2008 American Control Conference
Westin Seattle Hotel, Seattle, Washington, USA
June 11-13, 2008

Mean First-Passage Time Control Policy versus Reinforcement-Learning Control Policy in Gene Regulatory Networks

Golnaz Vahedi, Babak Faryabi, Jean-Francois Chamberland, Aniruddha Datta, Edward R. Dougherty

Abstract— **Probabilistic Boolean Networks are rule-based models for gene regulatory networks. They are used to design intervention strategies in translational genomics such as cancer treatment. Previously, methods for finding control policies with the highest effect on steady-state distributions of probabilistic Boolean networks have been proposed. These methods were derived using the theory of infinite-horizon stochastic control. It is well-known that the direct application of optimal control methods is problematic owing to their high computational complexity and the fact that they require the inference of the system model. To bypass the impediment of model estimation, two algorithms for approximating the optimal control policy have been introduced. These algorithms are based on reinforcement learning and mean first-passage times. In this work, the performance of these two methods are compared using both a melanoma-related network and randomly generated networks. It is shown that the mean-first-passage-time-based algorithm outperforms the reinforcement-learning-based algorithm for smaller amount of training data, which corresponds better to feasible experimental conditions. In contrary to the reinforcement-learning-based algorithm, during the learning period of the mean-first-passage-time-based algorithm, the application of control is not required. Intervention in biological systems during the learning phase may induce undesirable side-effects.**

I. Introduction

Modeling of genetic regulatory networks as a means for gaining insight into the underlying processes of living systems is becoming increasingly widespread. An ultimate objective of modeling genetic regulatory networks is the development of computational tools for the discovery of potential targets in diseases such as cancer [1].

To date, stochastic regulatory intervention has been studied in the context of probabilistic Boolean networks (PBNs) [2]. On the topic, initial efforts have focused on manipulating external (control) variables that affect the transition probabilities of the network and can, therefore, be used to desirably affect its dynamical evolution over a finite time horizon [3] [4].

These short-term policies can change the dynamical performance of the network for a small number of stages; however, they are not necessarily effective in changing the steady-state behavior of a PBN. To address this issue, the theory of infinite-horizon optimal stochastic control has been employed to find stationary policies that affect the steady-state distributions of PBN [5]. An optimum stationary policy can be found using dynamic programming. In general, the direct application of optimal control methods is limited by the size of the state-space – this is known as the curse of dimensionality [6]. In our scenario, the complexity of these methods increases exponentially with the number of genes included in the model. Therefore, they can only be applied to small network models.

For larger biological models involving interactions among many genes, a stochastic control method that has less complexity is needed. To this end, we proposed two methods in [7] and [8]. In [7], we proposed an approximate stochastic control policy based on reinforcement learning (RL) methods. More specifically, Q-learning is employed to overcome the curse of dimensionality. The proposed approximate method assigns a stationary control policy which is called the *RL control policy*. The RL control policy yields an effective stationary policy, while possessing constant complexity with respect to the number of genes [7]. In [8], we proposed an algorithm based on mean first-passage times (MFPT) and referred to it as the *MFPT algorithm*. The MFPT algorithm assigns a stationary control policy, called the *MFPT control policy*, that reduces the likelihood of being in undesirable states. We showed that to reduce the complexity of the optimal stochastic control problem, the MFPT control policy can be used as an approximate solution.

A salient feature in both methods ([7] and [8]) is that they are *model-free*, i.e. they do not require the perfect knowledge of all system's parameters. Traditionally, the proposed intervention methods for PBNs have been model dependent, requiring at least knowledge of the transition probability matrix. This can be derived from the PBN if the latter is known. Since in practice PBNs are not known except via system identification from experimental data, one is faced with a difficult inference problem [9]. This problem can be avoided by directly inferring the transition probability matrix; however, this is still a formidable task because the complexity of estimating the transition probabilities of a Markov chain increases exponentially with the number of genes in the model.

When time-course data are available, the RL and MFPT control policies can be implemented directly from the data points. In the RL method, the value of the Q-factor for a state-control pair is updated whenever a transition occurs in the system, given the control is selected randomly among all

This work was supported in part by the National Science Foundation (ECS- 0355227, CCF-0514644 & ECCS-0701531), the National Cancer Institute (R01 CA-104620, & CA-90301) and the Translational Genomics Research Institute.

