty, S. Kim, and W. Zhang. Probabilistic boolean networks: A rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, 18(2):261–274, 2002.

[23] I. Shmulevich, E.R. Dougherty, and W. Zhang. Gene perturbation and intervention in probabilistic boolean networks. *Bioinformatics*, 18(10):1319–1331, 2002.

[24] R. St-Aubin, J. Hoey, and C. Boutilier. APRICODD: Approximate Policy Construction using Decision Diagrams. In *Advances in Neural Information Processing 13*, 2000.

[25] L.S. Steelman, S.L. Abrams, and J. Whelan *et al.* Contributions of the Raf/MEK/ERK, PI3K/PTEN/akt/mTOR and Jak/STAT pathways to leukemia. *Leukemia*, 22:686–707, 2008.

[26] M. Tan, F. Polat, and R. Alhajj. Feature Reduction for Gene Regulatory Network Control. In *Proceedings of IEEE BIBE*, 2007.