

A Markovian approach to the control of genetic regulatory networks

Peter C.Y. Chen*, Jeremy W. Chen

Department of Mechanical Engineering, Faculty of Engineering, National University of Singapore, Singapore

Received 21 November 2005; received in revised form 7 December 2006; accepted 15 December 2006

Abstract

This paper presents an approach for controlling gene networks based on a Markov chain model, where the state of a gene network is represented as a probability distribution, while state transitions are considered to be probabilistic. An algorithm is proposed to determine a sequence of control actions that drives (without state feedback) the state of a given network to within a desired state set with a prescribed minimum or maximum probability. A heuristic is proposed and shown to improve the efficiency of the algorithm for a class of genetic networks.

© 2007 Elsevier Ireland Ltd. All rights reserved.

Keywords: Genetic networks; Markov chains; Minimum-cost control; Heuristic

1. Introduction

In a biological cell, genes may be expressed constantly (i.e., constitutive gene expression), or expressed based on molecular signals (i.e., regulated gene expression). The central dogma of molecular biology states that gene expression consists of two main processes, namely, transcription and translation, for prokaryotes, and with the additional step of RNA splicing for eukaryotes. During transcription, information in the relevant gene is transcribed from DNA to messenger RNA (i.e., mRNA), while during translation, the sequence of nucleotides in the mRNA is used in the synthesis of a protein.

Biological systems are complex and contain interacting sub-systems that perform various functions (Nelson and Cox, 2000). For instance, intricate forms of feedback exist in such systems. Proteins that are produced within a cell may interact among themselves and influence gene transcription, allowing protein production

on demand. An example of this is the SOS response of damage to DNA, where genes responsible for repair are down-regulated during normal operation by the molecular products of cellular functions. Another example is switching, which acts to prevent the superposition of the effect from different molecular signals when it is not useful, as in the case of the metabolism of galactose in yeast (where yeast cells preferentially metabolize glucose). In the presence of glucose, regardless of whether galactose is present or not, the genes for the metabolism of galactose are repressed.

A fundamental problem in the study of biological cellular behavior is to understand how biological activities are governed by the connectivity of genes and proteins. Such connectivity can be represented in the form of genetic regulatory networks (or simply, gene networks). A gene network consists of a group of genes that interact among themselves in order to synthesize certain products, i.e., proteins. The types and amount of proteins produced by a gene network have a fundamental effect on the development of the gene network itself, and on the biological systems with which the network interacts. The

* Corresponding author. Tel.: +65 6516 8837; fax: +65 6779 1459.
E-mail address: mpechenp@nus.edu.sg (P.C.Y. Chen).

ability to control the behavior of gene networks would have great impact on the fields of biomedicine and bio-engineering (e.g., development of new drugs and medical treatment techniques). Evidence suggests that such control is possible. An example is the behavior of the λ phage (Ptashne, 2004), a simple virus that chooses one or another mode of growth depending on extra-cellular signals. A λ phage can infect an *E. coli* cell, and would normally lie dormant within the host. When irradiated with a moderate dose of ultraviolet light, the λ phage multiplies rapidly and causes its host cell to burst. The liberated phages may infect and destroy other bacterial cells.

Understanding the behavior of gene networks is essential for developing methods for controlling such networks. Assisted by modern lab-automation technologies (such as microarrays), rapid progress has been made on gaining understanding of how gene networks function in natural biological environment (e.g., Davidson et al., 2002). Recently, researchers have also begun to exploit such understanding by designing and constructing synthetic gene networks (Hasty et al., 2002; McAdams and Shapiro, 1995; McAdams and Arkin, 2000). These networks are engineered to operate in prescribed ways so as to influence the behavior of biological systems that the networks interact with. The aim of this line of research is to develop synthetic gene networks that operate in a natural biological system in order to control the behavior of that system. For example, a synthetic oncolytic adenovirus has been engineered to detect and selectively kill tumor cells (Ramachandra et al., 2001).

A systematic approach to engineering of synthetic gene networks requires formal models for representing, and methods for controlling, the behavior of the target gene network. Formal models provide a mechanism for designers to analyze the behavior of gene networks (even before such networks are actually built) in order to determine whether the networks can achieve the desired outcome, while control methods are essential to ensure that the network will behave as desired. We note that the types of gene network models we discuss in this paper only concern the transcription process, and can be described as “phenomenological” as they do not explicitly deal with the actual and exact biological activities involved in gene expression. They are logical abstractions used to describe observed phenomena, where the presence and absence of actual proteins are represented as logical variables. The generation of such models from experimental data enables simulation of cellular dynamics, and serves as a starting point for investigating the control of biological processes.

Various approaches have been proposed in the literature for constructing dynamics models of gene networks qualitatively and quantitatively (Smolen et al., 2000). Quantitative approaches usually employ differential equations to model the dynamics of a gene network in order to examine the time-course of its evolution (e.g., Goodwin, 1965; Plahte et al., 1998; Smith, 1987). Such approaches often rely on computationally intensive numerical simulation to provide a detailed description of the behavior of the network. Qualitative approaches, on the other hand, characterize the state of a gene network in terms of discrete logical variables (e.g., Glass and Kauffman, 1973; Thomas et al., 1995; Thomas and Kaufman, 2001; Chen, 2004). Specifically, a gene is considered either on or off, and a product either present or absent. Interactions among the genes in the network are modelled by logical rules. Such approaches have yielded results that appear to confirm those obtained from quantitative approaches, and have been applied in the study of various actual biological processes, such as immunity control in bacteriophage lambda (Thieffry and Thomas, 1995), activation and anergy of T cells (Kaufman et al., 1999), and flower morphogenesis in *Arabidopsis thaliana* (Mendoza et al., 1999). There are also attempts to develop approaches that use both discrete and continuous variables in the modelling of biological systems (e.g., Duan et al., 2000; Edwards et al., 2001). In such approaches, the dynamics of a system is considered to consist of a mixture of continuous activities and discrete events (McAdams and Shapiro, 1995).