G. Vahedi, B. Faryabi, J.F. Chamberland, A. Datta are with Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843 golnaz, bfariabi, chmbrlnd, datta@ece.tamu.edu

E. Dougherty is with Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843 and Computational Biology Division, Translational Genomics Research Institute, Phoenix, AZ 85004 edward@ece.tamu.edu

the possible controls [7]. The MFPT control policy can be designed based on estimates of the mean first-passage times from the time-course data [8]. These model-free intervention methods have low complexity, are robust to modeling errors, and are adaptive to changes in the underlying biological system.

In this work, we compare the RL and MFPT methods. Numerical results suggest that for a limited amount of time-course data the approximation by the MFPT control policy is more accurate than the RL control policy. The performance of the RL-based algorithm becomes superior upon increasing the size of the training data. We discuss the necessary random application of control in the learning phase of the RL algorithm as another disadvantage of the RL control policy. The paper is organized as follows. We give necessary definitions and introduce the RL and MFPT policies in Section II. Comparative simulation results are presented in Section III and Section IV contains some concluding remarks.

II. Background

A. Probabilistic Boolean Networks

A context-sensitive probabilistic Boolean network (PBN) consists of a sequence $V = \{x_i\}_{i=1}^{n}$, of n nodes, where $x_i \in \{0,1\}$, and a sequence $\{\mathbf{f}_l\}_{l=1}^{k}$ of vector-valued functions, called *predictor functions*. In the framework of gene regulation, each x_i represents the expression value of a gene. Every vector-valued function $\mathbf{f}_l$, which has the form of $\mathbf{f}_l = (f_{l1}, \ldots, f_{ln})$, determines a constituent network of the context-sensitive PBN. The function $f_{li} : \{0,1\}^n \to \{0,1\}$ is a predictor of gene i, whenever network l is selected. At each time step, the context is switched with probability q. If the network is not switched, then the context-sensitive PBN behaves as a fixed network and synchronously updates the values of all the genes according to the current predictor function. If it is decided that the network should be switched, a predictor function is randomly selected according to a distribution $\{r_1, \ldots, r_k\}$. After selecting the predictor function $\mathbf{f}_l$, the values of genes are updated accordingly, that is, according to the network determined by $\mathbf{f}_l$. We consider context-sensitive PBNs with perturbations. Each gene may change its value with small perturbation probability p at every time step.

The gene-activity profile (GAP) is an n-digit binary vector $\mathbf{x}(t) = (x_1(t), \ldots, x_n(t))$ giving the expression values of genes at time t where $x_i \in \{0,1\}$. There is a natural bijection between the GAP ($\mathbf{x}(t)$) and its decimal representation z_t which takes values from 0 to $2^n - 1$.

In the presence of external controls, we suppose that the PBN has m binary control inputs: $c_1(t), \ldots, c_m(t)$. A control $c_i(t)$ takes values in $\{0,1\}$ at each dynamical step t. The decimal bijection of the control vector, $u(t) \in \mathcal{C} = \{0, 1, \ldots, 2^m - 1\}$, describes the complete status of all the control inputs. It is worthwhile to note that biological or economic considerations will likely constrain us to use only one treatment at a time. Hence in this work, we focus on the application of one gene as a control.

The system evolution can be modeled by the equation:

$$z_{t+1} = f(z_t, u_t, w_t) \quad \text{for } t = 0, 1, \ldots$$

where the state z_t is an element of the state-space $\mathcal{S} = \{0, 1, \ldots 2^n - 1\}$. The disturbance w_t is the manifestation of uncertainties in the context-sensitive PBN. The PBN is then modeled as a Markov chain with 2^n states where the state z_t at any time step t is the decimal bijection of the GAP. Originating from a state i, the successor state j is selected randomly within the set $\mathcal{S}$ according to the transition probability $p_{ij}(u)$:

$$p_{ij}(u) \overset{\triangle}{=} P(z_{t+1} = j | z_t = i, u_t = u)$$

for all i and j in $\mathcal{S}$, and for all u in $\mathcal{C}$. Gene perturbation insures that all the states in the Markov chain communicate with each other. Hence, the finite-state Markov chain has a unique steady-state distribution [1].

