

# On approximate stochastic control in genetic regulatory networks

B. Faryabi, A. Datta and E.R. Dougherty

**Abstract:** The control of probabilistic Boolean networks as a model of genetic regulatory networks is formulated as an optimal stochastic control problem and has been solved using dynamic programming; however, the proposed methods fail when the number of genes in the network goes beyond a small number. There are two dimensionality problems. First, the complexity of optimal stochastic control exponentially increases with the number of genes. Second, the complexity of estimating the probability distributions specifying the model increases exponentially with the number of genes. We propose an approximate stochastic control method based on reinforcement learning that mitigates the curses of dimensionality and provides polynomial time complexity. Using a simulator, the proposed method eliminates the complexity of estimating the probability distributions and, because the method is a model-free method, it eliminates the impediment of model estimation. The method can be applied on networks for which dynamic programming cannot be used owing to computational limitations. Experimental results demonstrate that the performance of the method is close to optimal stochastic control.

## 1 Introduction

Systems biology studies the multivariate interaction among biological components, for example, genes and proteins. From a translational perspective, the ultimate objective of genetic regulatory network modelling is to use the network model to design strategies for influencing system dynamics; for instance, in the case of a tumour, diminishing the long-run likelihood of metastasis. The first proposed intervention strategies involve resetting the state of the network, as necessary, to a more desirable initial state and letting the network evolve from there [1], or changing the steady-state (long-run) probability distribution of the network by minimally altering its rule-based structure [2]. Subsequent to these early proposals, the major effort has 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 [3, 4]. Given a cost function related to system behaviour, a control policy is derived from the model that is optimal with respect to the cost function. Such an approach is natural, because it uses systems theory to develop systems-based therapy.

To date, optimal stochastic regulatory intervention has been studied in the context of probabilistic Boolean networks (PBNs), which have been recently proposed as a paradigm for studying gene regulatory networks [5]. These networks, which allow the incorporation of uncertainty into the inter-gene relationships, are essentially probabilistic generalisations of the standard Boolean networks

introduced by Kauffman [6–8]. In a PBN, gene values are quantised into some finite discrete range. The values are updated synchronously at each time step according to regulatory functions. Stochasticity is introduced into the model by allowing several possible regulatory functions for each gene and allowing random perturbation. If the regulatory functions are allowed to change at every time point, then the PBN is said to be instantaneously random. On the other hand, in a context-sensitive PBN, function updation only occurs at time points selected by a binary switching random variable. The intent of a context-sensitive PBN is to incorporate the effect of latent variables outside the model, whose behaviours influence regulation within the model. In essence, the PBN is composed of a collection of networks, and between switches it acts like one of the constituent networks, each being referred to as a ‘context’. Control theory is applied to a PBN via the transition matrix for its state-space. To overcome dimensionality obstacles, we develop here an approximation method for the application of an infinite-horizon control policy in the framework of PBNs.

Since the dynamical behaviour of a PBN is described by a Markov process [1], the theory of optimal stochastic control is used to find an optimal sequence of interventions. This has been accomplished in various ways. In the first proposed control method, after formulation of the optimal stochastic control problem, dynamic programming is employed to find a finite-horizon optimal sequence of interventions [3]. Although the proposed method theoretically formulates the control method for PBN models, this work reveals the limitation of optimal stochastic control methods in complex systems: it is noted that designing a controller for the original network is beyond the available computational capacity, and a reduced seven-gene model is considered instead of the original ten-gene model. The optimal control policies of [3] assume an instantaneously random PBN, in which the gene regulatory functions are randomly selected at each time step. In [9], the finite-horizon procedure is extended to a context-sensitive PBN.

© The Institution of Engineering and Technology 2007

doi:10.1049/iet-syb:20070015

Paper first received 26th March and in revised form 18th June 2007

The authors are with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, 77843, USA

E.R. Dougherty is also with the Translational Genomics Research Institute, 400 North Fifth Street, Suite 1600, Phoenix, AZ 85004, USA

E-mail: bfariabi@ece.tamu.edu

Subsequently, it has been demonstrated that the finite-horizon optimal stochastic control of a PBN is an NP-hard problem [10]. In general, it is well known that the direct application of optimal control methods is limited by the size of the state-space – the curse of dimensionality [11]. Hence, the optimal methods such as the ones proposed in [3, 9] are only applicable to models with small state-spaces. To address this complexity issue, an approximate stochastic control method has recently been proposed for controlling a PBN in a finite-horizon [12].

Although controlling a regulatory network over a finite number of stages lowers the likelihood of visits to undesirable states within the control window, it may not alter the likelihood of visiting undesirable states in the long run. To address this issue, the theory of infinite-horizon optimal stochastic control has been employed to find stationary policies that affect the steady-state distribution of a PBN [13]. It is also known that the complexity of these kinds of methods increases exponentially with the size of the model's state-space [11]. Therefore they can only be applied to small network models. Owing to this limitation, the numerical experiments in [13] are designed for the seven-gene model presented in [3].

For larger biological models involving interactions among many genes, a stochastic control method that has polynomial time complexity is needed. To this end, we propose an approximate stochastic control method for context-sensitive PBNs. A reinforcement learning method is used to overcome the curse of dimensionality. The proposed approximate method can yield a near-optimal stationary policy, when possessing polynomial time complexity.