All these approaches are in general deterministic. That is, they implicitly assume that a gene network can be observed to be in a particular state with certainty. Such deterministic approaches may lead to elegant abstract models that yield “high-level” insight on the functions of gene networks, but they may not reflect the actual dynamics of the biological gene network accurately. In fact, it has been observed that transitions between states in a gene network take place stochastically (Thattai, 2001; Elowitz et al., 2002; Ozbudak et al., 2002). It has also been suggested that monitoring stochastic signals and responses in real time may be an effective means of exposing the true cell dynamics behind the population averages (Paulsson, 2004). Such results support the view that in principle the dynamics of gene networks are stochastic and their states are difficult to observe in real time, and highlight the need to develop probabilistic models of gene networks and stochastic methods for the control of such networks.

Probabilistic models of gene networks are emerging to complement the deterministic models. A major probabilistic model is probabilistic Boolean networks (PBNs)

(Shmulevich et al., 2002a,b,c) which can be considered as a special class of Markov chains (Meyn and Tweedie, 1993). PBNs are essentially probabilistic generalizations of Boolean networks, originally introduced in Kauffman (1969). In a PBN, state transitions are based on a finite number of Boolean functions, each with a given probability.

Markov chains have also been used as a generic framework for modelling gene networks. A finite state homogeneous Markov chain model has been constructed from microarray data (Kim et al., 2002), where it was found that the constructed model produced state distributions approximating biological observations and exhibited many properties associated with biological systems. This suggests that Markov chain models incorporating rule-based transitions between states are capable of mimicking biological phenomena. Such models have been shown to be useful in developing strategies for controlling (externally) the behavior of gene networks. In the investigation reported in Datta et al. (2003, 2004), gene networks are cast as controlled Markov chains, where transitions are governed by a stochastic matrix which is a function of the control input to the system. The problem is to control the gene network to reach a certain state within a finite time horizon, under the assumption that the state of the network after the application of each control action is completely or partially observable. Dynamic programming (Bellman and Dreyfus, 1962) is used to derive the optimal stationary policy, which gives (for each time step and state) the control action that leads to the best expected outcome with minimum cost over the rest of the control horizon.

Timely observation on the result of each transition in a given gene network may prove to be difficult in practice. This difficulty poses the problem of controlling a given gene network without state feedback. In this paper, an approach for dealing with this problem is presented. An algorithm is proposed to determine a sequence of control actions that drives (without state feedback) the state of a given network to within a desired state set with a prescribed minimum or maximum probability. A heuristic is proposed and shown to improve the efficiency of the proposed algorithm for a class of gene networks.

This paper is organized as follows. Section 2 presents mathematical and computational preliminaries. Section 3 formulates the problem. Section 4 presents a solution to the problem. Section 5 proposes a heuristic that improves the efficiency of the solution presented in Section 4. Section 6 presents examples for illustrating the effectiveness of the proposed approach. Section 7 discusses implications and possible extension of this work.

2. Preliminaries

A Markov chain model can be described by the four-tuple $(\mathcal{S}, \mathcal{U}, p, c)$, where \mathcal{S} is a (finite) set of states, \mathcal{U} is a (finite) set of actions, $p: \mathcal{S} \times \mathcal{U} \rightarrow [0, 1]$ is the state transition probability function, i.e., $p_{ij}^u = \mathcal{P}(s(k+1) = j | s(k) = i, u \in \mathcal{U})$ for all $i, j \in \mathcal{S}$ and $u \in \mathcal{U}$, where $s(k)$ denotes the state of the system at (discrete) time instant k , and $c: \mathcal{S} \times \mathcal{U} \rightarrow \mathcal{R}$ (where \mathcal{R} denotes the set of all real numbers) is the cost function. An action $u \in \mathcal{U}$ taken at state $s(k)$ is assumed to incur a certain fixed non-zero cost, denoted by $c(s(k), u)$. We note that all information in a time invariant/homogeneous Markov chain with n states can be encoded in a $n \times n$ stochastic matrix that tabulates the n transition probabilities for each of the n states. Hence, our approach presented below addresses time invariant/homogeneous Markov chains cast in the above form.

For a vector \mathbf{v} , let v_i represent the i th element. For a matrix \mathbf{m} , let m_{ij} represent the element at the i th row and j th column. For a finite set X , let $|X|$ denote the number of elements in X . Assuming that the $N = |\mathcal{S}|$ possible states of the system have been already enumerated, the state of the system $s(k)$ at time k can be represented as a state distribution $x_i(k)$, where $x_i(k) = \mathcal{P}(s(k) = i)$ with $i \in \mathcal{S}$. The components of $\mathbf{x}(t) \in \mathcal{R}^N$ are each mutually exclusive random variables and may be represented as the components of a (row) vector. The evolution of the state distribution can be expressed as

$$x_j(k+1) = \sum_{i=1}^N x_i(k) p_{ij}^u. \quad (1)$$

Implicit in Eq. (1) is that p_{ij}^u , for each $u \in \mathcal{U}$, are the components of a stochastic matrix \mathbf{p}^u (with row sums equal 1). We also refer to \mathbf{p}^u as the network representation.

The state distribution can be normalized such that

$$\sum_{i=1}^N x_i(k) = 1. \quad (2)$$

If the system is in a given state, it cannot be at the same time in a different state. It follows that \mathbf{x} is a partition of \mathcal{S} . Let \mathcal{D} denote the set of all normalized state distributions. The transition probability distribution function is restricted such that, for all $i, j \in \mathcal{S}$ and $u \in \mathcal{U}$:

$$\sum_{j=1}^N p_{ij}^u = 1. \quad (3)$$

From any given state distribution, the probability of the system transitioning into another possible state is bounded above and below by the maximum and mini-

num components in the relevant columns of the stochastic matrix \mathbf{p}^u . Specifically:

$$\hat{p}_j^u = \max_{\mathbf{x}(k) \in \mathcal{D}} x_j(k+1)|_u = \max_{\text{All } i} p_{ij}^u \quad (4)$$

and

$$\bar{p}_j^u = \min_{\mathbf{x}(k) \in \mathcal{D}} x_j(k+1)|_u = \min_{\text{All } i} p_{ij}^u. \quad (5)$$