B. Optimal Intervention

A cost-per-stage, $g(i,u,j)$, is associated to each intervention in the system. A cost-per-stage depends on the origin state i, the successor state j, and the control input u. We assume the cost-per-stage is stationary and bounded for all i, j, and u. We define the expected immediate cost in state i, when control u is selected, by

$$\overline{g}(i,u) = \sum_{j \in \mathcal{S}} p_{ij}(u)\, g(i,u,j).$$

To define the expected total cost, we consider a discounted cost formulation. The discounting factor, $\alpha \in (0,1)$, insures the convergence of the expected total cost over the long-run [6]. Including a discounting factor in the expected total cost signifies that the incurred cost at a later time is less significant than the incurred cost at an earlier time. In the case of cancer therapy, the discounting factor emphasizes that obtaining treatment at an earlier stage is better than doing so at a later stage since the former brings about a larger cost reduction.

Among all admissible policies Π, the infinite-horizon optimal stochastic control methodology finds a policy $\pi = \{\mu_0, \mu_1, \ldots\}$, where $\mu_t : \mathcal{S} \to \mathcal{C}$ is the decision rule at time step t that minimizes the expected total discounted cost.

The infinite-horizon expected total discounted cost, given the policy π and the initial state i, is

$$J_\pi(i) = \lim_{N \to \infty} E\left\{ \sum_{t=0}^{N-1} \alpha^t\, g(i, \mu_t(i), j) \right\}. \tag{1}$$

The vector of accumulated cost, $\mathbf{J} = (J(1), \ldots, J(|\mathcal{S}|))$, is called the *value function*. We seek a policy that minimizes the value function for each state i. The optimal value function, $\mathbf{J}^*$, is a solution to the infinite-horizon optimal stochastic control with discounted cost:

$$J^*(i) = \min_{\pi \in \Pi} J_\pi(i), \qquad \forall\, i \in \mathcal{S}. \tag{2}$$

A stationary policy is an admissible policy of the form $\pi = \{\mu, \mu, \ldots\}$. The vector $\mathbf{J}_\mu$ denotes its corresponding value function. The stationary policy π is optimal if $J_\mu(i) = J^*(i)$ for any state i. It is known that an optimal stationary policy exists for the discounted infinite-horizon stochastic optimal control problem, and it is given by the fixed-point solution of the Bellman optimality equation [6]. A dynamic programming algorithm is used to iteratively find the fixed point of the Bellman optimality equation. At each iteration the value function must be computed for all states in the state-space. Hence, the computational complexity of the method increases exponentially with the number of genes present in the system model.

C. Reinforcement Learning Control Policy

Using the definition of expected immediate cost, the Bellman optimality equation can be rewritten as follows: for each state $i \in \mathcal{S}$,

$$J^*(i) = \min_{u \in \mathcal{C}} \left[\sum_{j=0}^{2^n-1} p_{ij}(u) \Big(g(i,u,j) + \alpha\, J^*(j) \Big) \right].$$

Accordingly, we can define the *Q-factor* for each state-control pair (i,u) by

$$Q(i,u) \triangleq \sum_{j=0}^{2^n-1} p_{ij}(u) \Big(g(i,u,j) + \alpha\, J^*(j) \Big). \qquad (3)$$

The relation between the optimal value function of a state i and the Q-factors of the same state is given by

$$J^*(i) = \min_{u \in \mathcal{C}} Q(i,u).$$

To compute the Q-factor iteratively, the Bellman optimality equation can be written for the Q-factor as

$$Q(i,u) = \sum_{j=0}^{2^n-1} p_{ij}(u) \left[g(i,u,j) + \alpha \min_{u' \in \mathcal{C}} Q(j,u') \right]. \qquad (4)$$

Using the Q-factor, one can rewrite the value iteration algorithm [7] in a meaningful form. Estimation of the Q-factor becomes the objective of the value iteration algorithm. Q-factor can be expressed as the expected value of a random variable $\Psi(i,u)$. Equation (3) can be rewritten as

$$Q(i,u) = E\left[g(i,u,j) + \alpha \min_{u' \in \mathcal{C}} Q(j,u') \right] = E\big[\Psi(i,u)\big].$$