A second advantage of the proposed method is that it is model-free [14]. To date, the proposed optimal stochastic control methods for PBNs have been model-dependent: the probability distributions of the Markov processes representing a PBN are required. The computational complexity of estimating these distributions increases exponentially with the number of genes in the model. Datta *et al.* [3] use ratios involving the coefficient of determination as estimates for the transition probabilities to overcome this problem. Because it is model-free, the proposed reinforcement learning method does not require estimates of the transition probabilities of the context-sensitive PBN. Given the constituent Boolean networks and the distributions that define the variation in the model context, the proposed reinforcement learning method determines an approximate stationary policy.

The paper is organised as follows. In Section [2], we formulate the problem of controlling a context-sensitive PBN as a Markov chain with reward. The reinforcement learning method, Q-learning, is formulated in Section [3]. In Section 4, we formulate a ten-gene context-sensitive PBN model for a melanoma case study [15]. Using reinforcement learning, we determine the approximate stationary policy and the corresponding steady-state distribution of the gene-activity profile. It is investigated how the approximate stationary policy performs in comparison to the optimal stationary policy.

## 2 Systems and methods

A context-sensitive PBN consists of a set  $V = \{x_1, \dots, x_n\}$ , of  $n$  nodes, where  $x_i \in \{0, 1, \dots, d-1\}$ , and a set  $\{f_1, f_2, \dots, f_k\}$  of vector-valued functions, called predictor functions. In the framework of gene regulation, each  $x_i$ , for  $i = 1, \dots, n$ , represents the expression value of a gene. It is common to mix terminology by referring to  $x_i$  as the  $i$ th gene. Each vector-valued function  $f_i$ , which has the form

of  $f_i = (f_{i1}, \dots, f_{in})$ , determines a constituent network of the context-sensitive PBN. Each function  $f_{il}: \{0, \dots, d-1\}^n \rightarrow \{0, \dots, d-1\}$ , for  $i = 1, \dots, n$ , is a predictor of gene  $i$ , if network  $l$  is selected. At each time step, a decision is made whether to switch networks. The switching probability,  $q$ , is a system parameter. If it is decided that 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 selection distribution  $\{r_1, \dots, r_k\}$ . After selecting the predictor function  $f_i$ , the values of genes are updated accordingly, that is, according to the network determined by  $f_i$ . We consider context-sensitive PBNs with perturbation, in which each gene may change its value with small perturbation probability,  $p$ , at each time unit. Such a perturbation model enables us to capture a realistic situation where the activity of a gene undergoes a random alteration. As we will see later, in addition, it guarantees that the Markov chain modelling the gene regulatory network has a unique steady-state distribution, a property which significantly aids the development in this sequel.

To date, PBNs have been applied with  $d = 2$  or  $d = 3$ . If  $d = 2$  (binary), then the constituent networks are Boolean networks, with 0 or 1 meaning OFF or ON, respectively. The case  $d = 3$  (ternary) arises when we consider a gene to be downregulated (0), upregulated (2) or invariant (1). This situation commonly occurs with cDNA microarrays, where a ratio is taken between the expression values on the test channel (usually red) and the base channel (usually green). The biological example considered in this paper has  $d = 2$  so that gene values are either 0 or 1; however, there is no theoretical restriction to binary values and therefore the methodology in this paper is developed for a general  $d$ . In particular, the method could be applied to ternary-valued networks. Indeed, the purpose of this paper is to approximate optimal control for PBNs in situations where the optimal dynamic programming solution is computationally too complex.

The gene-activity profile (GAP) is an  $n$ -digit binary vector  $\mathbf{x}(t) = (x_1(t), \dots, x_n(t))$  giving the expression values of genes at time  $t$ , where  $\mathbf{x}(t) \in \{0, \dots, d-1\}^n$ . The GAP,  $\mathbf{x}(t)$ , can be represented in decimal form by

$$z_t = \sum_{j=1}^n d^{n-j} x_j(t)$$

where  $z_t$  takes values from 0 to  $d^n - 1$ .

In the presence of external control, we suppose that the PBN has  $m$  binary control inputs:  $c_1(t), \dots, c_m(t)$ . A control  $c_i(t)$  can take binary values in  $\{0, 1\}$  at each time  $t$ . The decimal representation

$$u_t = \sum_{j=1}^m 2^{m-j} c_j(t)$$

of the control vector  $(c_1(t), \dots, c_m(t))$ , where  $u_t \in \mathcal{C} = \{0, 1, \dots, 2^m - 1\}$ , describes the complete status of all the control inputs.

There are two basic ways to associate a Markov chain with a context-sensitive PBN. One is to make the states of the Markov chain consist of ordered pairs, a GAP and a constituent network; the other is to omit the constituent network and to make the GAPs the states of the Markov chain. In practice, when dealing with control we observe a single GAP and the network emitting the GAP will not be detected [9]. In this situation, the system evolution can be modelled

by a stationary discrete-time equation

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

where the state  $z_t$  is an element of the state-space  $\mathcal{S} = \{0, 1, \dots, d^n - 1\}$ . The disturbance  $w_t$  is the manifestation of uncertainties in the context-sensitive PBN, due either to network switching or a change in state resulting from a random gene perturbation. It is assumed that both the gene perturbation distribution and the network switching distribution are independent and identical for all time steps  $t$ . The PBN is then modelled as a Markov chain with  $d^n$  states, the state  $z_t$  at any time step  $t$  being a GAP. Originating from a state  $i$ , the successor state  $j$  is selected randomly within the set  $\mathcal{S}$  according to the transition probability  $p_{ij}(u)$

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

for all  $i$  and  $j$  in  $\mathcal{S}$  and for all  $u$  in  $\mathcal{C}$ . Gene perturbation ensures 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].