The expected cost of taking an action $u \in \mathcal{U}$ at time k , denoted by $\bar{C}(k, u)$, may be defined as

$$\bar{C}(k, u) = \sum_{i=1}^N (x_i(k)c(i, u)). \quad (6)$$

We refer to an (action) sequence as a string constructed by concatenating symbols from \mathcal{U} . For a sequence σ , let $|\sigma|$ denote the length of σ (i.e., the number of symbols in σ). Let σ_i denote the i th symbol in the sequence σ , and \mathcal{U}^+ the set of all possible sequences over \mathcal{U} . Let \emptyset denote the empty set, and ϵ denote the empty string with the properties: (i) $|\epsilon| = 0$ and (ii) $\epsilon \circ u = u \circ \epsilon = u$ for any string u , where \circ denotes concatenation. Let \mathcal{U}^* be the Kleene closure over \mathcal{U} , i.e., $\mathcal{U}^* = \epsilon \cup \mathcal{U}^+$. We refer to ϵ_\circ as the null sequence, i.e., ϵ_\circ is any sequence in $\{\epsilon\}^*$.

For a given sequence of actions σ , let $\bar{C}_q(\sigma)$ denote the total expected cost incurred after the execution of σ . We are interested in the time instants when the transitions take place. Thus, we consider that $\epsilon_\circ, \sigma_1, \sigma_2, \dots, \sigma_{|\sigma|}$ correspond to the time instants $0, k_1, k_2, \dots, k_{|\sigma|}$, respectively, and write

$$\bar{C}_q(\sigma) = \sum_{i=1}^{|\sigma|} \bar{C}(k_i, \sigma_i). \quad (7)$$

Let \mathbf{x}_σ be the state distribution reached after the execution of σ starting from $\mathbf{x}(0)$. Then, $\mathbf{x}_\sigma = \mathbf{x}(k_{|\sigma|})$.

A data structure called a priority queue (denoted by Q) is used in the subsequent development. In the context of this paper, Q is a linear ordering on a finite set \mathcal{T} with terms in \mathcal{U}^* . Let $|Q|$ denote the number of elements in Q , and Q_i the i th element of Q . Q is ordered such that

$$\bar{C}_q(Q_i) \leq \bar{C}_q(Q_{i+1}), \quad 1 \leq i < |Q|. \quad (8)$$

Elements can be added or removed from Q . If $|Q| > 0$, Q_1 will be the next element to be removed (for processing). When a sequence is added, Q is sorted according to Eq. (8).

3. Problem formulation

Iterating Eq. (1) with a sequence formed from a given set of actions yields two possible dynamics for the distribution. The first is a contraction mapping with the distri-

bution approaching a stable one (i.e., a fixed point). The second takes the system through what may be best described as cycles, where there is some form of oscillation in the probabilities of a number of states.

Given such a model of a gene network, a certain set of states of the network may be considered to be of particular importance. These states may indicate certain properties of the gene network, such as a gene being expressed, a protein reaching a certain concentration level, or an undesirable property exhibited by the gene network. Thus, one may desire that the total probability of transitioning to a state in a particular set is at least a minimum value \bar{p}' , or at most a maximum value \hat{p}' . Achieving this requires controlling the behavior of the gene network through application of certain control actions at specific states of the network.

This control problem can be stated as follows. Given an initial state $s(0) \in \mathcal{S}$ with the distribution $\mathbf{x}(0) \in \mathcal{D}$ and a set of target states $\mathcal{S}' \subseteq \mathcal{S}$, find a sequence of actions σ' that results in the system reaching a target state with a minimum probability (i.e., $\sum_{i \in \mathcal{S}'} x_i \geq \bar{p}'$) or a maximum probability (i.e., $\sum_{i \in \mathcal{S}'} x_i \leq \hat{p}'$) while minimizing the total expected cost $\bar{C}_q(\sigma')$, subject to a given amount of finite resources.

This problem can be interpreted in more tangible terms as follows. An instance of controlling the system to reach a target state with a minimum probability would be finding a sequence of actions such that the probability of a given set of proteins all being produced is higher than some minimum value \bar{p}' . An instance of controlling the system to reach a target state with a probability below some maximum would be finding a sequence of actions such that the probability of any one of a given set of genes being off does not exceed some maximum value \hat{p}' . The values of \hat{p}' and \bar{p}' can be specifically set to express the desire of achieving a particular outcome with certainty. For instance, setting $\bar{p}' = 1$ indicates the desire to find a sequence that guarantees that the target state set will be reached. Conversely, setting $\hat{p}' = 0$ means to find a sequence that guarantees that the target state set will be avoided.

It is noted that in problems of practical interest, often the initial state is determined exactly. Thus, the component of the initial state distribution corresponding to the initial state would be valued 1, and the others 0.

4. The cost-first approach

It is convenient to visualize the process of solving the problem as searching a tree by expanding nodes (which represent sequences of actions) in the order of increasing cost. For instance, given a set of actions $\{\epsilon_\circ, 1, 2\}$,

the children of “ ϵ_0 ” are “1” and “2”, and the children of “121” are “1211” and “1212”. This tree will be expanded in order of cost, with sequences of lower total cost evaluated first. Assuming that all costs are positive, a child node, or child sequence, will have a higher total cost than its parent.

The solution process proceeds as follows. Beginning from the initial state distribution, longer sequences are built up upon existing sequences of actions, and their resulting state distributions computed. This process terminates when a non-zero-length list of sequences (of the same cost) that meets the requirements of the problem has been found, if a solution exists.

This solution process can be formalized into an algorithm. For a given action sequence σ , the state distribution at time $k_{|\sigma|}$ (i.e., the time when $\sigma_{|\sigma|}$ is executed) is $x_i(k_{|\sigma|})$ for $i \in \mathcal{S}$. A predicate Φ defined on \mathcal{D} is a function $\Phi : \mathcal{D} \rightarrow \{1, 0\}$. We define two such predicates $\hat{\Phi}$ and $\bar{\Phi}$ with respect to \hat{p}' and \bar{p}' as follows:

$$\hat{\Phi}(\mathbf{x}_\sigma) = \begin{cases} 1 & \text{if } \sum_{i \in \mathcal{S}} x_i(k_{|\sigma|}) \leq \hat{p}' \\ 0 & \text{otherwise,} \end{cases} \quad (9)$$

$$\bar{\Phi}(\mathbf{x}_\sigma) = \begin{cases} 1 & \text{if } \sum_{i \in \mathcal{S}} x_i(k_{|\sigma|}) \geq \bar{p}' \\ 0 & \text{otherwise.} \end{cases} \quad (10)$$

In the sequel we simply use Φ to indicate either $\hat{\Phi}$ or $\bar{\Phi}$ depending on whether the problem concerns \hat{p}' or \bar{p}' .