If the samples of $\Psi(i,u)$ are generated using a system simulator, then one can estimate its expected value. Let

$$\overline{\Psi}_k(i,u) = \frac{\sum_{i=1}^k \psi_i}{k}$$

denote the empirical average of $\Psi(i,u)$ using k samples, where ψ_i is the ith sample of the random variable $\Psi(i,u)$. Upon a new observation of $\Psi(i,u)$, the value of $\overline{\Psi}_k(i,u)$ can be updated by

$$\overline{\Psi}_{k+1}(i,u) = \overline{\Psi}_k(i,u) - \frac{\overline{\Psi}_k(i,u)}{k+1} + \frac{\psi_{k+1}}{k+1}.$$

If $\alpha^{k+1} = \frac{1}{k+1}$, then

$$\overline{\Psi}_{k+1}(i,u) = \overline{\Psi}_k(i,u)\left(1 - \alpha^{k+1}\right) + \alpha^{k+1}\psi_{k+1}.$$

Hence, given a new system observation, the Q-factor is iteratively updated for the specific state-control pair (i,u) according to the following rule:

$$Q^{(k+1)}(i,u) \longleftarrow (1-\alpha) Q^{(k)}(i,u) + \alpha \left[g(i,u,j) + \alpha \min_{u' \in \mathcal{C}} Q^{(k)}(j,u') \right]. \qquad (5)$$

The revised value iteration algorithm in which the Q-factors are updated according to (5) is called the *Q-learning algorithm* [10]. The Q-learning algorithm is summarized as Algorithm 1. Since the updating rule of (5) is invariant with regard to the transition probabilities of the system, the Q-learning procedure is a model-free algorithm. In the Q-learning algorithm, the value of the Q-factor for a state-control pair (i,u) is updated whenever a transition from state i to state j occurs in the system simulator, given the control u is selected randomly among all the possible controls. The term $v(i,u)$ denotes the number of times the state-control pair (i,u) is visited. We define the step-size α^k equal to C/k, where C is a positive constant in the interval $(0,1)$ and k is $v(i,u)$, whenever the state-control pair (i,u) occurs.

In the Q-learning algorithm, the system simulator generates trajectories of state-control pairs; hence some Q-factors may be updated more often than others. An appropriate step-size α^k is needed to guarantee the convergence of the Q-learning algorithm to the optimal control strategy despite the asynchronous updating of the Q-factors [10]. Several step-sizes are proposed with the general form $\frac{C}{a+k}$, where C and a can be any positive constants [10]. As a general rule, the step-size should be small, and diminish to zero at a suitable rate [11]. Here, we assumed a simple form for the step-size inspired mainly by the argument that the ensemble average can be estimated using the time average of the sample data. Once the RL control policy is found by the Q-learning algorithm, it can be applied to the controlled system to affect the steady-state distribution of the network. In algorithm (1), the complexity of each iteration is $\mathbf{O}(1)$ with respect to n. Hence, Q-learning runs in constant complexity with respect to the number of genes in the network.

If all the state-control pairs (i,u) are visited infinitely often, then for each state-control pair the estimated expected value, $\overline{\Psi}_k(i,u)$, converges to its ensemble average $E\big[\Psi(i,u)\big]$ with probability one. Hence, we expect that an approximate stationary policy computed by the Q-learning algorithm converges to the optimal stationary policy. The convergence of the approximate stationary policy to the optimal stationary policy is proved in [11], and the numerical results in [7] illustrate this fact for a special case. The learning duration of the Q-learning algorithm should increase as the number of genes in the network increases in order to obtain an approximate stationary policy close to the optimal stationary policy. Therefore, the Q-learning algorithm, as any other learning algorithm, may not be suitable for very large networks.

