MODELING AND CONTROL OF GENETIC REGULATORY NETWORKS

A Dissertation

by

RANADIP PAL

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2007

Major Subject: Electrical Engineering

MODELING AND CONTROL OF GENETIC REGULATORY NETWORKS

A Dissertation

by

RANADIP PAL

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Aniruddha Datta
Committee Members,	Edward R. Dougherty
	Shankar Bhattacharyya
	Thomas Vogel
Head of Department,	Costas N. Georghiades

August 2007

Major Subject: Electrical Engineering

ABSTRACT

Modeling and Control of

Genetic Regulatory Networks. (August 2007)

Ranadip Pal, B.Tech., Indian Institute of Technology, Kharagpur;

M.S., Texas A&M University

Chair of Advisory Committee: Dr. Aniruddha Datta

In recent years, there has been considerable interest in the area of Genomic Signal Processing, which is the engineering discipline that studies the processing of genomic signals. Signal processing approaches, such as detection, prediction and classification, have been used in the recent past to construct genetic regulatory networks capable of modeling genetic behavior. One of the objectives of network modeling is to use the network to design different intervention approaches for affecting the time evolution of the gene activity profile of the network. More specifically, one is interested in intervening to help the network avoid undesirable states such as those associated with a disease.

This dissertation considers the inference of genetic regulatory networks in the context of Boolean and Probabilistic Boolean Networks along with the subsequent optimal control of these networks. Algorithms to infer Boolean Networks with prescribed attractor structure and Probabilistic Boolean Networks matching the steady state data are developed. Based on the time duration of application of the control policy, two forms of optimal control strategies are designed: (i) Finite horizon control to desirably affect the dynamic evolution of the network over a finite number of time steps and (ii) Infinite horizon control to alter the steady-state distribution of the network. The dissertation also examines the robustness of the intervention strategies to uncertainties in the state transition probabilities of the network.

The network generation algorithms presented in this dissertation can be used to gener-

ate synthetic networks to test proposed inference algorithms, for both Boolean and probabilistic Boolean networks. This dissertation extends earlier results on intervention in instantaneously random PBNs without perturbation to context-sensitive PBNs with perturbation. The results show that the expected cost with control is much lower than without control. Furthermore, we showed that the stationary policies obtained using infinite horizon formulation can be used to shift the steady state distribution from undesirable to desirable states. Finally, through analytical derivation and simulation studies, we demonstrated that the stationary infinite horizon optimal control policies proposed in this dissertation are quite robust with respect to network uncertainty.

ACKNOWLEDGMENTS

Graduate study at Texas A&M has been a great experience for me culminating in this dissertation. I would like to acknowledge a number of people whose help made all this possible. I especially thank my advisor, Dr. Aniruddha Datta, for his generous time and guidance throughout my research. He encouraged me to develop independent thinking and greatly assisted me with scientific writing. I owe my gratitude to Dr. Edward Dougherty for his invaluable suggestions and guidance throughout my research. I am also grateful to my other doctoral committee members, Dr. Shankar Bhattacharyya and Dr. Thomas Vogel, for their encouragement and support.

I extend many thanks to my colleagues and collaborators, especially Ashish Choudhary, Ivan Ivanov and Michael Bittner for the conversations which assisted my research. I am also thankful to my friends at Texas A&M for the highly required diversions and support.

Finally, I am grateful to my parents and brother for their unconditional support, love and inspiration throughout my life.

The research was funded by National Science Foundation under Grants ECS-0355227 and CCF-0514644. In addition, I would like to thank the Department of Electrical and Computer Engineering at Texas A&M for awarding me the Ebensberger/Fouraker Fellowship.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

TABLE Page

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

The sequencing of various genomes over the last decade has given a remarkable boost to genomic studies. The improved understanding of the genomes of various organisms, along with advances in microarray technology, have provided us with enormous opportunities for the mathematical modeling of biological networks. There are two major objectives for modeling of genetic regulatory networks: (i) first, to better understand the intergene interactions and relationships on a holistic level, thereby facilitating the diagnosis of disease; and (ii) second, to design and analyze therapeutic intervention strategies for shifting the state of a diseased network from an undesirable location to a desirable one. The first objective falls within the scope of the field known as *Systems Biology* while the second objective falls within the scope of the field known as *Systems Medicine*. Systems medicine approaches that make use of genome based systems engineering fall within the scope of the field known as *Translational Genomics*. The dissertation mainly focuses on problems that arise in Translational Genomics.

In order to set the stage for introducing the problems, we next present a broad overview of the steps involved in the modeling and control of genetic networks. These steps are shown in Fig. 1. The first step consists of data extraction, which basically involves signal acquisition, the signals in this case being the expression levels of various genes of interest. The next step denoted by A_1 involves the discretization of these gene expression levels. Obviously, this step is not required if we are interested in arriving at a analog or continuous state model. On the other hand, this step would be crucial for discrete models such

---

The journal model is *IEEE Transactions on Automatic Control.*

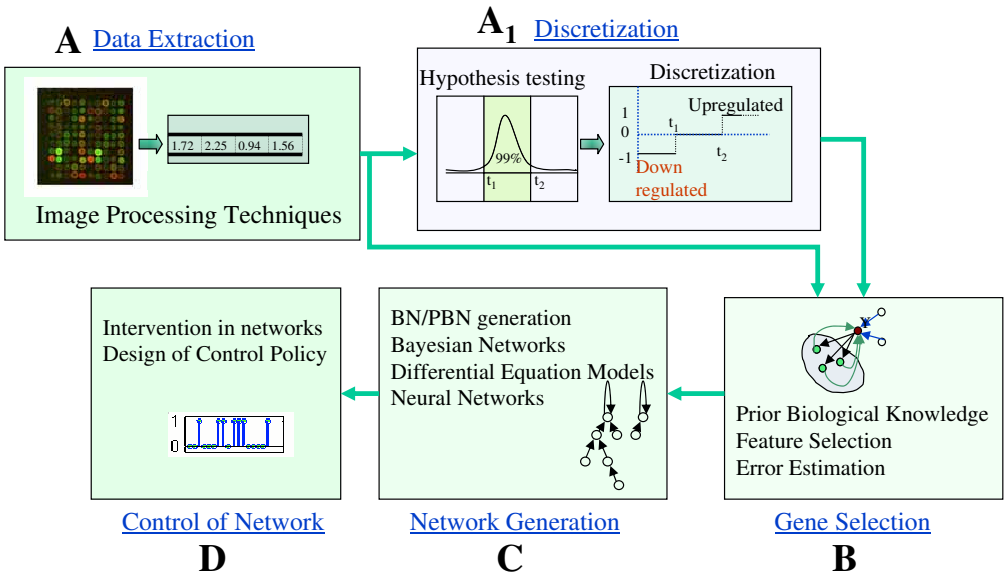


Fig. 1. Basic Steps Involved in Modeling and Control of Genetic Networks. (A) Extraction of gene expression data (A_1) Discretization of the Data (B) Selection of genes to build the network (C) Generation of network from the available data and prior biological knowledge (D) Intervention in the network with the objective of moving the network from undesirable to desirable states.

as Boolean Networks (BNs) [1, 2, 3], Probabilistic Boolean Networks (PBNs) [4, 5] and Bayesian Networks [6], all of which have been proposed as models for genetic regulatory networks. The next step denoted by B involves the selection of a small set of genes to be used in constructing the genetic regulatory network. This step is necessary because of at least two reasons: (i) first, building a network of thousands of genes would require an inordinately large amount of data for inference purposes, not to mention the computational intractability of the resulting network; and (ii) second, while modeling a particular biological pathway only a few genes may be playing an important role. Having selected the genes, the next step is the actual construction of the network, and a number of approaches can be used to carry out such construction [7, 8, 9].

Since systems biology is focussed on understanding the detailed molecular interactions that contribute to the functioning of a cell, a genetic regulatory network designed for facilitating such an understanding must necessarily mimic the actual biological interactions in as much detail as possible. On the other hand, in translational genomics the focus is on developing therapeutic interventions, and the network used for this purpose can be a coarse representation of the biological phenomena occurring at the molecular level as long as it has the capability to faithfully capture the overall effects of intervention that are manifested at the phenotypic (observational) level. Such a coarse model can then be used to develop and evaluate suitable (control) strategies for therapeutic intervention. Probabilistic Boolean Networks (PBNs), which constitute one class of coarse models, will be used in this dissertation as the network model of choice.

The focus of this dissertation is on parts C and D of Figure 1. We address four important issues arising in translational genomics.