Let Σ be the set of sequences that meet the requirement as stated in Eqs. (9) or (10), i.e., $\Sigma = \{\sigma | \Phi(\mathbf{x}_\sigma) = 1, \sigma \in \mathcal{U}^*\}$. The following algorithm finds a solution (if it exists) to the problem.

Algorithm 1. Basic algorithm

1. Add ϵ_0 to the (empty) Q
2. From Q_1 , obtain \mathbf{x}_σ using Eq. (1)
3. If $\Phi(\mathbf{x}_\sigma) = 1$
 - o Remove Q_i if $\bar{C}_q(Q_i) > \bar{C}_q(Q_1)$ for all $i \in \{2, \dots, |Q|\}$
 - o Add Q_1 to Σ
- Otherwise
 - o Add all child sequences that do not exceed resource limitations to Q subject to Eq. (8)
4. Remove Q_1
5. If $|Q| \neq 0$, goto 2
6. If $|\Sigma| \neq 0$, return success; otherwise, report failure
7. End

This algorithm is complete and optimal. If a solution exists, it will be found (complete), and the first solution

found will be the least costly control sequence (optimal). These properties can be deduced as the algorithm (in the extreme case) evaluates every possible sequence constructed from a finite set of actions in order of increasing cost, and stops once it finds a solution. If there is no solution, it will terminate after exhausting the space of sequences that do not cost more than the resources available.

Given the assumption of probabilistic transitions, this algorithm in general cannot ensure that the system will move through specific states. However, if an oscillation (within a set of attractor states) is purely due to internal system dynamics, and is triggered whenever the system enters one of the attractor states, then it is possible to apply the proposed method to ensure that the system will reach a particular attractor state with some maximum probability. As a consequence, the system will then oscillate inside the set of attractor states.

The value of \bar{p}' and \hat{p}' also have practical implications for the proposed solution process. In general, the higher the value for \bar{p}' , or the lower the value for \hat{p}' , the longer the sequences required to solve the problem. It is also usually the case that after going through a certain sequence of actions, some state distributions are closer to achieving the objective than others. Thus, it would be advantageous to add to the algorithm some measure of how close it is to the objective of bringing $\sum_{i \in \mathcal{S}} x_i$ above \bar{p}' or below \hat{p}' from a given state distribution, so as to enable the algorithm to better prioritize its effort. A suitable heuristic can serve this purpose to improve the efficiency of the algorithm.

5. Incorporating a heuristic

A heuristic function $h(\mathbf{x}) : \mathcal{R}^N \rightarrow \mathcal{R}$ is an estimate of the future cost of the best means of achieving the objective from the current state distribution. It is required that $h(\mathbf{x}) = 0$ if $\Phi(\mathbf{x}) = 1$. A suitable $h(\mathbf{x})$ can improve the performance of the solution process for typical cases. A $h(\mathbf{x})$ that never overestimates the cost of the best solution ensures the resulting solution is still complete and optimal. Such a heuristic is admissible. In fact, cost-first search coupled with an admissible heuristic is also known as A^* search (Russell and Norvig, 2002).

A good heuristic is monotonic. This means that along any state trajectory (induced by a sequence σ) starting from the initial state distribution and ending at a state distribution that meets the requirement, the value of $(\bar{C}_q(\sigma) + h(\mathbf{x}))$ does not decrease. An intuitive reason for a non-monotonic heuristic being less useful is that, given prior information that the total cost of the best sequence to achieve the objective from a state dis-

tribution is at least some value, it is useless to learn in subsequent child sequences that the total cost would be at least a lower value than the one previously reported for a parent.

5.1. A heuristic for similar networks

In this section we present a heuristic which is effective for a class of gene networks characterized by its degree of similarity. We define a measure of similarity (denoted by L) in a gene network as a norm, i.e.:

$$L(u, u') = \sum_{i=1}^N \sum_{j=1}^N |p_{ij}^u - p_{ij}^{u'}|, \quad u, u' \in \mathcal{U}. \quad (11)$$

A larger L indicates that a given network has higher degree of dissimilarity. Naturally $L(u, u) = 0$. It is also clear that $L(u', u) = L(u, u')$.

Let $\mathcal{U}_h(k) \subseteq \mathcal{U}$ denote the set of actions that are available at time k . We define a heuristic matrix $\mathbf{B}(t) \in \mathcal{R}^{N \times N}$ as

$$B_{ij}(k) = \max_{u \in \mathcal{U}_h(k)} p_{ij}^u \quad (12)$$

for problems where the system is to reach the target state set with a minimum probability, and as

$$B_{ij}(k) = \min_{u \in \mathcal{U}_h(k)} p_{ij}^u \quad (13)$$

for problems where the system is to reach the target state set with a maximum probability. The heuristic matrix represents pseudo-actions considered to be as effective as possible towards achieving the objective. The cost for performing each of these actions is denoted by

$$c_h(s(k), u) = \min_{u \in \mathcal{U}(k)} c(s(k), u). \quad (14)$$

The heuristic function $h(\mathbf{x})$ is defined as the cost of taking the so-called “heuristic best actions” until the objective is achieved, i.e.:

$$\begin{aligned} & \sum_{b=0}^{\beta-1} \sum_{i=1}^N x_i^h(b) c_{h_i}(t+b) h(\mathbf{x}) \\ & = \sum_{b=0}^{\beta-1} \sum_{i=1}^N (x_i^h(b) c(s(k), k+b)) \end{aligned} \quad (15)$$

where

$$\mathbf{x}^h(\tau) = \mathbf{x}(k) \prod_{b=1}^{\tau} \mathbf{B}(k+b) \quad (16)$$

and $\beta = \min(\tau)$ with the condition that $\Phi(\mathbf{x}^h(\tau)) = 1$. $\mathbf{x}^h(\tau)$ will not necessarily be a normalized distribution.