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

$$\bar{r}(i, u) = \sum_{j \in \mathcal{S}} p_{ij}(u) r(i, u, j)$$

For the numerical studies in Section 4, we assume that the reward  $r(i, u, j)$  depends on the successor state  $j$  and the control  $u$ , and is independent of the present state  $i$ . This assumption is motivated from the fact that the reward is low if the destination state is an undesirable state, and high if the destination state is desirable.

To define the expected total reward, we consider the discounted reward formulation. The discounting factor,  $\lambda \in (0, 1)$ , ensures the convergence of the expected total reward over the long run [11]. Including a discounting factor in the expected total reward signifies that the incurred reward at a later time is less significant than the incurred reward at an earlier time. In the case of cancer therapy, the discounting factor emphasises that the reward of obtaining treatment at an early stage is better than at a later stage.

Among all admissible policies  $\Pi$ , the infinite-horizon optimal stochastic control methodology finds a policy  $\pi = \{\mu_0, \mu_1, \dots\}$ , where  $\mu_t: \mathcal{S} \rightarrow \mathcal{C}$  is the decision rule at time step  $t$ , that maximises the expected total discounted reward. The infinite expected total discounted reward, given the policy  $\pi$  and the initial state  $i$ , is

$$J_\pi(i) = \lim_{N \rightarrow \infty} E \left\{ \sum_{t=0}^{N-1} \lambda^t r(i, \mu_t(i), j) \right\} \quad (1)$$

The vector of accumulated rewards  $\mathbf{J}_\pi = (J_\pi(0), \dots, J_\pi(d^n - 1))$ , is called the value function. We seek a policy that maximises 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 reward

$$J^*(i) = \max_{\pi \in \Pi} J_\pi(i) \quad \forall i \in \mathcal{S} \quad (2)$$

A stationary policy is an admissible policy of the form  $\pi = \{\mu, \mu, \dots\}$ . The vector  $\mathbf{J}_\mu$  is its corresponding value function. The stationary policy  $\pi$  is optimal if  $J_\mu(i) = J^*(i)$  for any state  $i$ . It is known that an optimal

policy exists for the discounted infinite-horizon stochastic optimal control problem, and it is given by the fixed point solution of the Bellman equation. Moreover, an optimal policy determined by the Bellman equation is also a stationary policy [11]. A dynamic programming algorithm is used to iteratively find the fixed point of the Bellman equation. At each iteration, the value function must be computed for all states in the state-space. Hence, the computational complexity of the method exponentially increases with the number of genes.

### 3 Algorithm

#### 3.1 Reinforcement learning for context-sensitive PBNs

If the system and cost structure can be simulated, then it is possible to use repeated simulation to calculate approximate transition probabilities and an expected immediate reward. Thereafter, dynamic programming methods such as the value iteration algorithm can be applied to find an optimal control policy. We assume that the distributions governing the PBN, the switching probability, the perturbation probability and the probability distribution of selecting constituent networks, are known. The complexity of estimating the transition probabilities and the complexity of dynamic programming exponentially increase as the number of genes increases. If we contemplate approximation to reduce the complexity, then reinforcement learning can be used. Given the aforementioned distributions, a reinforcement learning algorithm progressively computes the value function of a given policy by generating several sample trajectories of the PBN and their associated costs. Hence, it eliminates the computational complexity associated with the explicit estimation of the transition probabilities.

The fixed point of Bellman's optimality equation is an optimal policy of (2), so for each  $i \in \mathcal{S}$  the optimal value function is the solution of

$$J^*(i) = \max_{u \in \mathcal{C}} \left[ \bar{r}(i, u) + \lambda \sum_{j=0}^{d^n-1} p_{ij}(u) J^*(j) \right] \quad (3)$$

Numerical algorithms, such as the value iteration, iteratively apply the transformation derived by the Bellman optimality equation to each element of the value function until a fixed point of (3) is found [11]. Therefore the complexity of value iteration is exponential in the number of genes in the PBN. The computational complexity of each iteration of the value iteration is  $O(d^{2n} 2^m)$ , with respect to the number of genes in the network,  $n$ , and the number of binary control inputs,  $m$ . Using the definition of expected immediate reward, the Bellman optimality equation can be rewritten as follows: for each state  $i \in \mathcal{S}$ ,

$$J^*(i) = \max_{u \in \mathcal{C}} \left[ \sum_{j=0}^{d^n-1} p_{ij}(u) (r(i, u, j) + \lambda J^*(j)) \right]$$

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

$$Q(i, u) \triangleq \sum_{j=0}^{d^n-1} p_{ij}(u) (r(i, u, j) + \lambda J^*(j)) \quad (4)$$

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

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

$k \leftarrow 0$   
 Setting  $\epsilon > 0$   
 Selecting an arbitrary initial Q-factor vector,  $Q^0$   
**repeat**  
    $k \leftarrow k + 1$   
   **Q-factors updating:** Computing the transformation (5) for all  $i \in \mathcal{S}$ .

$$Q^{(k)}(i, u) \leftarrow \sum_{j=0}^{d^n-1} p_{ij}(u) \left[ r(i, u, j) + \lambda \max_{u' \in \mathcal{C}} Q^{(k-1)}(j, u') \right]$$