Algorithm 1 Q-learning algorithm

```
$Q(i,u) \leftarrow 0$ for all $i \in \mathcal{S}, u \in \mathcal{C}$
$v(i,u) \leftarrow 0$ for all $i \in \mathcal{S}, u \in \mathcal{C}$
Setting $0 < C < 1$
Setting $k_{max}$
Selecting an arbitrary initial state $i$.
for $k = 0$ to $k_{max}$ do
    Control selection: Selecting control $u \in \mathcal{C}$ randomly, given the current state is $i$.
    Extracting information from system's simulator: According to the transition in the system simulator, the successor state $j$ as well as the earned cost-per-stage $g(i,u,j)$ are determined in a transition from state $i$ to state $j$ under control $u$. Hence, the following updates are performed:
    $v(i,u) \leftarrow v(i,u) + 1$
    $\alpha \leftarrow \frac{C}{v(i,u)}$.
    Updating $Q(i,u)$ : The value of $Q(i,u)$ is updated according to (5).
    $$Q(i,u) \leftarrow (1-\alpha)Q(i,u)+\alpha\left[g(i,u,j) + \alpha \min_{u' \in \mathcal{C}} Q(j,u')\right]$$
    Setting $i \leftarrow j$
end for
Finding the suboptimal policy: Choose the suboptimal policy for all $i \in \mathcal{S}$
    $$\mu(i) = \arg\min_{u \in \mathcal{C}} Q(i,u)$$
```

D. Mean First-Passage Time Control Policy

The state space of a PBN can be partitioned into subsets of "desirable" and "undesirable" states. Without loss of generality, one can assume that P, the Markov chain transition probability matrix, can be accordingly partitioned as follows:

$$\mathbf{P} = \begin{pmatrix} P_{\mathcal{D},\mathcal{D}} & P_{\mathcal{D},\mathcal{U}} \\ P_{\mathcal{U},\mathcal{D}} & P_{\mathcal{U},\mathcal{U}} \end{pmatrix},$$

where $\mathcal{D}$ and $\mathcal{U}$ are the subsets of desirable and undesirable states, respectively.

The mean first-passage time vectors are computed by solving the following linear system of equations [12]:

$$K_{\mathcal{U}} = e + P_{\mathcal{D},\mathcal{D}} \cdot K_{\mathcal{U}}, \tag{6}$$

$$K_{\mathcal{D}} = e + P_{\mathcal{U},\mathcal{U}} \cdot K_{\mathcal{D}}, \tag{7}$$

where e is a column vector of ones with appropriate length, $K_{\mathcal{U}}$ is a vector containing the mean first-passage times from each state in $\mathcal{D}$ to $\mathcal{U}$, and $K_{\mathcal{D}}$ is a vector containing the mean first-passage times from each state in $\mathcal{U}$ to $\mathcal{D}$. The proposed algorithm finds the MFPT control policy $\mu_{\mathbf{g}}$ based on mean first-passage times for each control gene g.

When the network consists of n genes, there are 2^n states and $\mu_{\mathbf{g}}$ is a vector of size 2^n. The value of $\mu_{\mathbf{g}}(\mathbf{x})$ represents the stationary control policy of state $\mathbf{x}$. Having $\mu_{\mathbf{g}}(\mathbf{x}) = 0$ signifies that whenever the PBN reaches state $\mathbf{x}$, no control is applied and the system continues its progression based on the transition probabilities of state $\mathbf{x}$. Having $\mu_{\mathbf{g}}(\mathbf{x}) = 1$ means whenever the PBN reaches state $\mathbf{x}$, the control should be applied. The system then continues its progression based on the transition probabilities of state $\tilde{\mathbf{x}}$, which is the flipped state of $\mathbf{x}$ when control is applied to gene g. Algorithm 2 summarizes the proposed procedure. The complexity of the MFPT algorithm, which consists of two matrix inversions, is $\mathbf{O}(2^{3n})$.

Algorithm 2 MFPT algorithm

```
Partition the state-space into undesirable $\mathcal{U}$ and desirable $\mathcal{D}$ subsets.
Compute $K_{\mathcal{U}}$ and $K_{\mathcal{D}}$.
Select the control gene $g$ .
for All states $\mathbf{x}$ in $\mathcal{U}$ do
    $\tilde{\mathbf{x}} \leftarrow$ flip control gene $g$ in $\mathbf{x}$.
    if $K_{\mathcal{D}}(\mathbf{x}) > K_{\mathcal{D}}(\tilde{\mathbf{x}})$ then
        $\mu_{\mathbf{g}}(\mathbf{x}) = 1$;
    else
        $\mu_{\mathbf{g}}(\mathbf{x}) = 0$;
    end if
end for
for All states $\mathbf{x}$ in $\mathcal{D}$ do
    $\tilde{\mathbf{x}} \leftarrow$ flip control gene $g$ in $\mathbf{x}$.
    if $K_{\mathcal{U}}(\tilde{\mathbf{x}}) > K_{\mathcal{U}}(\mathbf{x})$ then
        $\mu_{\mathbf{g}}(\mathbf{x}) = 1$;
    else
        $\mu_{\mathbf{g}}(\mathbf{x}) = 0$;
    end if
end for
```