This is because in general $\mathbf{B}(t)$ will not be a row stochastic matrix, i.e., Eq. (3) is generally not satisfied. However, $\Phi(\mathbf{x}^h(\tau))$ will be evaluated with the numerical values in the components of $\mathbf{x}^h(\tau)$ as if it was. It is noted that Φ in this case is defined on R^N instead of \mathcal{D} .

The heuristic function $h(\mathbf{x})$ will never overestimate the cost of achieving the objective. It will also be monotonic, as every actual action costs more than the action represented by the heuristic. If an action takes the system further away from the objective, the cost reported will increase. If an action brings the state distribution of the system closer to the objective, the actual cost would be at least that given by the cost of the heuristic action, and the system will never be closer to a state distribution that meets the requirement than what the heuristic expects. The basic algorithm presented in Section 4 is modified to include the heuristic.

Algorithm 2. Heuristic algorithm

1. Add ϵ_0 to the (empty) Q
2. Set \mathbf{x}_σ to be the state distribution corresponding to Q_1
3. If $\Phi(\mathbf{x}_\sigma) = 1$
 - Remove Q_i if $\bar{C}_q(Q_i) > \bar{C}_q(Q_1)$ for all $i \in \{2, \dots, |Q|\}$
 - Add Q_1 to Σ
- Otherwise, for each child sequence that does not exceed resource limitations
 - Determine \mathbf{x}_σ using Eq. (1)
 - Calculate $h(\mathbf{x}_\sigma)$
 - Add sequence to Q in increasing order of $(\bar{C}_q(\sigma) + h(\mathbf{x}_\sigma))$
4. Remove Q_1 from Q
5. If $|Q| \neq 0$, goto Step 2
6. If $|\Sigma| \neq 0$ return success; otherwise, report failure
7. End

This algorithm differs from the basic algorithm in that all child sequences of a given sequence are evaluated if the sequence has a lower value of $(\bar{C}_q(\sigma) + h(\mathbf{x}_\sigma))$ than other sequences. If this heuristic is not effective for the network representations corresponding to the set of available actions, it would not enhance the performance of the basic algorithm. The heuristic requires an additional matrix-vector multiplications (the same as computing the next state), until the distribution meets the requirements or the upper bound on heuristic executions imposed by cost stops the process. Noting that in real biological systems for each given state, the system does not transition to just any other state directly, but only to a much smaller subset (with respect to the state space).

Thus, what can be expected of real systems is a sparse transition matrix with $O(2^N)$ entries, allowing sparse storage and resulting in matrix-vector multiplication taking $O(2^N)$ operations.

This heuristic works well with networks that have a higher degree of similarity. Given such networks, the heuristic function is likely to be more helpful in differentiating how close state distributions are from meeting the objective. The heuristic matrix multiplied with a distribution gives a vector with components summing to a value greater or smaller than unity. If the heuristic matrix causes these components to grow (or shrink) too fast, it will not be able to effectively differentiate between a given state and another as to how close they are to a desired state. The lower the degree of similarity in a network, the greater the growth (or shrinkage) of the components, hence the less useful the heuristic.

A suitable measure is $\hat{L} = \max L(u, u')$ where $u, u' \in \mathcal{U}$. More pertinent to the choice of using the heuristic, we can define $L_h = (-N) + \sum_{i=1}^N \sum_{j=1}^N B_{ij}$, and set a criterion of incorporating the heuristic as $L_h \leq N^2\mu$, where μ is a user-defined constant. Alternatively, one may adopt a practical approach, noting (during the solution process) whether the heuristic provides an effective bias in driving the network towards the desired state set. For instance, one may set a maximum number of consecutive expansions where the heuristic gives values of τ such that the difference between the maximum and minimum of τ is smaller than a user-defined threshold.

As mentioned previously, a heuristic would be the most helpful if it takes advantage of certain properties of the problem being considered. As different classes of networks have differing properties, there will be scenarios in which a heuristic will, instead of improving performance, become just an additional computational burden, since no single heuristic is a panacea that is useful in all problems.

5.2. Extension to partially observable systems

The algorithm discussed above assumes that the state of the system after each control action cannot be timely determined with certainty, and thus considered unobservable. However, this does not preclude this algorithm from being extended for partially observable systems where a given set of states are observable and the target state set resides wholly or partially within the unobservable state set.

Since the state of the system is represented as a probability distribution, conditional probability can be applied to assign state probabilities to those state that are ob-

servable, as in the case of Datta et al. (2004). Given the information from observation, the probabilities of states that are inconsistent with the observation are set to 0, and the probabilities of the remaining states may be then obtained by normalizing the distribution. Moreover, if there are specific states of interest (that occur with greater frequency for instance), one may derive control sequences for each such state. If the system transitions to such a state, one need only to begin the appropriate control sequence corresponding to the current state. Naturally, no further control action is required if an observable target state is reached.

6. Examples

6.1. A simple example

We present an example using a hypothetical single-gene network to illustrate the application of the proposed algorithm. The gene, denoted by A , is induced by a certain biological signal. When A is expressed, a product P is produced. There are three possible forms of external control: (i) chemical signal absent, (ii) chemical signal present in low concentration, and (iii) chemical signal present in high concentration.

The networks, their representations as (row) stochastic matrices and costs are given in Figs. 1–3, with values chosen for the purpose of illustration. The states of A and P corresponding to the numerical states in the figures are as summarized in Table 1. The objective here is to ensure that the total probability of A being expressed is at least 0.8, given an initial state distribution of $\mathbf{x}(0) = [1\ 0\ 0\ 0]$. Thus, the condition for achieving the desired outcome is $x_3 + x_4 \geq 0.8$.