**Checking the convergence:** Computing the following for all  $i \in \mathcal{S}$

$$\begin{aligned} J^{(k)}(i) &= \max_{u \in \mathcal{C}} Q^{(k)}(i, u) \\ J^{(k-1)}(i) &= \max_{u \in \mathcal{C}} Q^{(k-1)}(i, u) \end{aligned}$$

**until**  $\|J^{(k)} - J^{(k-1)}\|_{\infty} < \epsilon$

**Finding the optimal policy:** Choose the stationary policy for all  $i \in \mathcal{S}$

$$\mu(i) = \arg \max_{u \in \mathcal{C}} Q^{(k)}(i, u)$$

**Fig. 1** Q-factor version of the value iteration algorithm

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

$$Q(i, u) = \sum_{j=0}^{d^n-1} p_{ij}(u) [r(i, u, j) + \lambda \max_{u' \in \mathcal{C}} Q(j, u')] \quad (5)$$

Using the Q-factor, one can rewrite the value iteration algorithm in the form of Fig. 1, in which the value function is replaced by the Q-factor vector.

Estimation of the Q-factor is the objective of the presented value iteration algorithm, in which a Q-factor is an average of a random variable  $\Psi(i, u)$ . Equation (4) can be expressed as

$$Q(i, u) = E[r(i, u, j) + \lambda \max_{u' \in \mathcal{C}} Q(j, u')] = E[\Psi(i, u)]$$

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

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

denote the estimation of  $E[\Psi(i, u)]$  using  $k$  samples, where  $\psi_i$  is the  $i$ th sample of the random variable  $\Psi(i, u)$ . Upon a new observation of  $\Psi(i, u)$ , the value of  $\bar{\Psi}_k(i, u)$  can be updated by

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

If  $\alpha^{k+1} = 1/(k+1)$ , then

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

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

$$\begin{aligned} Q^{(k+1)}(i, u) &\leftarrow (1 - \alpha)Q^{(k)}(i, u) + \alpha[r(i, u, j) \\ &\quad + \lambda \max_{u' \in \mathcal{C}} Q^{(k)}(j, u')] \end{aligned} \quad (6)$$

The revised value iteration algorithm in which the Q-factors are updated according to (6) is called the Q-learning algorithm [14]. Since the transformation (6) is

independent of the transition probabilities of the system, the Q-learning algorithm is a model-free algorithm. In Q-learning, 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's simulator, given the control  $u$  is selected randomly among all the possible controls,  $\mathcal{C}$ . The term  $v(i, u)$  denotes the number of times the state-control pair  $(i, u)$  is visited. We define the step size  $\alpha_k$  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's simulator generates trajectories of state-control pairs; hence some Q-factors may be updated more often than others. An appropriate step size,  $\alpha_k$ , is needed to guarantee the convergence of the Q-learning algorithm to the optimal control strategy despite the asynchronous updating of the Q-factors [14]. Several step sizes are proposed with the general form  $C/(a+k)$ , where  $C$  and  $a$  can be any positive constants [14]. As a general rule, the step size should be small and diminish to zero at a suitable rate [16]. 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. The Q-learning algorithm is summarised in Fig. 2.

In Fig. 2, the complexity of each iteration is  $O(2^m)$  with respect to  $n$ , the number of genes in the network, and  $m$ , the number of binary control inputs. Hence, Q-learning runs in polynomial time complexity with respect to the number of genes in the network.

Moreover, Q-learning reduces the memory complexity of an optimal algorithm such as the value iteration. In Fig. 2, the values of the Q-factors are stored explicitly in a tabular form. The algorithm requires  $O(m d^m)$  memory units. Since the number of binary control inputs is small, the memory complexity is on the order of  $O(d^m)$  memory units, whereas in the value iteration algorithm the required memory is  $O(d^{2n})$  memory units. This latter quantity stems from the fact that we must store  $d^n$  values of the value function for  $m$  control inputs along with  $d^{2n}$  entries of the transition probability matrices at each iteration of the algorithm. Given the number of binary control inputs is small, the required memory of the value iteration algorithm,  $O(d^{2n} + m d^m)$ , has the growth of  $O(d^{2n})$ . Consequently, the memory complexity of the value iteration algorithm is

$Q(i, u) \leftarrow 0$  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's simulator, the successor state  $j$  as well as the earned reward-per-stage  $r(i, u, j)$  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 (6).  
         
$$Q(i, u) \leftarrow (1 - \alpha) Q(i, u) + \alpha \left[ r(i, u, j) + \lambda \max_{u' \in \mathcal{C}} Q(j, u') \right]$$
  
     Setting  $i \leftarrow j$   
**end for**  
**Finding the suboptimal policy:** Choose the suboptimal policy for all  $i \in \mathcal{S}$   
         
$$\mu(i) = \arg \max_{u \in \mathcal{C}} Q(i, u)$$

**Fig. 2** *Q-learning algorithm*

considerably reduced. The high memory complexity of the value iteration algorithm, as well as other dynamic programming algorithms, contributes to their limited direct applicability to intervention problems.