Dynamical modeling of gene regulation via network models constitutes a key problem for genomics. The long-run characteristics of a dynamical system are critical and their determination is a primary aspect of system analysis. In the other direction, system synthesis

involves constructing a network possessing a given set of properties. This constitutes the inverse problem. Generally, the inverse problem is ill-posed, meaning there will be many networks, or perhaps none, possessing the desired properties. Relative to long-run behavior, we may wish to construct networks possessing a desirable steady-state distribution. One of the goals of this dissertation is to address the long-run inverse problem pertaining to Boolean networks (BNs) and Probabilistic Boolean Networks (PBNs). The long-run behavior of a BN is characterized by its attractors. The rest of the state transition diagram is partitioned into level sets, the j-th level set being composed of all states that transition to one of the attractor states in exactly j transitions. We present two algorithms for the attractor inverse problem. The attractors are specified, and the sizes of the predictor sets and the number of levels are constrained. Algorithm complexity and performance are analyzed. The algorithmic solutions have immediate application. Under the assumption that sampling is from the steady state, a basic criterion for checking the validity of a designed network is that there should be concordance between the attractor states of the model and the data states. This criterion can be used to test a design algorithm: randomly select a set of states to be used as data states; generate a BN possessing the selected states as attractors, perhaps with some added requirements such as constraints on the number of predictors and the level structure; apply the design algorithm; and check the concordance between the attractor states of the designed network and the data states.

From a translational perspective, the ultimate objective of genetic regulatory network modeling is to use the network to design different approaches for affecting network dynamics in such a way as to avoid undesirable phenotypes, for instance, cancer. To date, intervention studies using PBNs have used three different approaches: (i) resetting the state of the PBN, as necessary, to a more desirable initial state and letting the network evolve from there [10]; (ii) changing the steady-state (long-run) behavior of the network by minimally altering its rule-based structure [11]; and (iii) manipulating external (control) variables that

alter the transition probabilities of the network and can, therefore, be used to desirably affect its dynamic evolution [12]. In this dissertation, we extend the control-theoretic approach in two important directions. First, whereas the original control-theoretic approach has been developed in the framework of *instantaneously random* PBNs, here we design optimal intervention for *context-sensitive* PBNs [13]. This extension is significant because the latter class more closely models small biological subnetworks whose logical behavior is affected by conditions external to the network. Second, the earlier finite horizon results are extended to the infinite horizon case in an effort to alter the steady-state behaviour of the genetic regulatory network. Moreover, the stationary policies obtained in case of infinite horizon control are much easier to implement than a policy that changes with time.

Finally, we study the robustness of the infinite horizon intervention and examine how uncertainties in the transition probability matrix of the uncontrolled PBN show up in the steady-state distribution of the controlled PBN. Since the steady-state distribution of a PBN is thought to characterize the phenotype, our studies essentially seek to examine the effect of network uncertainty on the phenotype that would result from the application of intervention strategies. Through analytical derivation and simulation studies, we demonstrate that the stationary infinite horizon optimal control policies proposed are quite robust with respect to network uncertainty.

The dissertation is organized as follows. Chapter II provides a review of Genetic Regulatory Networks. In Chapter III, we design algorithms for generating Boolean Networks and Probabilistic Boolean Networks from Steady State Data. The first part of Chapter IV deals with the design of optimal intervention strategy for context-sensitive PBNs to desirably affect the dynamic evolution of the network over a finite number of time steps. The second part of the chapter formulates and solves the optimal infinite-horizon control problem for PBNs to alter the stationary distribution of the network. Chapter V examines the robustness of the stationary policies to uncertainties in the state transition probabilities of

the PBNs. Finally, Chapter VI contains some concluding remarks. For clarity of presentation, some of the technical details are relegated to the appendices.

CHAPTER II

MODELING OF GENETIC REGULATORY NETWORKS

A central focus of genomic research concerns understanding the manner in which cells execute and control the enormous number of operations required for normal function and the ways in which cellular systems fail in disease. In biological systems, decisions are reached by methods that are exceedingly parallel and extraordinarily integrated, as even a cursory examination of the wealth of controls associated with the intermediary metabolism network demonstrates. Feedback and damping are routine even for the most common of activities, cell cycling, where it seems that most proliferative signals are also apoptosis priming signals as well, and the final response to the signal results from successful negotiation of a large number of checkpoints, which themselves involve further extensive cross checks of cellular conditions.

Traditional biochemical and genetic characterizations of genes do not facilitate rapid sifting of these possibilities to identify the genes involved in different processes or the control mechanisms employed. Of course, when methods do exist to focus genetic and biochemical characterization procedures on a smaller number of genes likely to be involved in a process, progress in finding the relevant interactions and controls can be substantial. The earliest understandings of the mechanics of cellular gene control were derived in large measure from studies of just such a case, metabolism in simple cells. In metabolism, it is possible to use biochemistry to identify stepwise modifications of the metabolic intermediates and genetic complementation tests to identify the genes responsible for catalysis of these steps, and those genes and cis-regulator[1] elements involved in control of their

[1] A *cis-regulator* is a DNA sequence that controls the transcription of a related gene.

expression. Standard methods of characterization guided by some knowledge of the connections could thus be used to identify process components and controls. Starting from the basic outline of the process, molecular biologists and biochemists have been able to build up a very detailed view of the processes and regulatory interactions operating within the metabolic domain.

In contrast, for most cellular processes, general methods to implicate likely participants and to suggest control relationships have not emerged from classical (often correlation-based) approaches. The resulting inability to produce overall schemata for most cellular processes has meant that gene function has been, for the most part, determined in a piecemeal fashion. Once a gene is suspected of involvement in a particular process, research focuses on the role of that gene in a very narrow context. This typically results in the full breadth of important roles for well-known, highly characterized genes being slowly discovered. A particularly good example of this is the relatively recent appreciation that oncogenes such as Myc can stimulate apoptosis in addition to proliferation. Because transcriptional control is accomplished by a complex method that interprets a variety of inputs, the development of analytical tools that detect multivariate influences on decision-making present in complex genetic networks is essential. Modeling and analysis of gene regulation can substantially help to unravel the mechanisms underlying gene regulation and to understand gene function [14, 15, 16]. This, in turn, can have a profound effect on developing techniques for drug testing and therapeutic intervention for effective treatment of disease [17].

A. Nonlinear Dynamical Modeling of Gene Networks

Two salient aspects of a genetic regulatory system must be modeled and analyzed. One is the topology (connectivity structure) and the other is the set of interactions between the

elements, the latter determining the dynamical behavior of the system. Exploration of the relationship between topology and dynamics may lead to valuable conclusions about the structure, behavior, and properties of genetic regulatory systems [18, 19].

Numerous mathematical and computational methods have been proposed for construction of formal models of genetic interactions. Generally, these models share certain characteristics: (1) they represent *systems* in that they characterize an interacting group of components forming a whole, can be viewed as a process that results in a transformation of signals, and generate outputs in response to input stimuli; (2) they are *dynamical* in that they capture the time-varying quality of the physical process under study and can change their own behavior over time; and (3) they can be considered to be generally *nonlinear*, in that the interactions within the system yield behavior that is more complicated than the sum of the behaviors of the agents.

The preceding characteristics are representative of nonlinear dynamical systems. These are composed of states, input and output signals, transition operators between states, and output operators. In their abstract form, they are very general. More mathematical structure is provided for particular application settings. For instance, in computer science they can be structured into the form of dataflow graphical networks that model asynchronous distributed computation, a model that is very close to genomic regulatory models. Indeed, most attempts to model gene regulatory networks fall within the scope of nonlinear dynamical systems, including probabilistic graphical models, such as Bayesian networks [20, 6, 21]; neural networks [22, 23]; and differential equations [24]; see [25] for a review. Based on long experience in electrical and computer engineering, and more recent evidence from genomics itself, nonlinear dynamical systems appear to provide the appropriate framework to support the modeling of genomic systems. To build a model for a specific application requires abstracting from the specifics of the problem, and the breadth of nonlinear dynamical systems facilitates modeling within their framework.

Many concepts relevant to genomic regulation have been characterized from the perspectives of mathematical theory, estimation of model parameters, and application paradigms. We mention a few. *Structural stability* concerns the persistent behavior of a system under perturbation. It captures the idea of behavior that is not destroyed by small changes to the system. This is certainly a property of real genetic networks, since the cell must be able to maintain homeostasis[2] in the face of external perturbations and stimuli. *Uncertainty* relative to model behavior and knowledge acquisition has been extensively explored. Information theory, traditionally used for communications technology applications, is well suited to study uncertainty measures, quantified through the use of probability theory. *Distributed control* is common for complex systems, which have the property that no single agent is singularly in control of the system behavior; rather, control is dispersed among all agents, with varying levels of influence. This is the current view of genetic regulatory networks. To significantly change the global behavior of a system in a desired manner via external control, it is necessary to consider the effects holistically. This property is consistent with the inherent global stability of genetic networks in the presence of small changes to the system. This issue is addressed within control theory, where a central problem is *controllability*: how to select inputs so that the state of the system takes a desired value after some period of time. This is precisely the kind of issue that must be addressed for treatment of cancer and other genetically related diseases. In sum, nonlinear dynamical systems provide a framework for modeling and studying gene regulatory networks.