The solution process using the basic algorithm can be represented as a tree (as shown in Fig. 4), with the nodes identified with specific sequence of actions and state distribution after executing the sequence. Evaluation of state distributions resulted from executing the sequences is done using Eq. (1). The nodes in the tree are labeled in order of evaluation. Following the numbered steps shown in the figure, a solution is found in Step 14, with the desired sequence being “23”. After Step 14, all nodes with higher cost are not expanded,

Table 1
State of gene A and product P

State	A	P
1	Off	Absent
2	Off	Present
3	On	Absent
4	On	Present

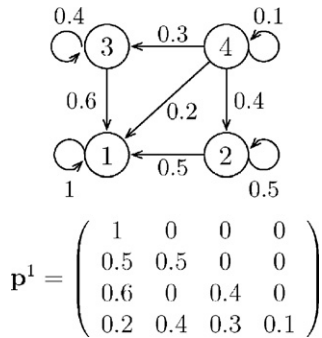


Fig. 1. Chemical signal absent; cost = 1.0.

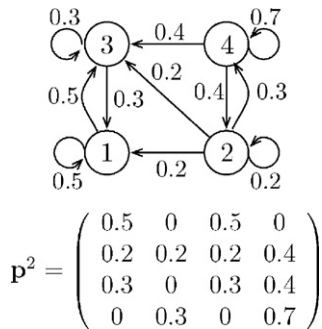


Fig. 2. Chemical signal in low concentration; cost = 1.5.

thus the algorithm continues to Step 15 before terminating. Explicitly, the desired action sequence “23” means that, starting from the initial state distribution $x(0)$, the optimal control to achieve $x_3 + x_4 \geq 0.8$ is to set the concentration of the inducer to be low initially and then increase it in the next step.

We next examine how a heuristic may affect the performance of the solution process. Considering a heuristic defined with respect to Eq. (11), we have $L(1, 2) = 4.2$, $L(1, 3) = 5.6$, and $L(2, 3) = 1.6$, hence, $\hat{L} = L(1, 3) = 5.6$ and $L_h = 2.8$. The elements of a heuristic matrix may

be defined according to Eq. (12) and its cost determined by Eq. (14), e.g.:

$$B = \begin{pmatrix} 1 & 0 & 0.7 & 0 \\ 0.5 & 0.5 & 0.3 & 0.5 \\ 0.6 & 0 & 0.4 & 0.5 \\ 0.2 & 0.4 & 0.3 & 0.9 \end{pmatrix}$$

The tree diagram in Fig. 5 illustrates the use of the algorithm with a heuristic on the single-gene problem. The value of the heuristic function evaluated at each state is given in brackets.

For comparison, the basic algorithm requires 13 evaluations to reach a solution (with the initial state as well as state 15 excluded). With the heuristic, only 2 evaluations are required, but this requires the work of 15 evaluations as in the basic algorithm, since evaluating the heuristic requires a variable number of matrix multiplications—a single evaluation in the basic algorithm requires just such multiplication.

When the additional computation associated with the heuristic is taken into account, introduction of the heuristic actually degrades the overall performance of the solution process for this particular case. However, it should be noted that this is a small problem (which we use here mainly to illustrate the solution process), while heuristics are usually designed to be used with large problems where the additional overhead can be compensated by fewer evaluations. (Incidentally, by choosing the order in which operations are performed, that is, by evaluating all the states before computing the heuristic function for each of them, the number of matrix multiplications is reduced by $(2 + 1 + 0)$ to 12, showing an actual performance improvement for this particular problem.) Furthermore, heuristics are chosen on the basis of specific problem under consideration, as there is no one best heuristic independent of the problem at hand.

6.2. A computational example

The work of Kim et al. (2002) provides evidence that even under model reduction, finite state Markov chains are able to replicate the behavior of biological systems. Their example in particular involved 587 genes using data from a survey of melanoma. Through data analysis and parameter estimation, the number of genes used to predict subsequent states was reduced to 10, which, as reported, are able to emulate the behavior of the actual gene network adequately. This represents an example that there are grounds for pursuing the control of Markovian models of biological systems with model reduction to increase tractability. In particular, our focus is

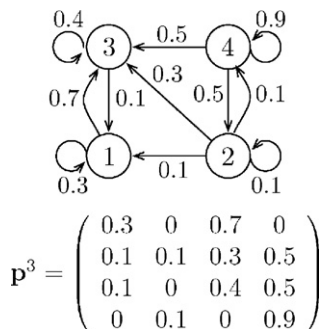


Fig. 3. Chemical signal in high concentration; cost = 2.0.

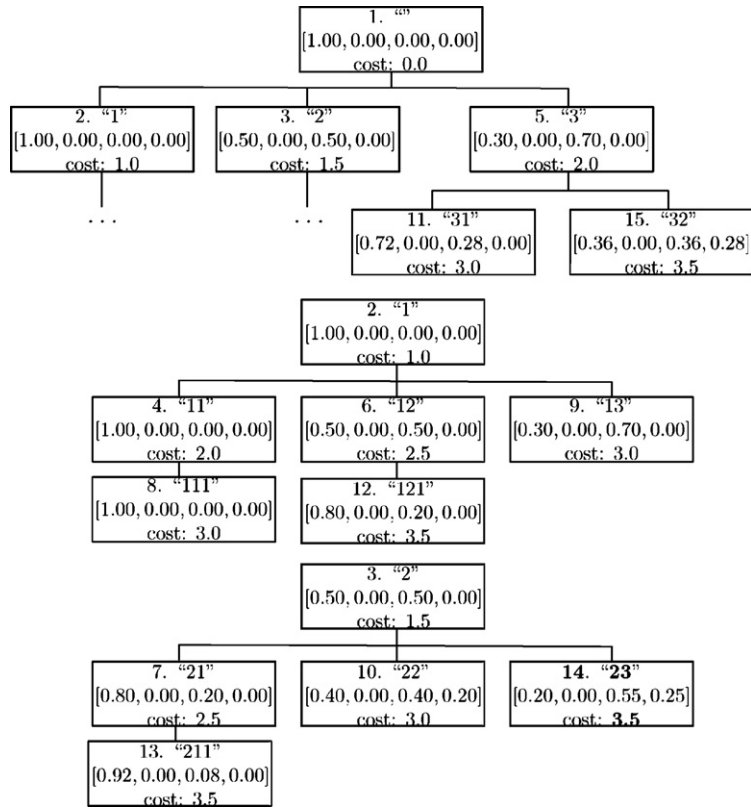


Fig. 4. Evaluation tree.

on models with their parameters estimated from data. We make no assumptions on the underlying structure of the systems under study save the model whose parameters are to be estimated.