If all the state-control pairs,  $(i, u)$ , are visited infinitely often then for each state-control pair the estimated expected value,  $\bar{\Psi}_k(i, u)$ , converges to its ensemble average,  $E[\Psi(i, u)]$ , with probability 1. 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 [16], and our numerical results in Section 4 reveal this fact for a special case. In other words, 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. The maximum size of the intervention problem which can be solved by our approximate method is hardware-dependent. For instance, our current hardware configuration (single Xeon processor and 1-GB memory) can obtain near-optimal intervention policy within  $10^7$  learning periods for a synthetic 15-gene regulatory network. Given more memory and processing power, an accurate intervention strategy can be determined for significantly larger networks within reasonable time. Hence, the computation time is not an issue for us. In the application of interest, the goal is not to model fine-grained molecular interactions among a host of genes, but rather to model a limited number of genes, typically with very coarse quantisation, whose regulatory activities are significantly related to a particular aspect of a specific disease, such as metastasis in melanoma [4, 17]. The proposed Q-learning algorithm is easily up to the task of handling the limited size networks with which we are dealing.

## 4 Implementation

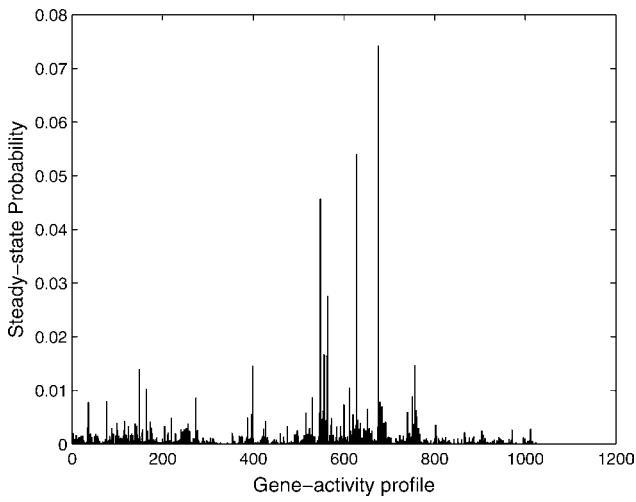
In this section, we apply Q-learning to control a network relating to metastasis in melanoma and compare the

performance of the Q-learning control algorithm to that of optimal control. Here, we only consider a ten-gene network with 1024 states, because our objective is to investigate how the approximate stationary policy performs in comparison to the optimal stationary policy, where computing the optimal stationary policy for networks beyond ten genes was not possible with our current computational capability. We will close with concluding remarks.

### 4.1 Control of a melanoma-related network via the Q-learning algorithm

A study of gene-expression data from melanoma patients has revealed that the abundance of mRNA for the *WNT5A* gene is a highly discriminating factor for cells associated with high metastatic competence against those with low metastatic competence [18]. A subsequent study shows that increasing the level of the Wnt5a protein product of the *WNT5A* gene via genetic engineering methods alters the metastasising state of melanoma cells [19]. This study also reveals that inactivating the Wnt5a protein by blocking its receptor with the appropriate antigen substantially reduces the metastatic phenotype of melanoma cells. Therefore a control strategy which reduces the level of the Wnt5a protein is desirable.

After quantifying the multivariate relationships of 587 genes among a sample of melanoma patients, a previous study constructed a ten-gene regulatory network involving *WNT5A* [15]. Using context-sensitive PBNs consisting of seven of these genes, subsequent studies developed finite-horizon [9] and infinite-horizon [13] intervention strategies. The reduction to seven genes was dictated by computational requirements. In particular, the transition probabilities of the Markov chain associated with a context-sensitive PBN are required in the optimal stochastic control methods presented in these former studies. Owing to exponentially increasing complexity of dynamic programming with the number of genes, approximation of a ten-gene PBN with a seven-gene PBN is a way to make dynamic programming feasible. Moreover, the complexity of estimating transition

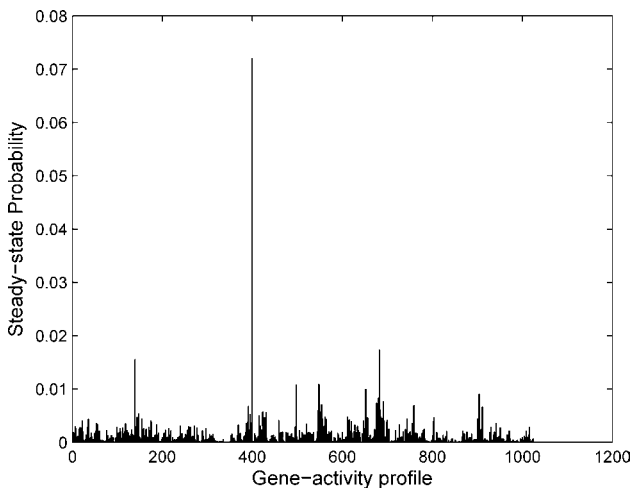


**Fig. 3** Steady-state distribution of gene-activity profile of the ten-gene PBN prior to intervention

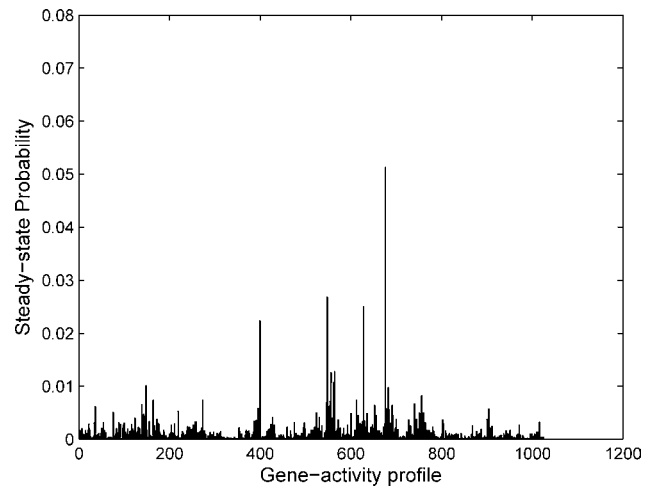
probabilities restricts the number of constituent networks in the PBN.