A key question concerns which model one should use. Model selection depends on the kind and amount of data available and the goals of the modeling and analysis. This choice involves classical engineering trade-offs. Should a model be *fine*, with many parameters to capture detailed low-level phenomena, such as protein concentrations and kinetics of

[2]Homeostatis is the ability of living systems to maintain internal equilibrium by adapting their physiology.

reactions, but thereby requiring a great deal of data for inference; or should it be *coarse*, with fewer parameters and lower complexity, thus being limited to capturing high-level phenomena, such as whether a gene is ON or OFF at a given time, but thereby having the advantage of requiring much smaller amounts of data [26]. Ultimately, model selection needs to obey the principle of Occam's razor; model complexity should be sufficient to faithfully explain the data but not be greater. From a pragmatic engineering perspective, this is interpreted to mean that the model should be as simple as possible to sufficiently solve the problem at hand. In the context of a functional network, complexity is determined by the number of nodes, the connectivity between the nodes, the complexity of the functional relations, and the quantization.

B. Boolean Networks (BNs)

This section focuses on the original deterministic version of the Boolean model. The more recently proposed stochastic extension will be presented in Section C. The Boolean model is archetypical of logical functional models and many of the issues that arise with it arise in other regulatory network models. A key issue in this dissertation is intervention in gene regulatory networks and this has mainly been considered in the context of a probabilistic generalization of the Boolean model.

1. Boolean Model

The regulatory model that has perhaps received the most attention is the Boolean network model [1, 2, 27]. The model has been studied both in biology and physics. In the Boolean model, gene expression is quantized to two levels: ON and OFF. The expression level (state) of each gene is functionally related to the expression states of other genes using logical rules. Although binarization provides very coarse quantization, we note that it is

commonplace to describe genetic behavior in binary logical language, such as *on* and *off*, *up-regulated* and *down-regulated*, and *responsive* and *non-responsive*. In the context of expression microarrays, consideration of differential expression leads to the categories of *low-expressed* and *high-expressed*, thereby leading to binary networks, or to the categories of *low-expressed*, *high-expressed*, and *invariant*, thereby leading to ternary valued networks that are treated in much the same way as binary networks and often referred to as Boolean networks.

Successful application of the Boolean model requires the inclusion of genes whose behavior is essentially binary (bi-modal). It has been demonstrated in the context of microarrays that there can be sufficiently many switch-like genes so that binary quantization can be successfully utilized for clustering [28] and classification [29]. From the perspective of logical prediction, numerous Boolean relations have been observed in the NCI 60 Anti-Cancer Drug Screen cell lines [30]. Some examples are

$$MRC1 = VSNL1 \vee HTR2C$$
$$SCYA7 = CASR \wedge MU5SAC \tag{2.1}$$

Moreover, using classical methods there is ample evidence demonstrating inherent logical genomic decision making [31, 32]. Figure 2 shows a biologically studied regulatory pathway and its corresponding Boolean representation. A full description of the biological model is given in [5]; here we restrict ourselves to noting that for cells to move into the S phase, cdk2 and cyclin E work together to phosphorylate the Rb protein and inactivate it, thereby releasing cells into the S phase, and that misregulation can result in unregulated cell growth.

A *Boolean network* is defined by a set of nodes, $V = \{x_1, x_2, \ldots, x_n\}$ and a list of

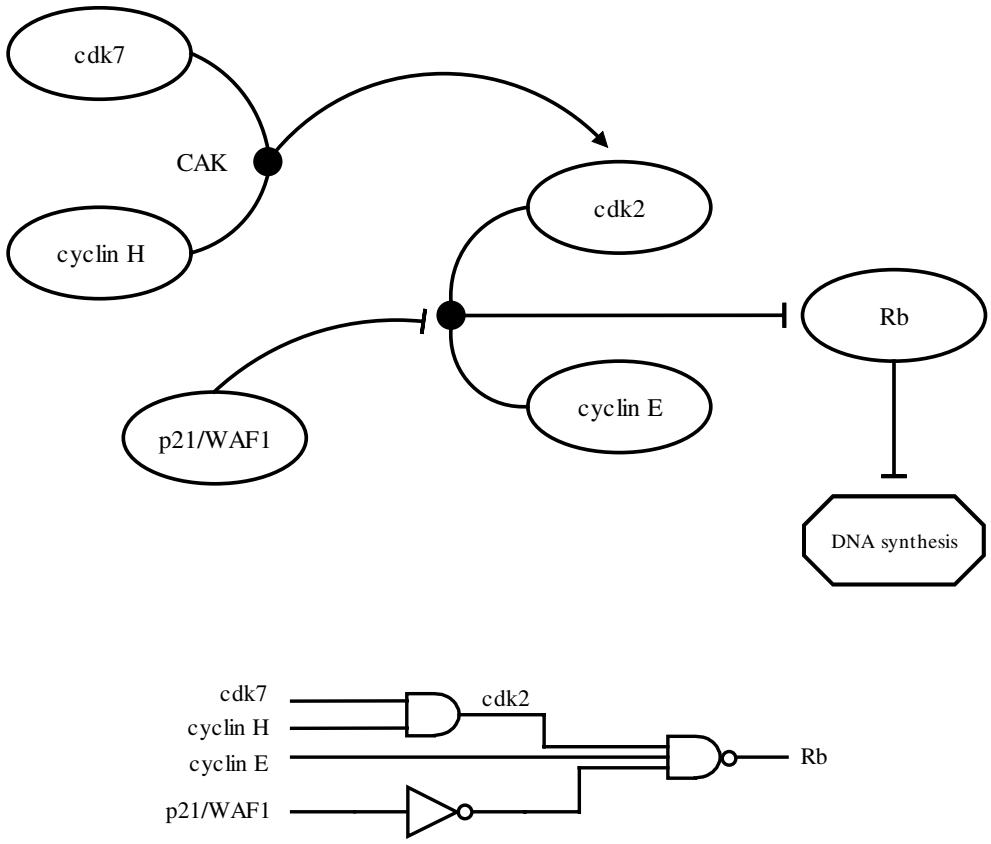


Fig. 2. Regulation of the Rb Protein in the Cell Cycle:(a) Biological Model; (b) Boolean Representation.

Boolean functions, $F = \{f_1, f_2, \ldots, f_n\}$. Each x_k represents the state (expression) of a gene, g_k, where x_k = 1 or x_k = 0, depending on whether the gene is expressed or not expressed. The Boolean functions represent the rules of regulatory interaction between the genes. Network dynamics result from a synchronous clock with times t = 0, 1, 2,.... The value of gene g_k at time t + 1 is determined by

$$x_k(t+1) = f_k(x_{k1}, x_{k2}, \ldots, x_{k,m(k)}) \tag{2.2}$$

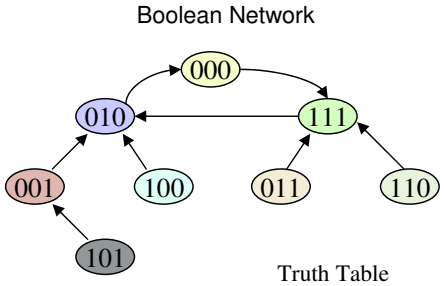
where the nodes in the argument of f_k form the *regulatory set* for x_k (gene g_k). The numbers of genes in the regulatory sets define the *connectivity* of the network, with maximum connectivity often limited. At time point t, the state vector

$$x(t) = (x_1(t), x_2(t), \ldots, x_n(t)) \tag{2.3}$$

is called the *gene activity profile* (*GAP*). The functions together with the regulatory sets determine the network wiring. An example BN of 3 genes is shown in Figure. 3 along with the Truth Table. The states are shown as binary numbers.

A Boolean network is a very coarse model; nonetheless, it facilitates understanding of the generic properties of global network dynamics [3, 33], and its simplicity mitigates data requirements for inference.

Attractors play a key role in Boolean networks. Given a starting state, within a finite number of steps, the network will transition into a cycle of states, called an *attractor*, and absent perturbation will continue to cycle thereafter. Each attractor is a subset of a *basin* composed of those states that lead to the attractor if chosen as starting states. The basins form a partition of the state space for the network. Non-attractor states are transient. They are visited at most once on any network trajectory. Figure 4 provides a transition-flow schematic for a Boolean network containing six genes, with states 0 = 000000, 1 =



Truth Table

x_1	x_2	x_3	f_1	f_2	f_3
0	0	0	1	1	1
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	1	1
1	0	0	0	1	0
1	0	1	0	0	1
1	1	0	1	1	1
1	1	1	0	1	0

Fig. 3. Boolean Network and Corresponding Truth Table.

000001,. . . , 63 = 111111. There are three singleton attractors, 32, 41, and 55. There are four transient levels, where a state in level k transitions to an attractor in k time points.

The attractors of a Boolean network characterize the long-run behavior of the network and have been conjectured by Kauffman to be indicative of the cell type and phenotypic behavior of the cell [3]. Real biological systems are typically assumed to have short attractor cycles, with singleton attractors being of special importance. For instance, it has been suggested that apoptosis and cell differentiation correspond to some singleton attractors and their basins, while cell proliferation corresponds to a cyclic attractor along with its associated basin [33]. Changes in the Boolean functions, via mutations or rearrangements, can lead to a rewiring in which attractors appear that are associated with tumorigenesis.