The following computational example illustrates issues related to practical implementation of the proposed algorithm for a given gene network model. Consider a gene network with eight genes, i.e., g_i with $i = 1, 2, \dots, 8$, each may be either on or off, thus giving a total of 256 possible states. Starting from an ini-

tial state of $s(0) = [00000001]$, the objective is to find a sequence of actions (with the lowest cost) from a set of actions \mathcal{U} such that the probability that genes g_1 through g_8 are on is at least 0.8 (i.e., $\bar{p}' = 0.8$). This amounts to 8 target states out of a total state space of 256. When the elements of the stochastic matrices that each action represents were generated completely independently and randomly, the expected probability of the system transitioning to any one of the target states would be $8/256 = 0.03125$.

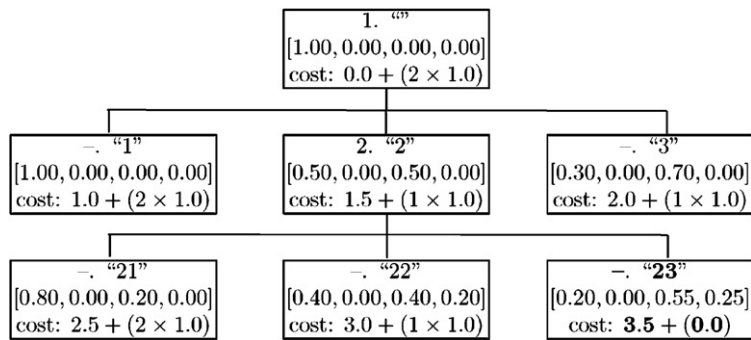


Fig. 5. Evaluation tree (with heuristic).

In this example, eight possible actions were allowed (i.e., $|\mathcal{U}| = 8$). Their corresponding network representations were randomly generated as follows. Starting with a network with states that transition deterministically back to themselves, add connections with weights much higher than 1. These connections alone would be recognizable in the graphical representation as (possibly lengthy) cycles, stable states, attractors, and deterministic transitions. Then all outgoing transitions from each node would be normalized according to Eq. (2) such that the resulting network can be represented by a stochastic matrix.

The size of the priority queue could become an important factor in the implementation of the proposed algorithm. If the optimal solution was too long, the priority queue could overflow, and thus rendering the algorithm ineffective. When using the basic algorithm, the first solution found is the optimal solution. Unfortunately this property makes the priority queue fill up much faster than in the case of the heuristic algorithm. If memory were a concern and one does not wish to use a heuristic, it may be prudent to use the heuristic algorithm while setting the output of the heuristic function to zero. However, this introduces two issues: (i) the first solution found would not always be optimal, and so (ii) additional evaluations of sequences would be necessary. The first issue is often not important as the optimal solution would be found anyway, given enough memory. As for the second issue, depending on the nature of the problem, either memory for priority queue storage or the evaluation count may take precedence.

Both versions of the algorithm were implemented and run on a Pentium 4 PC with 512 MB of RAM and a CPU speed of 2.0 GHz. Sequences of length 1 were rejected and only runs that found solutions were recorded. The performance of the algorithm was examined in the context of two sets of solutions. The first set consisted of 215 runs, in which solutions were found by both algorithms. The second set consisted of 8 runs, in which solutions were found only through the heuristic algorithm after the basic algorithm failed due to the problem of insufficient memory for the priority queue. From the first set of solutions, a least-squares estimation of the number of evaluations required when the heuristic was used (as a fraction of the number of evaluations necessary without a heuristic) gave a value of 0.32843 and a standard error of 0.00408. This implies a statistical expectation of a computational saving of about 60% for problems similar to this one, not considering the additional overhead due to the use of the heuristic.

The longest optimal sequence (found after 275,613 evaluations) was of length 7. This is an indication of the

remarkable effectiveness of the heuristic, considering the fact that there are a total of 2,396,745 sequences of length 7 and below. Hence, only a little over 13% of the possible sequences was evaluated using the heuristic before the optimal solution was found.

7. Concluding remarks

We have presented a solution to the problem of controlling (without state feedback) a Markovian gene network such that it reaches a target state set with a prescribed maximum or minimum probability. We have developed a heuristic and shown that such a heuristic leads to significant improvement on the efficiency of the proposed solution for a class of gene networks. These results represent an incremental step towards development of feasible methods for controlling gene networks.

In this study, the transition probabilities of the networks corresponding to actions taken are assumed to be known. In practice, this is usually not the case. However, for biological gene networks, much data are available from gene microarrays and other forms of measurement, and ready to be subject to data mining. Methods for the inference of network models from microarray data have been reported, such as using the method of coefficient of determination to select the genes in the data that are most likely to contribute to describing subsequent gene expression (Shmulevich et al., 2002b; Datta et al., 2003).

It is noted that the general approach of constructing probabilistic models of gene networks based on microarray data typically focuses on certain subsystems of a larger biological network, and thus how such subsystems are eventually connected together to accurately represent the dynamics of the larger network remains an important issue that needs to be addressed in the model construction process.

A problem related to representing a system by a network which enumerates all possible states is that of complexity. With the addition of a property that may take π possible values, the size of the representation increases by a factor of π . Thus, a system with N properties, with the i th property possessing π_i possible values, would have a total of $\prod_{i=1}^N \pi_i$ states. Hence, a potentially profitable route of advance would be to develop methods for discovery of the (Boolean) functional dependencies that govern state transitions using the large amounts of data garnered from experimental studies. A breakthrough in this area would, most certainly, render the approach of enumerating states obsolete, and possibly pave the way for real-time control of genetic networks. However, the problem of computational steps being exponential in the number of properties remains.

In this work, we only consider minimization of one measure of cost. This approach of minimizing a single cost can be readily extended to minimizing multiple costs to derive a pareto-optimal front of sequences that represent possible alternatives to achieve a particular goal. Naturally, suitable termination criteria would have to be found as there is no single optimal solution.