We consider a ten-gene network consisting of *WNT5A*, *Pirin*, *SI00P*, *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. This order of genes in the gene-activity profile facilitates the presentation of our results and does not affect the computed stationary policy. A modification of the algorithm described in [20] is used to infer ten Boolean networks that constitute the context-sensitive PBN. To arrive at a PBN resembling a cancerous situation, we have selected Boolean networks whose states with upregulated *WNT5A* possess larger aggregated probability. Thus, control of the PBN resembles a therapeutic 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.

Having the downregulation of *WNT5A* as the objective, we apply the Q-learning described in Fig. 2 to the inferred context-sensitive PBN. Here we consider only a single control  $u$ . If the control is high,  $u = 1$ , then the state of



**Fig. 4** Steady-state distribution of gene-activity profile after intervention with optimal control policy



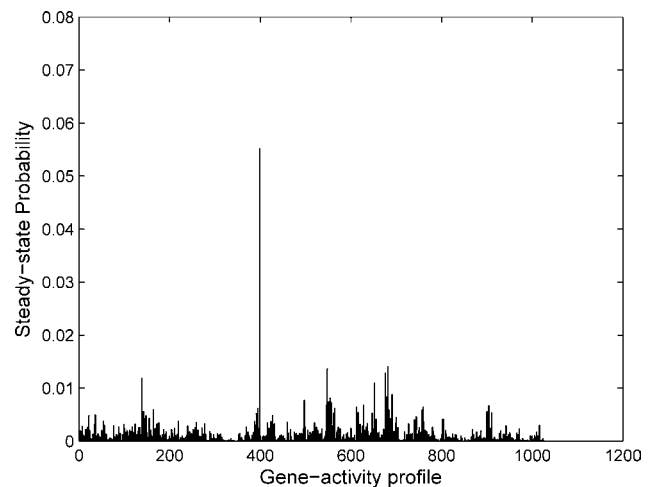
**Fig. 5** Steady-state distribution of gene-activity profile after applying the approximate control policy computed by the Q-learning algorithm with  $k_{max} = 10^3$

gene *Pirin* is reversed; if  $u = 0$ , then the state of *Pirin* remains unchanged. *Pirin* has been chosen as the control gene owing to its effectiveness in previous studies for down-regulating the expression of *WNT5A* [13].

A reward-per-stage,  $r(i, u, j)$ , is used to compare the Q-learning approximate control policy with the optimal control policy. It is assumed that the reward-per-stage is independent of the current state  $i$ . It is higher, if in the successor state  $j$ , *WNT5A* is downregulated. It is also assumed that whenever the control is applied, given *WNT5A* in the successor state remains the same, the reward-per-stage is lower in comparison to when the control is not applied. The rewards are assigned in such a way that applying the control to prevent the undesirable states is preferable in comparison to not applying control and transiting to an undesirable state. We postulate the following reward-per-stage function

$$r(u, j) = \begin{cases} 6 & \text{if } u = 0 \text{ and } WNT5A = 0 \text{ for state } j \\ 1 & \text{if } u = 0 \text{ and } WNT5A = 1 \text{ for state } j \\ 5 & \text{if } u = 1 \text{ and } WNT5A = 0 \text{ for state } j \\ 0 & \text{if } u = 1 \text{ and } WNT5A = 1 \text{ for state } j \end{cases}$$

The values have been chosen to be in line with earlier studies [13]. In practice, the reward values will have to



**Fig. 6** Steady-state distribution of gene-activity profile after applying the approximate control policy computed by the Q-learning algorithm with  $k_{max} = 10^5$

**Table 1: Steady-state probability of the most probable gene-activity profile prior and after intervention**

Gene expression	No-cont.	Opt-cont.	$k_{\max}$			
			$10^3$	$10^4$	$10^5$	$10^7$
Max prob. gene exp.	676	399	676	399	399	399
Gene exp. 676 prob.	0.07	0.007	0.05	0.02	0.01	0.009
Gene exp. 399 prob.	0.01	0.07	0.02	0.04	0.07	0.07

mathematically capture the benefits and costs of intervention and the relative preference of states, and may have to be set by physicians in accordance with their clinical judgement. Although this is not feasible within the domain of current medical practice, we do believe that such an approach will become increasingly mainstream once engineering approaches are demonstrated to yield significant benefits in translational medicine.

Assuming the preceding reward-per-stage function, we compute the optimal control policy for the ten-gene PBN. With ten constituent Boolean networks, estimation of the transition probabilities takes more than 3 days with our current hardware configuration (single Xeon processor and 1-GB memory). In this hardware configuration, going beyond the binary-valued PBN with 1024 states to a ternary-valued PBN with 59 049 states enormously increases the estimation time of the transition probabilities. Fig. 3 depicts the steady-state distribution of the gene-activity profile when there is no intervention. On the basis of Fig. 1, the aggregated probability of the gene-activity profile with upregulated *WNT5A* is higher than the gene-activity profile with downregulated *WNT5A*. Also, the most probable undesirable gene-activity profile, 676, has the highest probability.