When time-course data are available, the MFPT algorithm can be implemented by directly estimating the mean first-passage times. The estimated mean first-passage times are used to construct the matrices of the mean first-passage times, $K_{\mathcal{U}}$ and $K_{\mathcal{D}}$. The MFPT algorithm can then be applied to the estimated matrices $K_{\mathcal{U}}$ and $K_{\mathcal{D}}$ to devise a model-free MFPT control policy. Similar to the Q-learning algorithm, the model-free MFPT algorithm runs in constant time complexity with respect to the number of genes in the network.

An advantage to the model-free approach is that the estimated matrices $K_{\mathcal{U}}$ and $K_{\mathcal{D}}$ can be updated whenever new time-course data become available. The possibility of updating the estimated mean first-passage times enables the MFPT algorithm to adapt its control policy to the status of gene interactions. Similar to the Q-learning algorithm, the learning duration of the MFPT algorithm should increase as the number of genes in the network increases in order to obtain an approximate stationary policy close to the optimal stationary policy.

III. Results and Discussion

In this section, we first apply the RL and MFTP policies to control a network relating to metastasis in melanoma and compare the performance of the two methods to that of the optimal control. We also study the average performance of these RL and MFTP policies for random PBNs.

In the first study, we consider a ten-gene network consisting of WNT5A, Pirin, S100P, RET1, MMP3, PHOC, MART1, HADHB, Synuclein, and STC2. The above order of genes is used in the binary representation of the gene-activity profile, with WNT5A as the most significant bit and STC2 as the least significant bit. The abundance of mRNA for the gene WNT5A has been found to be highly discriminating between cells with properties typically associated with high versus low metastatic competence [13]. This motivates us to select WNT5A as the target gene. Hence, the state-space is partitioned to desirable states with down-regulated WNT5A and undesirable states with up-regulated WNT5A. A modification of the algorithm described in [14] is used to infer ten Boolean networks that constitute the context-sensitive PBN. Interactions among genes of one of these Boolean networks is shown in Fig. **??**.

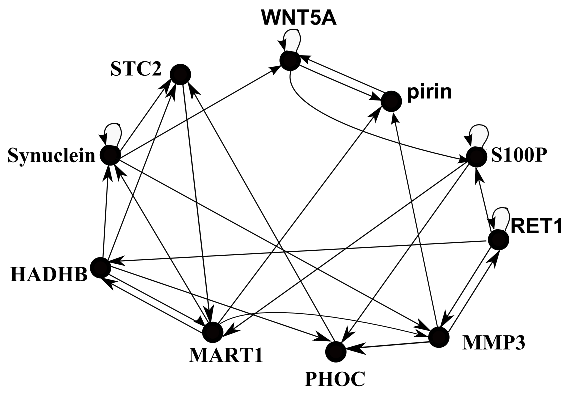

Fig. 1. This is the regulatory graph of one Boolean network that corresponds to WNT5A experiment.

Control of the PBN resembles a therapeutical situation in which the goal of the control is to reduce the likelihood of reaching undesirable states. In order to define the context-sensitive PBN, the switching probability, the perturbation probability, and probability of selecting each constituent Boolean network are assumed to be known. In the case of time-course data, these probabilities can be inferred [9].

We postulate an appropriate cost-per-stage function. The undesirable states are assigned higher cost compared to the desirable states. For fair comparison of the two algorithms, the cost of control is considered negligible compared to the cost of undesirable states. In practice, the assigned values have to capture the benefits and costs of the intervention and the relative preference of the states. They have to be set in consultation with physicians relying on their clinical judgement.

Having the down-regulation of WNT5A as the objective, we apply the RL and MFPT control policies described in Algorithms 1 and 2 to the inferred context-sensitive PBN. Here, we only consider a single control. Pirin has been chosen as the control gene [7]. If the control is high, $u = 1$, then the state of gene Pirin is reversed; if $u = 0$, then the state of Pirin remains unchanged.