Finally, from a practical implementation perspective, the approach proposed in this paper can be further improved by (intermittently) incorporating state feedback whenever such information is available. The current approach assumes that the state of the system after each control action cannot be timely determined with certainty, and thus considered unobservable. Consequently, the proposed algorithm may need to evaluate the entire space of sequences of control actions in order to determine the desired sequences. Obviously, such an approach is computationally more intensive than those based on dynamic programming (which require state feedback). An efficient use of this approach would be to derive sequences of control actions that may be collectively applied as a single “composite” control action at certain time instants where one is able to observe the state (into which the system has transitioned) after the composite control action was taken, at which point dynamic programming can be effectively applied.

References

- Bellman, R., Dreyfus, S., 1962. *Applied Dynamic Programming*. Princeton University Press, New Jersey.
- Chen, P.C.Y., 2004. A computational model of a class of gene networks with positive and negative controls. *BioSystems* 73, 13–24.
- Datta, A., Choudhary, A., Bittner, M.L., Dougherty, E.R., 2003. External control in Markovian genetic regulatory networks. *Mach. Learn.* 52, 169–191.
- Datta, A., Choudhary, A., Bittner, M.L., Dougherty, E.R., 2004. External control in Markovian genetic regulatory networks: the imperfect information case. *Bioinformatics* 20 (6), 924–930.
- Davidson, E.H., et al., 2002. A genomic regulatory network for development. *Science* 295, 1669–1678.
- Duan, Z., Holcombe, M., Bell, A., 2000. A logic for biological systems. *BioSystems* 55, 93–105.
- Edwards, R., Aziza, K., Glass, L., 2001. Symbolic dynamics and computation in model gene networks. *Chaos* 11 (1), 160–169.
- Elowitz, M.B., Levine, A.J., Siggia, E.D., Swain, P.S., 2002. Stochastic gene expression in a single cell. *Science* 297, 1183–1186.
- Glass, L., Kauffman, S.A., 1973. The logical analysis of continuous, non-linear biochemical control networks. *J. Theor. Biol.* 39, 103–129.
- Goodwin, B.C., 1965. Oscillatory behavior of enzymatic control processes. *Adv. Enzyme Regul.* 3, 425–439.
- Hasty, J., McMillen, D., Collins, J.J., 2002. Engineered gene circuits. *Nature* 420, 224–230.
- Kauffman, S.A., 1969. Metabolic stability and epigenesis in randomly constructed genetic nets. *J. Theor. Biol.* 22, 437–467.
- Kaufman, M., Andris, F., Leo, O., 1999. A logical analysis of T cell activation and anergy. *Proc. Natl. Acad. Sci.* 96, 3894–3899.
- Kim, S., Li, H., Dougherty, E.R., Cao, N., Chen, Y., Bittner, M., Suh, E.B., 2002. Can Markov chain models mimic biological regulation? *J. Biol. Syst.* 10 (4), 337–357.
- McAdams, H.H., Shapiro, L., 1995. Circuit simulation of genetic networks. *Science* 269, 650–656.
- McAdams, H.H., Arkin, A., 2000. Towards a circuit engineering discipline. *Curr. Biol.* 10 (8), R318–R320.
- Mendoza, L., Thieffry, D., Alvarez-Buylla, E.R., 1999. Genetic control of flower morphogenesis in *Arabidopsis thaliana*: a logical analysis. *Bioinformatics* 15 (7/8), 593–606.
- Meyn, S.P., Tweedie, R.L., 1993. *Markov Chains and Stochastic Stability*. Springer-Verlag, New York.
- Nelson, D.L., Cox, M.M., 2000. *Lehninger Principles of Biochemistry*, 3rd ed. Worth Publishers, New York.
- Ozbudak, E.M., Thattai, M., Kurtser, I., Grossman, A.D., van Oudenaarden, A., 2002. Regulation of noise in the expression of a single gene. *Nat. Gen.* 31, 69–73.
- Paulsson, J., 2004. Summing up the noise in gene networks. *Nature* 427, 415–418.
- Plahte, E., Mestl, T., Omholt, S.W., 1998. A methodological basis for description and analysis of systems with complex switch-like interactions. *J. Math. Biol.* 36, 321–348.
- Ptashne, M., 2004. *A Genetic Switch*, 3rd ed. Cold Spring Harbor Laboratory Press, New York.
- Ramachandra, M., et al., 2001. Re-engineering adenovirus regulatory pathways to enhance oncolytic specificity and efficacy. *Nat. Biotechnol.* 19, 1035–1041.
- Russell, S.J., Norvig, P., 2002. *Artificial Intelligence: A Modern Approach*, 2nd ed. Prentice Hall, New Jersey.
- Shmulevich, I., Dougherty, E.R., Zhang, W., 2002a. From Boolean to probabilistic Boolean networks as models of genetic regulatory networks. *Proc. IEEE* 90 (11), 1778–1792.
- Shmulevich, I., Dougherty, E.R., Kim, S., Zhang, W., 2002b. Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics* 18, 261–274.
- Shmulevich, I., Dougherty, E.R., Zhang, W., 2002c. Control of stationary behavior in probabilistic Boolean networks by means of structural intervention. *J. Biol. Syst.* 10 (4), 431–445.
- Smith, H., 1987. Oscillation and multiple steady states in a cyclic gene model with repression. *J. Math. Biol.* 25, 169–190.
- Smolen, P., Douglas, A.B., Byrne, J.H., 2000. Modeling transcriptional control in gene networks: methods, recent results, and future directions. *Bull. Math. Biol.* 62, 247–292.
- Thattai, M., van Oudenaarden, A., 2001. Intrinsic noise in gene regulatory networks. *Proc. Natl. Acad. Sci.* 98, 8614–8619.
- Thieffry, D., Thomas, R., 1995. Dynamical behavior of biological regulatory networks. II. Immunity control in bacteriophage lambda. *Bull. Math. Biol.* 57 (2), 277–297.
- Thomas, R., Thieffry, D., Kaufman, M., 1995. Dynamical behavior of biological regulatory networks. I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state. *Bull. Math. Biol.* 57 (2), 247–276.
- Thomas, R., Kaufman, M., 2001. Multistationarity, the basis of cell differentiation and memory. II. Logical analysis of regulatory networks in terms of feedback circuits. *Chaos* 11 (1), 180–195.