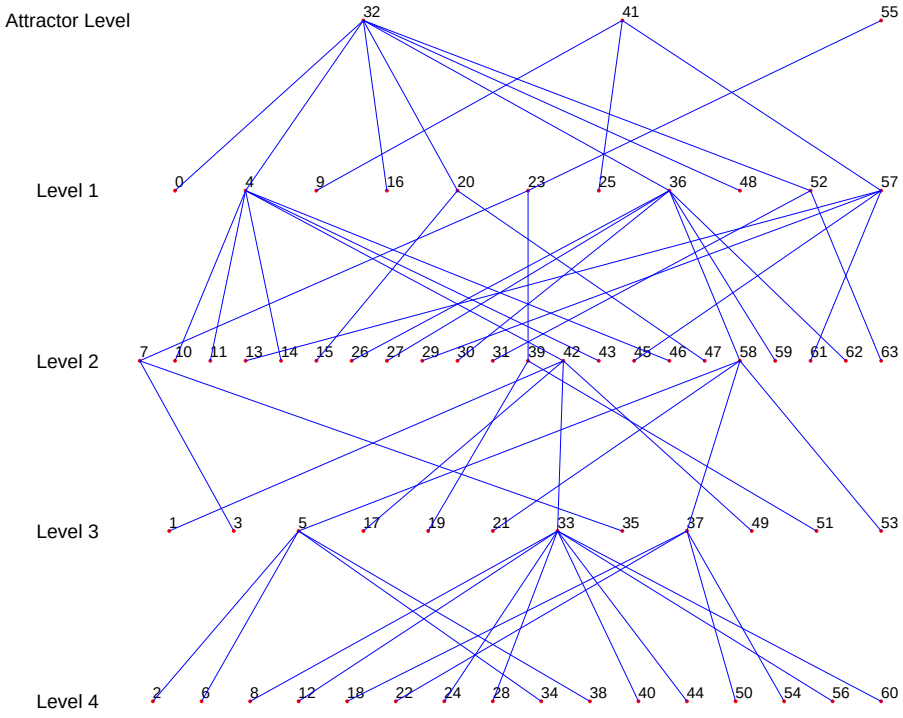


Fig. 4. A Boolean Network with Three Singleton Attractors and Four Transient Levels.

This is likely to lead to a cancerous phenotype unless the corresponding basins are shrunk via new-rewiring, so that the cellular state is not driven to a tumorigenic phenotype, or, if already in a tumorigenic attractor, the cell is forced to a different state by flipping one or more genes. The objective of cancer therapy would be to use drugs to do one or both of the above.

2. BN Representation

The binary n-digit state vector $x(t)$ can be mapped to positive integers $z(t)$ so that as $x(t)$ ranges from $00\cdots0$ to $11\cdots1$, $z(t)$ goes from 0 to 2^n-1. Here we employ the decimal representation $z(t)$ and the set $S=\{0,1,2,\cdots,2^n-1\}$ constitutes the state space for the Boolean network. Furthermore, each $z(t)$ can be uniquely represented by a basis

vector $w(t) \in R^{2^n}$, where $w(t) = e_{z(t)}$, e.g. if $z(t) = 0$, then $w(t) = [1, 0, 0, \ldots]$. Then, as discussed in [5], the evolution of the vector $w(t)$ proceeds according to the difference equation

$$w(t+1) = w(t)A \tag{2.4}$$

where A is a $2^n \times 2^n$ matrix having only one non-zero entry (equal to one) in each row.

3. Coefficient of Determination

By viewing gene status across different conditions, say, via microarrays, it is possible to establish relationships between genes that show variable status across the conditions. Owing to limited replications, we assume that gene expression data are quantized based on some statistical analysis of the raw data. One way to establish multivariate relationships among genes is to quantify how the estimate for the expression status of a particular *target gene* can be improved by knowledge of the status of some other *predictor genes*. This is formalized via the *coefficient of determination* (*CoD*) [34], which is defined by

$$CoD = \frac{\varepsilon_0 - \varepsilon_{opt}}{\varepsilon_0} \tag{2.5}$$

where ε_0 is the error of the best numerical predictor of the target gene in the absence of observation and ε_{opt} is the error of the optimal predictor of the target gene based on the predictor genes. This nonlinear form of the CoD is essentially a nonlinear, multivariate generalization of the familiar goodness of fit measure in linear regression. The CoD measures the degree to which the best estimate for the transcriptional activity of a target gene can be improved using the knowledge of the transcriptional activity of some predictor genes, relative to the best estimate in the absence of any knowledge of the transcriptional activity of the predictors. The CoD is a number between 0 and 1, a higher value indicating a tighter relationship.

Figure 5 shows a CoD diagram for the target gene p53 and predictor genes p21 and MDM2, in which the CoDs have been estimated in the context of a study involving stress response [35]. We see that the individual CoDs for p21 and MDM2 are 0.227 and 0.259, respectively, but when used jointly, the CoD for the predictor set $\{$p21, MDM2$\}$ increases to 0.452. Biologically, it is known that p53 is influential but not determinative of the up regulation of both p21 and MDM2, and hence it is not surprising that some level of prediction of p53 should be possible by a combination of these two genes. Note that the prediction of p53 by p21 and MDM2 apparently results from the regulation of p53 on them, not the other way around. Going the other way, the same study found the CoD for p53 predicting p21 to be 0.473. The increased predictability of p53 using both MDM2 and p21 is expected because increasing the size of the predictor set cannot result in a decrease in CoD. The extent of the increase can be revealing. In Fig. 5, MDM2 and p21 have very similar CoDs relative to p53 and there is a significant increase when they are used in combination. On the other hand, it may be that very little, if any, predictability is gained by using predictors in combination. Moreover, it may be that the individual predictors have CoDs very close (or equal) to 0, but when used in combination the joint CoD is 1. This kind of situation shows that it is risky to assume that a predictor g_1 and target g_0 are unrelated because the CoD of g_1 predicting g_0 is very low. This situation is akin to that in classification where a feature may be poor if used alone but may be good if used in combination with other features. The issue in both settings is the danger of *marginal analysis* – drawing conclusions about variables from marginal relations instead of joint (multivariate) relations. The complex nonlinear distributed regulation ubiquitous in biological systems makes marginal analysis highly risky.

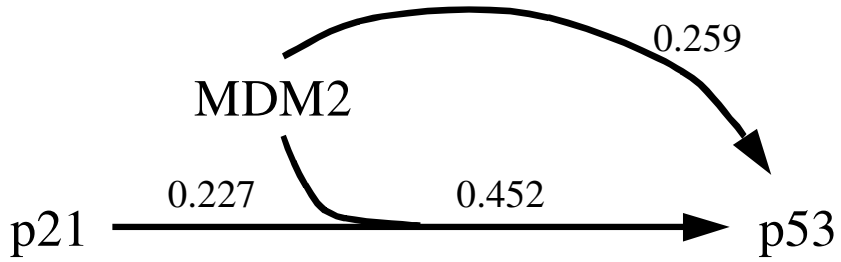


Fig. 5. CoD Diagram for p21 and MDM2 Predicting p53.

C. Probabilistic Boolean Networks (PBNs)

Given a target gene, several predictor sets may provide equally good estimates of its transcriptional activity, as measured by the CoD. Moreover, one may rank several predictor sets via their CoDs. Such a ranking provides a quantitative measure to determine the relative ability of each predictor set to improve the estimate of the transcriptional activity of the particular target gene. While attempting to infer inter-gene relationships, it makes sense to not put all our faith in one predictor set; instead, for a particular target gene, a better approach is to consider a number of predictor sets with high CoDs. Considering each retained predictor set to be indicative of the transcriptional activity of the target gene with a probability proportional to its CoD represents feature selection for gene prediction.

Having inferred inter-gene relationships in some manner, this information can be used to model the evolution of the gene activity profile over time. It is unlikely that the determinism of the Boolean-network model will be concordant with the data. One could pick the

predictor set with the highest measure of predictability, but as remarked previously in the case of the CoD, there are usually a number of almost equally performing predictor sets, and for them we will have only estimates from the data. By associating several predictor sets with each target gene, it is not possible to obtain with certainty the transcriptional status of the target gene at the next time point; however, one can compute the probability that the target gene will be transcriptionally active at time t + 1 based on the gene activity profile at time t. The time evolution of the gene activity profile then defines a stochastic dynamical system. Since the gene activity profile at a particular time point depends only on the profile at the immediately preceding time point, the dynamical system is Markovian. Such systems can be studied in the established framework of Markov Chains and Markov Decision Processes. These ideas are mathematically formalized in *probabilistic Boolean networks* (*PBN*s) [5, 4]. In a PBN, the transcriptional activity of each gene at a given time point is a Boolean function of the transcriptional activity of the elements of its predictor sets at the previous time point. The choice of Boolean function and associated predictor set can vary randomly from one time point to another. For instance, when using the CoD, the choice of Boolean function and predictor set can depend on CoD-based selection probabilities associated with the different predictor sets. This kind of probabilistic generalization of a Boolean network, in which the Boolean function is randomly selected at each time point, defines an *instantaneously random PBN*.