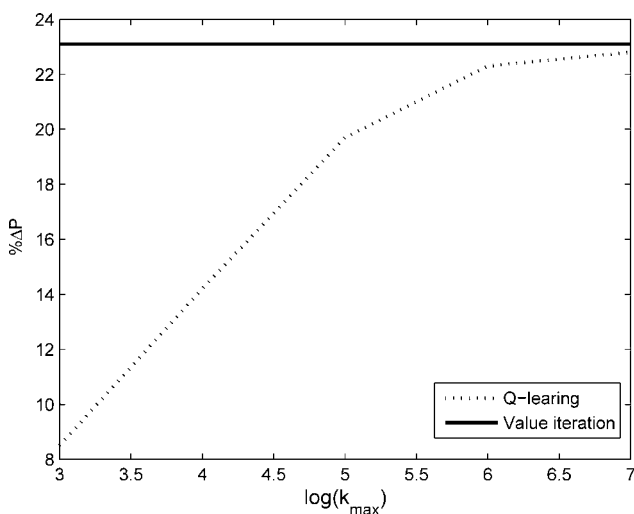
After controlling the PBN based on the optimal control policy, the steady-state distribution of the gene-activity profile is modified. According to Fig. 4, the probability of desirable states with downregulated *WNT5A* is increased and the probability of the most probable undesirable gene-activity profile, 676, is reduced to 0.007. After intervention with the optimal control policy, the most probable gene-activity profile is 399, which has downregulated *WNT5A*. We define  $\Delta P$  to be the percentage of change in the aggregated probability of the gene-activity profile with *WNT5A* = 1 before and after the intervention. As a performance measure,  $\Delta P$  indicates the percentage of reduction in the

total probability of the undesirable gene-activity profile in the steady state. For the optimal control policy, determined by the value iteration, we have  $\Delta P = 23.1\%$ .

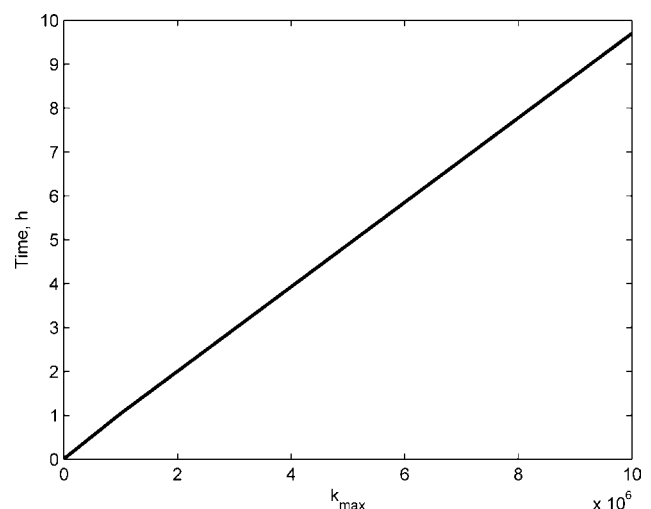
In order to compare the Q-learning algorithm with the value iteration, we execute Q-learning for different learning durations,  $k_{\max}$ . The approximate control policy is used to find the steady-state distribution of the gene-activity profile. Figs. 5 and 6 show the steady-state distributions of the gene-activity profile when  $k_{\max} = 10^3$  and  $k_{\max} = 10^5$ , respectively. The computed policy after only a short learning duration,  $k_{\max} = 10^3$ , does not reduce the likelihood of an undesirable gene-activity profile, but when the algorithm's learning duration increases,  $k_{\max} = 10^5$ , the aggregated probability of undesirable gene-activity vectors is reduced. Comparing Figs. 6 and 4, we observe that the probability distributions of the gene-activity profile are similar.

Table 1 compares the steady-state probabilities of the two gene-activity profiles with the highest probability before and after intervention. The steady-state probability of the gene-activity profile, 676, with the highest probability prior to any intervention reduces by almost the same amount when either the optimal control policy or the approximate control policy computed after a sufficient learning duration is used. Moreover, after intervention with either the optimal control policy or the approximate control policy, the desirable gene-activity profile 399 has the highest steady-state probability.

As the duration of learning in the Q-learning algorithm increases, its performance gets closer to that of the optimal control algorithm. Fig. 7 shows the value of  $\Delta P$  for the optimal control policy, as well as this value for approximate control policies derived by the Q-learning algorithm with various learning durations. The performance of approximate stochastic control converges to optimal stochastic control. We expect to observe this behaviour



**Fig. 7**  $\Delta P$  of approximate control policy against the optimal control policy as a function of logarithm of learning duration



**Fig. 8** Execution time of the Q-learning algorithm against learning duration

because as the learning duration increases, the estimate of the  $Q$ -factor vector becomes more accurate. Moreover, we observe that the time-complexity of Q-learning increases linearly with the number of iterations. Fig. 8 depicts the time it takes to run Q-learning with our current hardware configuration. The execution time of Q-learning is still tolerable when the learning duration is increased to achieve an acceptable performance for the algorithm. Hence, the melanoma case study reveals that Q-learning not only provides near-optimal performance, but also considerably reduces the time complexity and the memory complexity of optimal control algorithms. Through selecting an appropriate learning duration, we can have a trade off between the desirable accuracy of the approximate control policy and the execution time of the Q-learning.