We define ΔP to be the percentage change in the aggregated probability of the states with WNT5A$= 1$ before and after the intervention. As a performance measure, ΔP^{opt}, ΔP^{RL}, and ΔP^{MFPT} indicate the percentages of reduction in the total probability of the undesirable states in the steady state when the optimal, RL, and MFPT control policies are applied, respectively.

We generate time-course data for 10^6 time-steps from the existing model. We apply RL and MFPT policies after each 10^k time-steps, for $k = 3, \ldots, 6$. Hence, ΔP^{RL} and ΔP^{MFPT} are functions of the learning duration. On the other hand, ΔP^{opt} is computed from the original PBN by directly solving the dynamic programming problem. Fig. 2 shows $\Delta P^{\text{opt}} - \Delta P^{\text{RL}}$ and $\Delta P^{\text{opt}} - \Delta P^{\text{MFPT}}$ as a function of the logarithm of the learning duration. After 10^3 time-course data points, $\Delta P^{\text{opt}} - \Delta P^{\text{MFPT}}$ is 0.114 while $\Delta P^{\text{opt}} - \Delta P^{\text{RL}}$ is 0.166. In particular, for lower numbers of observations, which corresponds better to feasible experimental conditions, the approximation by the MFPT algorithm outperforms the RL-based algorithm. On the other hand, after 10^6 time-course data points, $\Delta P^{\text{opt}} - \Delta P^{\text{MFPT}}$ is 0.003 while $\Delta P^{\text{opt}} - \Delta P^{\text{RL}}$ is 0.002. As the size of the training data increases, the performance of the RL-based algorithm overtakes the one for the MFPT algorithm.

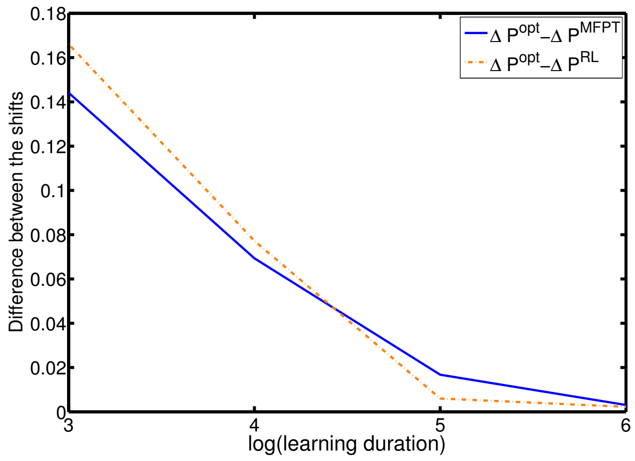

Fig. 2. $\Delta P^{\text{opt}} - \Delta P^{\text{RL}}$ and $\Delta P^{\text{opt}} - \Delta P^{\text{MFPT}}$ as a function of the log of the learning duration.

As we mentioned earlier, in Q-learning the value of the Q-factor for a state-control pair (i, u) is updated given the control u is selected randomly among all the possible controls. To this end, in the RL-based algorithm, the two possible values of control, $u \in \{0, 1\}$, should be applied to the model with equal probability. In contrast, the MFPT algorithm does not require any application of the control for obtaining the MFPT control policy. This is another advantage of the MFPT method over the RL method when the application of the external control may induce undesirable side-effects in biological systems.

We also study the performance of these two methods for random PBNs. We generate 1000 random PBNs. For each

PBN, time course data is generated as we explained earlier for 10^2 to 10^5 observations. The average of $\Delta P^{\text{opt}} - \Delta P^{\text{RL}}$ and $\Delta P^{\text{opt}} - \Delta P^{\text{MFPT}}$ is shown in Fig.3. The study on random PBNs confirms our observation in the melanoma case study.