Instead of simply assigning Boolean functions at each time point, one can take the perspective that the data come from distinct sources, each representing a *context* of the cell. From this viewpoint, the data derive from a family of deterministic networks and, were we able to separate the samples according to the contexts from which they have been derived, then there would in fact be CoDs with value 1, indicating deterministic biochemical activity for the wiring of a particular constituent network. Under this perspective, the only reason that it is not possible to find predictor sets with CoD equal (or very close to)

1 is because they represent averages across the various cellular contexts. This perspective results in the view that a PBN is a collection of Boolean networks in which one constituent network governs gene activity for a random period of time before another randomly chosen constituent network takes over, possibly in response to some random event, such as an external stimulus or genes not included in the model network. Since the latter is not part of the model, network switching is random. This model defines a *context-sensitive PBN*. The probabilistic nature of the constituent choice reflects the fact that the system is open, not closed, the idea being that network changes result from the genes responding to *latent* variables external to the model network. The context-sensitive model reduces to the instantaneously random model by having network switching at every time point.

Much of the theory and application of PBNs applies directly to the more general case which need not possess binary quantization and which are also called PBNs, owing to the multi-valued logical nature of functional relations for finite quantization. A particularly important case is ternary quantization, where expression levels take on the values +1 (up-regulated), -1 (down-regulated), and 0 (invariant).

A PBN is composed of a set of n genes, $x_1, x_2, \ldots, x_n$, each taking values in a finite set V (containing d values), and a set of vector-valued *network functions*, $\mathbf{f}_1, \mathbf{f}_2, \ldots \mathbf{f}_r$, governing the state transitions of the genes. To every node x_i, there corresponds a set

$$F_i = \{f_j^{(i)}\}_{j=1,\ldots,l(i)}, \tag{2.6}$$

where each $f_j^{(i)}$ is a possible function, called a *predictor*, determining the value of gene x_i and $l(i)$ is the number of possible functions assigned to gene x_i. Each network function is of the form $\mathbf{f}_k = (f_{k_1}^{(1)}, f_{k_2}^{(2)}, \ldots, f_{k_n}^{(n)})$, for $k = 1, \ldots, r$, $1 \leq k_i \leq l(i)$ and where $f_{k_i}^{(i)} \in F_i (i = 1, 2, \ldots, n)$. Each vector function $\mathbf{f}_k : \{0,1\}^n \rightarrow \{0,1\}^n$ acts as a transition function (mapping) representing a possible realization of the entire PBN. Thus, given the value of all genes, $(x_1, \ldots, x_n)$, $\mathbf{f}_k(x_1, x_2, \ldots, x_n)$=$(x_1', x_2', \ldots, x_n')$ gives us the state of the

genes after one step of the network given by the realization $\mathbf{f}_k$.

The choice of which network function $\mathbf{f}_j$ to apply is governed by a selection procedure. At each time point a random decision is made as to whether to switch the network function for the next transition, with a probability q of a change being a system parameter. If a decision is made to change the network function, then a new function is chosen from among $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_r$, with the probability of choosing $\mathbf{f}_k$ being the selection probability c_k.

Now, let $\mathbf{F}=(f^{(1)}, f^{(2)}, \ldots, f^{(n)})$ be a random vector taking values in $F_1 \times F_2 \ldots \times F_n$. That is, $\mathbf{F}$ can take on all possible realizations of the PBN. Then, the probability that predictor $f_j^{(i)}$ is used to predict gene i $(1 \leq j \leq l(i))$ is equal to

$$c_j^{(i)} = P\{f^{(i)} = f_j^{(i)}\} = \sum_{k: f_{k_i}^{(i)} = f_j^{(i)}} P\{\mathbf{F} = \mathbf{f}_k\}. \tag{2.7}$$

Since $c_j^{(i)}$ are probabilities , they must satisfy

$$\sum_{j=1}^{l(i)} c_j^{(i)} = 1. \tag{2.8}$$

It is not necessary that the selection of Boolean functions composing a specific network be independent. This means that it is not necessarily the case that

$$P\{f^{(i)} = f_j^{(i)}, f^{(l)} = f_k^{(l)}\} = P\{f^{(i)} = f_j^{(i)}\}.P\{f^{(l)} = f_k^{(l)}\}. \tag{2.9}$$

A PBN is said to be independent if the random variables $f^{(1)}, f^{(2)}, \ldots, f^{(n)}$ are independent. In the dependent case, product expansions such as the one given in Eq. 2.9, as well as ones involving more functions, require conditional probabilities. If the PBN is independent, then there are $L = \prod_{i=1}^{n} l(i)$ realizations (constituent Boolean networks). Moreover, for an independent PBN, if the k^{th} network is obtained by selecting $f_{i_r}^{(i)}$ for gene i , $i = 1, 2, \ldots, n$, $1 \leq i_r \leq l(i)$, then the selection probability c_k is given by $c_k = \prod_{i=1}^{n} c_{i_r}^{(i)}$.

A PBN with perturbation can be defined by there being a probability p of any gene

changing its value uniformly randomly to another value in V at any instant of time. Whereas a network switch corresponds to a change in a latent variable causing a structural change in the functions governing the network, for instance, in the case of a gene outside the network model that participates in the regulation of a gene in the model, a random perturbation corresponds to a transient value flip that leaves the network wiring unchanged, as in the case of activation or inactivation owing to external stimuli such as mutagens, heat stress, etc. [3].

The state space S of the network together with the set of network functions, in conjunction with transitions between the states and network functions, determine a Markov chain, the states of the Markov chain being of the form $(\mathbf{x}^i, \mathbf{f}_j)$. If there is random perturbation, then the Markov chain is ergodic, meaning that it has the possibility of reaching any state from another state and that its stationary distribution becomes a steady-state distribution. In the special case when $q = 1$, a network function is randomly chosen at each time point and the Markov chain consists only of the PBN states.

For a PBN, characterization of its long-run behavior is described via the Markov chain it defines. In particular, an instantaneously random PBN has equivalence classes of communicating states analogous to the basins of attraction for Boolean networks, and if there is perturbation, which we will always suppose, then the Markov chain is ergodic, which then guarantees the existence of a global steady-state distribution. In general, whether the PBN is instantaneously random or context-sensitive, by definition its attractors consist of the attractors of its constituent Boolean networks. Two events can remove a network from an attractor cycle C: (1) a perturbation can send it to a different state, and assuming the constituent network remains unchanged and there are no further perturbations for a sufficient time, then it will return to C if the perturbation leaves it in the basin of C or it will transition to a different attractor cycle of the same constituent network if the perturbation sends it to a different basin; (2) a network switch will put it in the basin of an attractor cycle for the

new constituent network and it will transition to the attractor cycle for that basin so long as the constituent network remains unchanged and there are no further perturbations for a sufficient time. Whereas the attractor cycles of a Boolean network are mutually disjoint, the attractor cycles of a PBN can intersect because different cycles can correspond to different constituent Boolean networks. Assuming that the switching and perturbation probabilities are very small, a PBN spends most of its time in its attractors. The probabilities of PBN attractors have been analytically characterized [36].

1. PBN Representation

In case of PBNs, we have a stochastic counterpart of Eq. 2.4 given by

$$w(t+1) = w(t)A \tag{2.10}$$

where $w(t)$ denotes the probability distribution vector at time t, i.e. $w_i(t\} = \Pr\{z(t) = i\}$ and A denotes the probability transition matrix.

D. Network Inference

For genetic regulatory networks to be of practical benefit, there must be methods to design them based on experimental data. We confront three impediments: (1) model complexity, (2) limited data, and (3) lack of appropriate time-course data to model dynamics. Numerous approaches to the *network inference problem* have been proposed in the literature, many based on gene-expression microarray data. Here, we briefly outline some of the proposed methods for PBNs and the rationale behind each of them (there also having been substantial study of inferring Boolean networks [4, 37]).

As first proposed, the inference of the PBN is carried out using the CoD [5]. For each gene in the network, a number of high CoD predictor sets are found and these high-

CoD predictor sets determine the evolution of the activity status of that particular gene. Furthermore, the selection probability of each predictor set for a target gene is assumed to be the ratio of the CoD of that predictor set to the sum of the CoDs of all predictor sets used for that target gene. This approach makes intuitive sense since it is reasonable to assign the selection probability of each predictor set in a PBN to be proportional to its predictive worth as quantified by the CoD.

A second approach to PBN construction uses mutual information clustering and reversible -jump Markov-chain-Monte-Carlo predictor design [38]. First, mutual- information-minimization clustering is used to determine the number of possible parent gene sets and the input sets of gene variables corresponding to each gene. Thereafter, each (predictor) function from the possible parent gene sets to each target gene is modeled by a simple neural network consisting of a linear term and a nonlinear term, and a reversible-jump Markov-chain-Monte-Carlo (MCMC) technique is used to calculate the model order and the parameters. Finally, the selection probability for each predictor set is calculated using the ratio of the CoDs.