## 4.2 Concluding remarks

We have formulated the Q-learning algorithm to find an approximate stochastic control policy for a context-sensitive PBN. Q-learning not only lowers computational complexity in comparison to the optimal stochastic control, but also performs virtually the same as the optimal stochastic control when the learning duration is long enough. As shown in the melanoma case, applying the suboptimal policy has the same effect in reducing the likelihood of visiting undesirable states, the ones with high chance of metastasis in the long run. The time complexity of the approximate control method is polynomial, whereas the time complexity of the optimal control algorithm is exponential in the number of genes. We can have a trade off between the desirable accuracy of the approximate control policy and its execution time. In addition, Q-learning is a model-free algorithm. Hence, for estimation of transition probabilities for the Markov chain modelling, the dynamics of a context-sensitive PBN is not required. Consequently, the proposed method also eliminates the time complexity of estimation processes prior to control.

## 5 Acknowledgments

This work was supported in part by the National Science Foundation (ECS-0355227, CCF-0514644 and ECCS-0701531), the National Cancer Institute (R01 CA-104620 and CA-90301) and the Translational Genomics Research Institute. The authors thank Jean-Francois Chamberland for his useful comments.

## 6 References

- 1 Shmulevich, I., Dougherty, E.R., and Zhang, W.: 'Gene perturbation and intervention in probabilistic Boolean networks', *Bioinformatics*, 2002, **18**, (10), pp. 1319–1331
- 2 Shmulevich, I., Dougherty, E.R., and Zhang, W.: 'Control of stationary behavior in probabilistic Boolean networks by means of structural intervention', *Biol. Syst.*, 2002, **10**, (4), pp. 431–446
- 3 Datta, A., Choudhary, A., Bittner, M., and Dougherty, E.R.: 'External control in Markovian genetic regulatory networks', *Mach. Learning*, 2003, **52**, (1/2), pp. 169–191
- 4 Datta, A., Pal, R., and Dougherty, E.R.: 'Intervention in probabilistic gene regulatory networks', *Current Bioinformatics*, 2006, **1**, (2), pp. 167–184
- 5 Shmulevich, I., Dougherty, E.R., Kim, S., and Zhang, W.: 'Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks', *Bioinformatics*, 2002, **18**, (2), pp. 261–274
- 6 Kauffman, S.A.: 'Metabolic stability and epigenesis in randomly constructed genetic nets', *J. Theor. Biol.*, 1969, **22**, (3), pp. 437–467
- 7 Kauffman, S.A.: 'The origins of order: self-organization and selection in evolution' (Oxford University Press, 1993, 1st edn.)
- 8 Kauffman, S.A., and Levin, S.: 'Towards a general theory of adaptive walks on rugged landscapes', *J. Theor. Biol.*, 1987, **128**, (1), pp. 11–45
- 9 Pal, R., Datta, A., Bittner, M., and Dougherty, E.R.: 'Intervention in context-sensitive probabilistic Boolean networks', *Bioinformatics*, 2005, **21**, (7), pp. 1211–1218
- 10 Akutsu, T., Hayashida, M., Ching, W.-K., and Ng, M.K.: 'Control of Boolean networks: hardness results and algorithms for tree structured networks', *J. Theor. Biol.*, 2007, **244**, (4), pp. 670–679
- 11 Bertsekas, D.P.: 'Dynamic programming and optimal control' (Athena Scientific, 1995, 2005, 3rd edn.)
- 12 Ng, M.K., Zhang, S.-Q., Ki Ching, W., and Akutsu, T.: 'A control model for Markovian genetic regulatory networks', *Trans. Comput. Syst. Biol.*, 2006, **V**, (LNBI 4070), pp. 36–48
- 13 Pal, R., Datta, A., and Dougherty, E.R.: 'Optimal infinite-horizon control for probabilistic Boolean networks', *IEEE Trans. Signal Process.*, 2006, **54**, (6), pp. 2375–2387
- 14 Bertsekas, D.P., and Tsitsiklis, J.N.: 'Neuro-dynamic programming' (Athena Scientific, 1996, 1st edn.)
- 15 Kim, S., Li, H., Dougherty, E.R., Cao, N.W., Chen, Y.D., Bittner, M., and Suh, E.B.: 'Can Markov chain models mimic biological regulation?', *J. Biol. Syst.*, 2002, **10**, (4), pp. 337–357
- 16 Tsitsiklis, J.N.: 'Asynchronous stochastic approximation and Q-learning', *Mach. Learning*, 1994, **16**, (3), pp. 185–202
- 17 Datta, A., Pal, R., Choudhary, A., and Dougherty, E.R.: 'Control approaches for probabilistic gene regulatory networks', *IEEE Signal Process. Mag.*, 2007, **24**, (1), pp. 54–63
- 18 Bittner, M., and Meltzer, P., *et al.*: 'Molecular classification of cutaneous malignant melanoma by gene expression profiling', *Nature*, 2000, **406**, (6795), pp. 536–450
- 19 Weeraratna, A.T., Jiang, Y., Hostetter, G., Rosenblatt, K., Duray, P., Bittner, M., and Trent, J.M.: 'Wnt5a signalling directly affects cell motility and invasion of metastatic melanoma', *Cancer Cell*, 2002, **1**, pp. 279–288
- 20 Pal, R., Ivanov, I., Datta, A., Bittner, M., and Dougherty, E.R.: 'Generating Boolean networks with a prescribed attractor structure', *Bioinformatics*, 2005, **21**, (21), pp. 4021–4025