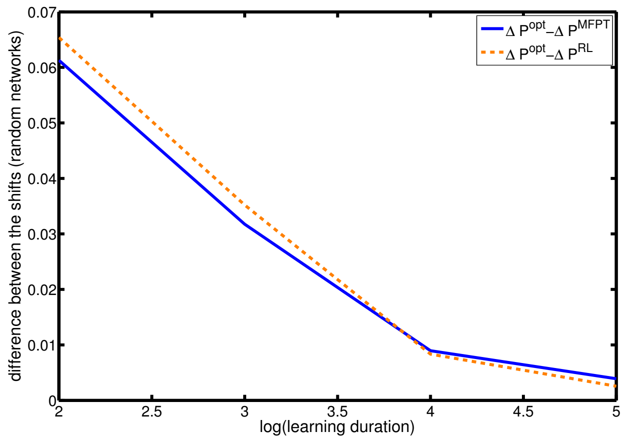

Fig. 3. Simulation studies for random PBNs: $\Delta P^{\text{opt}} - \Delta P^{\text{RL}}$ and $\Delta P^{\text{opt}} - \Delta P^{\text{MFPT}}$ as a function of the log of the learning duration.

IV. Conclusion

In this work, we compared the performance and the complexity of the reinforcement-learning and mean first-passage time methods using both a melanoma-related network and randomly generated networks. These two methods have been suggested earlier but never compared. We employed both methods in a model-free fashion. These model-free intervention methods have low complexity, are robust to modeling errors, and are adaptive to changes in the underlying biological system. We showed that the MFPT control policy outperforms the RL control policy for smaller numbers of training data. As the size of the training data increases, the RL-based algorithm performs better. The current definition of the MFPT algorithm does not take into account the cost of control. Therefore, the presented comparison is valid when the cost of control is negligible compared to the cost of undesirable states. Further study is being pursued to analyze the effect of the cost of control on the performance of both algorithms.

References

[1] I. Shmulevich, E. R. Dougherty, and W. Zhang, "Gene perturbation and intervention in probabilistic boolean networks," *Bioinformatics*, vol. 18, no. 10, pp. 1319–1331, 2002.

[2] I. Shmulevich, E. R. Dougherty, S. Kim, and W. Zhang, "Probabilistic boolean networks: a rule-based uncertainty model for gene regulatory networks," *Bioinformatics*, vol. 18, no. 2, pp. 261–274, 2002.

[3] A. Datta, A. Choudhary, M. Bittner, and E. R. Dougherty, "External control in markovian genetic regulatory networks," *Machine Learning*, vol. 52, no. 1/2, pp. 169–191, 2003.

[4] A. Datta, R. Pal, and E. R. Dougherty, "Intervention in probabilistic gene regulatory networks," *Current Bioinformatics*, vol. 1, no. 2, pp. 167–184, 2006.

[5] R. Pal, A. Datta, and E. R. Dougherty, "Optimal infinite-horizon control for probabilistic boolean networks," *IEEE Trans. on Signal Processing*, vol. 54, no. 6, pp. 2375–2387, 2006.

[6] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, Athena Scientific, Belmont, MA, 2001.

[7] B. Faryabi, A. Datta, and E. Dougherty, "On approximate stochastic control in genetic regulatory networks," *IET Systems Biology*, vol. 1, no. 6, pp. 361–368, 2007.

[8] G. Vahedi, B. Faryabi, J.F. Chamberland, A. Datta, and E.R. Dougherty, "Intervention in gene regulatory networks via a stationary mean-first-passage-time control policy," *submitted to IEEE Transactions on Biomedical Engineering*, 2007.

[9] S. Marshall, L. Yu, Y. Xiao, and E. R. Dougherty, "Inference of a probabilistic Boolean network from a single observed temporal sequence," *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2007, pp. 32454–32569, 2007.

[10] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynammic Programming*, Athena Scientific, Belmont, MA, 1996.

[11] J. N. Tsitsiklis, "Asynchronous stochastic approximation and q-learning," *Machine Learning*, vol. 16, no. 3, pp. 185–202, 1994.

[12] T. Dayar and N. Akar, "Computing moments of first passage times to a subset of states in markov chains," *SIAM J. Matrix Anal. Appl.*, vol. 27, no. 2, pp. 396–412, 2005.

[13] M. Bittner, P. Meltzer, and et. al, "Molecular classification of cutaneous malignant melanoma by gene expression profiling," *Nature*, vol. 406, no. 6795, pp. 536–450, 2000.

[14] R. Pal, I. Ivanov, A. Datta, M. Bittner, and E. R. Dougherty, "Generating boolean networks with a prescribed attractor structure," *Bioinformatics*, vol. 21, no. 21, pp. 4021–4025, 2005.