In most expression studies, there is some degree of previous knowledge regarding genes that play a role in the phenotypes of interest, for instance, p53 in unregulated proliferation. To take advantage of this knowledge, and to obtain networks relating to genes of interest, it has been proposed to construct networks in the context of directed graphs by starting with a seed consisting of one or more genes believed to participate in a meaningful subnetwork [7]. Given the seed, a network is grown by iteratively adjoining new genes that are sufficiently interactive with genes in the growing network in a manner that enhances subnetwork autonomy. The proposed algorithm has been applied using both the CoD and the Boolean-function influence [5], which measures interaction between genes. The algorithm has the benefit of producing a collection of small tightly knit autonomous subnetworks as opposed to one massive network with a large number of genes. Such small

subnetworks are more amenable to modeling and simulation studies, and when properly seeded are more likely to capture a small set of genes that may be maintaining a specific core regulatory mechanism.

A key issue in network design arises because much of the currently available gene-expression data comes to us from *steady-state* phenotypic behavior and does not capture any temporal history. Consequently, the process of inferring a PBN, which is a dynamical system, from steady-state data is a severely *ill-posed inverse problem*. Steady-state behavior constrains the dynamical behavior of the network but does not determine it and, therefore, building a dynamical model from steady-state data is a kind of overfitting. It is for this reason that a designed network should be viewed as providing a regulatory structure that is consistent with the observed steady-state behavior. Also, it is possible that several networks may emerge as candidates for explaining the steady-state data. Under the assumption that we are sampling from the steady-state, a key criterion for checking the validity of a designed network is that much of its steady state mass lies in the states observed in the sample data because it is expected that the data states consist mostly of attractor states [39].

A number of recent papers have focused on network inference keeping in mind that most of the data states correspond to steady-state behavior. In one of these, a fully Bayesian approach has been proposed that emphasizes network topology [9]. The method computes the possible parent sets of each gene, the corresponding predictors and the associated probabilities based on a neural-network model, using a reversible jump MCMC technique; and an MCMC method is employed to search the network configurations to find those with the highest Bayesian scores to construct the PBNs. This method has been applied to a melanoma cell line data set. The steady-state distribution of the resulting model contains attractors that are either identical or very similar to the states observed in the data, and many of the attractors are singletons, which mimics the biological propensity to stably occupy a given state. Furthermore, the connectivity rules for the most optimally generated networks

constituting the PBN were found to be remarkably similar, as would be expected for a network operating on a distributed basis, with strong interactions between the components.

If we consider network inference from the general perspective of an ill-posed inverse problem, then one can formalize inference by postulating criteria that constitute a solution space in which a designed network must lie. For this we propose two kinds of criteria [40]:

- *Constraint criteria* are composed of restrictions on the form of the network, such as biological and complexity constraints.
- *Operational criteria* are composed of relations that must be satisfied between the model and the data.

Examples of constraint criteria include limits on connectivity and attractor cycles. One example of an operational criterion is some degree of concordance between sample and model CoDs, and another is the requirement that data states are attractor states in the model. The inverse problem may still be ill-posed with such criteria, but all solutions in the resulting space can be considered satisfactory relative to the requirements imposed by the criteria. We will implement this kind of approach in the next section by finding constituent Boolean networks satisfying constraints such as limited attractor structure, transient time, and connectivity [8].

In addition to the ongoing effort to infer PBNs, there has been a continuing effort to infer Bayesian and dynamic Bayesian networks (DBNs) [6, 41, 42]. A *Bayesian network* is essentially a compact graphical representation of a joint probability distribution [43, 44, 45]. This representation takes the form of a directed acyclic graph in which the nodes of the graph represent random variables and the directed edges, or lack thereof, represent conditional dependencies, or independencies. The network also includes conditional probability distributions for each of the random variables. In the case of genetic

networks, the values of the nodes can correspond to gene-expression levels or other measurable events, including external conditions. There is a precisely characterized relation between certain DBNs and PBNs in the sense that they can represent the same joint distribution over their corresponding variables [46]. PBNs are more specific in the sense that the mapping between PBNs and DBNs is many-to-one, so that a DBN does not specify a specific PBN.

CHAPTER III

GENERATING BOOLEAN NETWORKS WITH A PRESCRIBED ATTRACTOR STRUCTURE*

A. BN Generation

The dynamical modeling of gene regulation via network models constitutes a fundamental problem for genomics. In any dynamical system the long-run characteristics of the system are critical and determining these characteristics is a primary aspect of system analysis. In the other direction, and typically more difficult, is system synthesis: construct a network possessing a given set of properties. This constitutes the *inverse problem*. Generally, the inverse problem is ill-posed, meaning there will be many networks, or perhaps none, possessing the desired properties. Relative to long-run behavior, we may have a desirable stationary distribution and wish to construct networks possessing that stationary distribution. Here we are concerned about a long-run inverse problem pertaining to *Boolean networks*.

Boolean networks compose a class of discrete models where the expression levels of each gene are assumed to have two possible values only: up-regulated (ON) or down-regulated (OFF). Such a model cannot capture the underlying continuous and stochastic biochemical nature of protein production and gene regulation; however, one often encounters genes that are essentially ON or OFF throughout a given biochemical pathway. The switch-like regulatory function of those genes determines their role in regulation and their activity is well-represented by a coarse-grain model like a Boolean network. This, together with the relative simplicity of the dynamical system described by a Boolean network ex-

---

*Part of this chapter is reprinted, with permission, from ”Generating Boolean networks with a prescribed attractor structure”, R. Pal, I. Ivanov, A. Datta, M. L. Bittner, and E. R. Dougherty, *Bioinformatics*, vol. 21, pp. 40214025, 2005

plains why such networks have attracted significant attention from the research community – for instance, [1, 2, 3, 33, 4]. The dynamics of different classes of Boolean networks have been extensively studied from the ensemble point of view [3]. Statistical properties of the attractor structure, the connectivity, and the evolution of an ensemble of Boolean networks provide important insights about the genetic regulatory network modeled by those networks.

Given the relative simplicity of the model, the rich dynamical behavior that can be observed in different classes of Boolean networks, and the biologically sound interpretation of the attractor structure and the connectivity, a significant effort has been directed in designing such networks from real gene expression data. Much of this effort has concerned the strength of prediction among genes [4, 7] and the related issue of optimal connectivity [38, 9]. The inverse problem with respect to attractors is related to design from steady-state data, and therefore the algorithmic solution to that inverse problem proposed in this dissertation has immediate application.

As explained in Section 1, a Boolean network (BN) $B = (V, F)$ on n genes is defined by a set of nodes/genes $V = \{x_1, ..., x_n\}$, $x_i \in \{0, 1\}$, $i = 1, ..., n$, and a vector of Boolean predictor functions $F = (f_1, ..., f_n)$, $f_i : \{0, 1\}^n \rightarrow \{0, 1\}$, $i = 1, ..., n$. Each node x_i represents the state/expression of the gene x_i, where $x_i = 0$ means that the gene i is OFF and $x_i = 1$ means that the gene i is ON. The function f_i is the *predictor function* for that gene. Updating the states of all of the genes in B is done synchronously at every time step according to their predictor functions. A subset $W_i \subseteq V$ is called the *predictor set* for the gene x_i if the restriction $f_{i|W_i}$ of the predictor function f_i equals f_i. It is clear from this definition that the cardinality of the set W_i is related to the number of edges incident with the vertex x_i in the directed graph $\Gamma = (V, E)$, where an edge $(x_i, x_j) \in E$ indicates that gene x_i is one of the factors determining the value of the gene x_j. $W = (W_1, ..., W_n)$ is called the *predictor set* for the Boolean network. A *state* in B is a vector $(x_1, ..., x_n)$

of gene values. We shall always assume that the states of B are interpreted as binary numbers, and are ordered accordingly. Thus, there are $N = 2^n$ states in a Boolean network on n genes, and they are enumerated as $0, 1, 2, ..., N - 1$. There is a $N \times n$ truth table associated with and equivalent to B, where the rows correspond to the states in B and the columns contain the corresponding values for the predictor functions. The truth table of B induces a directed graph $\tilde{\Gamma} = (\tilde{V}, \tilde{E})$ with the states in B as the set $\tilde{V}$ of its vertices, and with edges $(s_i, s_j) \in \tilde{E}$ connecting the state s_i with the state s_j if $F(s_i) = s_j$. It is clear that the truth table associated with B determines $\tilde{\Gamma}$ and vice versa. $\tilde{\Gamma}$ is called the *state transition diagram* of B, and $\tilde{\Gamma}$ is called *compatible* with W if the truth table induced by $\tilde{\Gamma}$ has W as the predictor set for the Boolean network associated with that truth table. The state transition diagram represents the dynamics of the network.

The long-run behavior of a Boolean network is straightforward to describe. Given an initial state, the network will eventually enter a set of states through which it will repeatedly cycle forever. Each such set is called an *attractor cycle* and the states within the family of attractor cycles are called *attractor states*. The rest of the state transition diagram is partitioned into *level sets*, where the level set l_j is composed of all of the states of the network that transition to one of the attractor states in exactly j transitions. One can think of the set of attractor states as the level set l_0. Then non-attractor states compose the transient states of the network. The transient states are partitioned according to the attractor cycles because each transient state begins a sequence of transitions that eventually ends up in a unique attractor cycle. The attractor cycles are mutually disjoint. The class of the partition corresponding to an attractor cycle is called the *basin* of the cycle. Given any transient state, it belongs to a unique basin and unique level.

A state transition diagram constitutes a single-rooted tree if it possesses exactly one singleton attractor (a single-state attractor cycle): the network reaches its attractor cycle via the tree. If it possesses k singleton attractors, then it is composed of k single-rooted

trees and is called a k-*forest*: the network reaches an attractor state via one of the single-rooted trees. From the perspective of modeling gene regulation, the attractor cycles of a Boolean network are especially important because they are presumed to provide a representation/approximation of phenotypes. Singleton attractors are especially important. First, gene regulation should be modeled by Boolean networks in an ordered regime and a Boolean network that functions in an ordered regime has short attractor cycles, often singleton attractors [3]. Second, the presence of long cycles in the cell dynamics will lead to an entropy increase, which is exactly the opposite of the biological state stability and determinism that characterize living systems. In this chapter, we present two algorithms for the attractor inverse problem under the assumption of singleton attractor states. Complexity and performance of the two proposed algorithms are discussed.

Besides being of mathematical interest relative to understanding the nature of Boolean networks that lead to certain attractor structures, the attractor inverse problem is very important to network inference from state data, in particular, gene expression data. Most microarray-based gene-expression studies do not involve controlled temporal experimental data; rather, it is assumed that data result from sampling from the steady state. Under the Boolean-network model, this means that the data come from the attractors. If one considers a more general Boolean-type model, such as a Boolean network with random perturbations or a probabilistic Boolean network, the dynamical system represented by the network is an ergodic Markov chain and there exists a steady-state distribution; nevertheless, under mild stability assumptions that reflect biological state stability, most of the steady-state probability mass is concentrated in the attractors and it is expected that most data correspondingly come from the attractors [36].

Thus, under the assumption that we are sampling from the steady state, a key criterion for checking the validity of a designed network is that much of its steady-state mass lies in the states observed in the sample [39]. In the case of Boolean networks, this means that

there should be close concordance between the attractor states of the model and the data states. Such a criterion can be used to test a design algorithm [9]: randomly select a set of states to be used as data states; generate a Boolean network possessing the selected states as attractors, perhaps with some added requirements such as constraints on connectivity and the level structure; apply the design algorithm; and check the concordance between the attractor states of the designed network and the states used as data. This can be done repeatedly for different data states and constraints. The algorithms provided next can be used to generate the Boolean networks in this scenario. Owing to the concentration of mass in the attractors of probabilistic Boolean networks and the fact that a PBN can be viewed as a collection of Boolean networks, the procedure can be applied to PBNs by generating the constituent Boolean networks. The PBNs so synthesized can be used to design intervention strategies where the only available gene expression data for network design comes from the steady-state phenotypic behavior.

It is important to keep in mind that the inverse problem, attractors to network, is a one-to-many mapping, and there may be a multitude of networks possessing a given attractor structure. In the other direction, if the problem is constrained, say by the number of predictors permitted, there may be no solution. Generally speaking, steady-state behavior restricts the dynamical behavior, but does not determine it. In particular, for Boolean networks it does not determine the basin structure. Thus, while we might obtain good inference regarding the attractors, we may obtain poor inference relative to their probabilities relative to random initializations (or to random perturbations in more general networks). This is because if the basin of an attractor is small, it is less likely to catch a random initialization than if it is large. When sampling from the steady state, the data attractors with small basins are less likely to appear (and may not appear at all), whereas those with large basins may appear numerous times. A key advantage of checking a design algorithm with generated synthetic networks is that the levels and basins of a synthetic network are known

and therefore one can better evaluate algorithm performance.

In the following sections we present two algorithms for solving this inverse problem under the assumption of singleton attractor states in the designed network. As with any other algorithmic solution to a design problem, the important issues of complexity and performance of the proposed two algorithms are discussed. The problem we address can be formulated in the following manner. Given a set V consisting of n nodes (genes), a family of n subsets $W_1, W_2, ..., W_n$ of V with cardinalities not less than m and not bigger than M, $0 < m \leq M$, a set A containing k states, and two positive integers $l_d \leq l_u$, accordingly construct a Boolean network with node set V, having predictor set $W = (W_1, W_2, ..., W_n)$, possessing only singleton attractors, and these consisting precisely of the states in A, and containing between l_d and l_u level sets. The requirement on the predictor set means that the state transition diagram of the designed network must be compatible with W. There may exist none or many compatible networks. The algorithms are typically initiated by specifying a minimum and maximum number of predictors for each gene, randomly selecting $W_1, W_2, ..., W_n$ subject to the specified maximum and minimum, and randomly selecting k attractor states. In this way, one can utilize the algorithms to search for Boolean networks constrained by the connectivity of the network. For instance, if $|W_i| \leq M$ for $i = 1, 2, ..., n$, then the algorithms find networks with connectivity bounded by M – if any exist with the required attractor structure.

The problem can be reformulated as a search problem in the following way: In the space of all k-forests $\tilde{\Gamma}$ for a Boolean network on n genes, and with the number of their level sets ranging between l_d and l_u, find at least one which is compatible with a given (randomly generated) predictor set W, where the cardinality of each W_i ranges between m and M.

1. Size of the Search Space

Under the assumption that we are sampling from the steady state, biological state stability means that most of the steady-state probability mass is concentrated in the attractors and that real-world attractors are most likely to be singleton attractor cycles consisting of a single state. A state transition diagram constitutes a single-rooted tree if it possesses exactly one singleton attractor (a single-state attractor cycle): the network reaches its attractor via the tree. If it possesses k singleton attractors, then it is composed of k single-rooted trees and is called a *k-forest*: the network reaches an attractor state via one of the single-rooted trees. In this section we examine the size of the search state under the assumption of singleton attractors. To simplify the formulas, it is assumed that $m = 1$ and $l_d = 1$. One can easily make the necessary changes in the general case.

There are $A = A_{n,M} = \sum_{i=1}^{M} \binom{n}{i}$ possible predictor sets W_i for each gene x_i, $i = 1, ..., n$. Thus, there are A^n possible predictor sets W to select from when searching for a compatible state transition diagram $\tilde{\Gamma}$. The different choices for $\tilde{\Gamma}$ depend on the number of attractor states and on the level set structure of the state transition diagram. There are $\binom{N}{k}$ possible ways of selecting the k singleton attractors for $\tilde{\Gamma}$. The remaining $N_k = N - k$ non-attractor states will form the level sets of $\tilde{\Gamma}$. There are $n(l_{i+1})^{n(l_i)}$ ways for connecting any two successive level sets l_i and l_{i+1} with $n(l_i)$ and $n(l_{i+1})$ states in them, respectively. Therefore, the number of possible ways to structure the level sets for $\tilde{\Gamma}$ with no more than L level sets is

$$\Lambda = \Lambda_{L,k,n} = \sum_{i=1}^{L} \sum_{i} \frac{N_k!}{n(l_1)!...n(l_i)!} k^{n(l_1)} n(l_1)^{n(l_2)} ... n(l_{i-1})^{n(l_i)} \tag{3.1}$$

where $\sum_i$ is a summation over all of the different choices of the positive integers $n(l_1)$,...,

$n(l_i)$ such that $\sum_{j=1}^{i} n(l_j) = N_k$. Combining this with the choices for selecting attractor states, and with the choices for selecting predictor sets, yields the size of the search space,

$$S = S_{n,m,k} = A^n \binom{N}{k} \Lambda \tag{3.2}$$

To appreciate the size of the search space, consider an example of a very small BN, where $n = 4$, $m = 4$, $k = 4$, and the state transition diagram has exactly 4 levels. The computations show that $S \geq 10^{17}$.

The following theorem extends a well known result [47], about 1 -trees, or single rooted trees.

Theorem 1. The cardinality of the collection of all forests on N vertices is $(N+1)^{N-1}$.

Proof: In the proof we can assume without loss of generality that the vertices of each k-forest are labeled using the integers from $\{1, 2, ..., N\}$. First we prove that there is a bijection between the collection $\mathcal{F}_k$ of all k-forests, k-a fixed nonnegative integer less than N , and the collection $\mathcal{B}_k$ of triples (ω_k, A_k, r), $r \in A_k$, where the set $A_k \in \mathcal{A}_k$-the collection of all k element subsets of $\{1, ..., N\}$, and $\omega_k \in \Omega_{N_k}$-the collection of all sequences of length $N - k - 1$ integers formed using the integers $i \in \{1, ..., N\}$. Define the mapping

$$\lambda : \mathcal{F}_k \to \Omega_{N_k} \times \mathcal{A}_k \times \{1, ..., N\}$$

(A) In particular, given a k-forest F, the first component of $\lambda(F)$ is generated recursively in the following way:

Set $i = 1$

1. Search for the leaf in F with the smallest label v_i.
2. Remove the edge (v_i, v) from F.
3. Set the i-th element of ω_k to v i.e. set $\omega_{k,i} = v$.
4. Set $i = i + 1$.

5. If $i < N - k$ start from 1 again, otherwise set the third component of $\lambda(F) = r$ where (v_r, r) is the only remaining edge in F.

In the above procedure a root of a tree forming F is not considered to be a leaf after removal of the tree stemming from it. The second component of $\lambda(F)$ simply lists all of the roots of F. It is obvious that after repeating the above procedure $N - k - 1$ times we will end up with only the roots of $k - 1$ of the trees forming F plus the remaining leaf v_r connected to r. Notice that the linear order of the set $\{1, ..., N\}$ implies that the mapping λ is well defined.

(B) Next we start with a triple (ω_k, A_k, r), $r \in A_k$, and show that there is a k-forest F, such that $\lambda(F) = (\omega_k, A_k, r)$. Indeed, one can set the k roots of F to be the elements of A_k, and after that one can apply the following procedure generating the rest of F: Set $j = 1$. Form the set $B_k = \{1, ..., N\} \setminus A_k$. Create k roots for F using the elements in A_k.

(a) Find the smallest element $i \in B_k$ that is not equal to $\omega_{k,l}$, $l = j, j + 1, ..., N - k - 1$.

(b) Form an edge $(i, \omega_{k,j})$ in F.

(c) Set $B_k = B_k \setminus i$ and set $j = j + 1$

(d) If $j > N - k - 1$ connect the only element of B_k to r and then stop, otherwise start from (a) again.

Since at every step j we remove from B_k elements not present in ω_k starting from the j-th position on, none of the elements of B_k can participate in a cycle. Therefore, the resulting graph is a k -forest with its roots the elements in A_k.

(C) Finally, given two different k-forests F_1 and F_2 we claim that $\lambda(F_1) \neq \lambda(F_2)$, where the equality between two points in the space $\Omega_{N_k} \times \mathcal{A}_k \times \{1, ..., N\}$ is defined in the obvious way. Clearly $\lambda(F_1) \neq \lambda(F_2)$ if F_1 and F_2 have different sets of roots. If both k-forests have the same set of roots, then they must differ in at least one edge. If now, we assume to the contrary that $\lambda(F_1) = \lambda(F_2)$ that means that the only way F_1 can differ from F_2 is if there is at least one edge in F_1 not present in F_2 or vice versa. At the same time the equality of the components of $\lambda(F_1)$ and $\lambda(F_2)$ means that the procedure in part (A) removes consecutively exactly the same edges from F_1 and from F_2 which in its turn implies that the two k-forests have the same set of edges, which contradicts our assumption about F_1 and F_2 being different from each other.

(A), (B) and (C) together show that the mapping λ is, indeed, a bijection from $\mathcal{F}_k$ to $\mathcal{B}_k$. Thus, the problem of counting all of the k-forests, $k = 1, ..., N$, on N vertices, reduces to counting the elements of $\cup_{k=1}^{N} \mathcal{B}_k$. One should notice that the mapping λ is not defined for $k = N$ but this is the trivial case where $\mathcal{B}_k$ has just one member, namely the N -forest where each vertex in the graph happens to be a root for one of the N trees composing the forest. There are $\binom{N}{k}$ ways of selecting an element of $A_k \in \mathcal{A}$, k ways of selecting $r \in A_k$, and N^{n-k-1} of sequences in Ω_{N_k}. Therefore, for each fixed k, the cardinality of $\mathcal{B}_k$ is $\binom{N}{k} k N^{N-k-1}$, and since the sets $\mathcal{B}_k$, $k = 1, ..., N$ are pair wise disjoint, the cardinality of $\cup_{k=1}^{N} \mathcal{B}_k$ is

$$\sum_{k=1}^{N} \binom{N}{k} k N^{N-k-1} = (N+1)^{N-1}.$$

The following two examples illustrate the procedures described in part (A) and part (B) of the proof of *Theorem 1*.

Example 3.1: Suppose we are given the state transition diagram in Figure 6. If one applies the procedure from part (A), then one will produce the triple $(\omega_2, A_2, 2)$, where $\omega_2 =$

$4, 6, 1, 4, 1, 2, 4, 7, 9$; $A_2 = \{2, 7\}$, and where the edges $(3, 4)$, $(5, 6)$, $(6, 1)$, $(8, 4)$, $(10, 1)$, $(11, 4)$, $(4, 7)$, and $(12, 9)$ have been consecutively removed from the 2-forest in Figure 6.

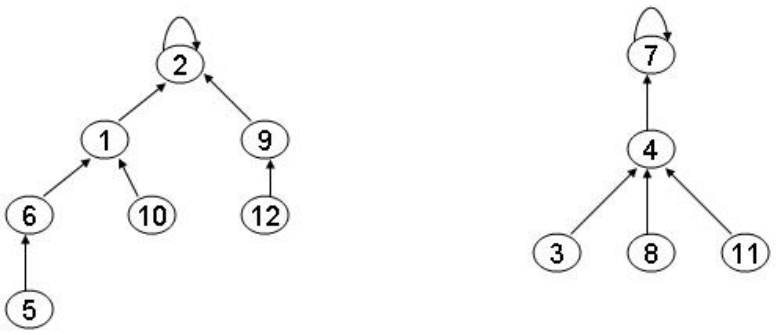


Fig. 6. State Transition Diagram for Example 3.1 .

Example 3.2: Suppose we are given the triple $(\omega_3, A_3, 1)$ where ω_3 = $3, 10, 12, 2, 5, 5, 2, 5$; and where $A_3 = \{1, 5, 9\}$. If one applies the procedure from part (B), then the following edges will be generated in the order they are listed: $(4, 3)$, $(3, 10)$, $(6, 12)$, $(7, 2)$, $(8, 5)$, $(10, 5)$, $(11, 2)$, $(2, 5)$, and $(12, 1)$. Thus, the 3-forest in Figure 7 will be generated.

No restrictions are imposed on the structure of the level sets. Consequently the theorem provides us with an upper bound for the term $\binom{N}{k}\Lambda$ appearing in equation (2). While this upper bound is by no means tight, it is much tighter in comparison to the number of all possible directed graphs on N vertices, N^N. One can easily see that the ratio $(N + 1)^{N-1}/N^N$ is asymptotically equal to e/N. Since $N = 2^n$, the probability mass decreases exponentially relative to the number of genes n . This shows that a brute force search for an acceptable BN by randomly filling in a BN truth table has very little chance of success. Therefore, if one wants to efficiently generate a BN with the desired character-

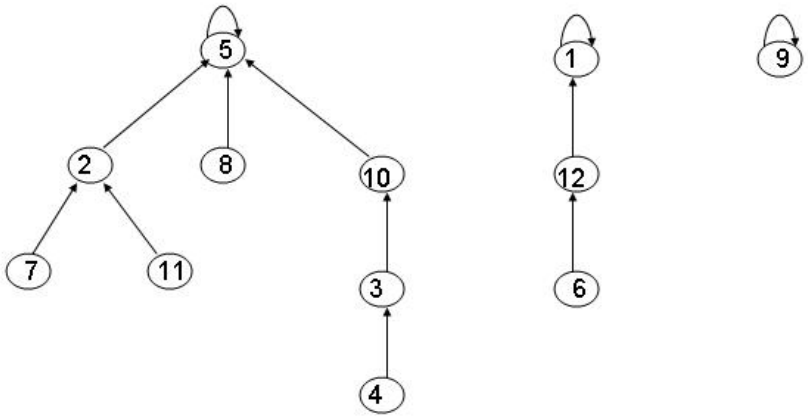


Fig. 7. State Transition Diagram for Example 3.2 .

istics, one has to incorporate information from the state transition diagram, as well as the information about the predictor set of the network, into the algorithm.

2. Design of Efficient Algorithms

We present two algorithms to generate Boolean networks given attractor and connectivity information. The first algorithm works directly with the truth table, incorporating at the same time the information about the attractor set, as well as the information about the predictor set of the BN. There is no control over the level set structure, and the transition diagram generated by the algorithm has to be checked for the presence of cycles. We present the algorithms for the case of singleton attractors and provide the adaptation for multiple-state cyclic attractors in Appendix A.

a. Algorithm 1

STEP1: Randomly generate a set of k attractor states. If STEP1 has been repeated more than a pre-specified number of times terminate the algorithm.

STEP2: Randomly pick up a predictor set W, where each W_i has not less than m and not more than M elements. If STEP2 has been repeated more than a pre-specified number of times go back to STEP1.

STEP3: Check if the selected attractor set is compatible with W, i.e., only the attractor set of the state transition diagram is checked for compatibility against W. If the attractor set is not compatible with W go back to STEP2, otherwise continue to STEP4.

STEP4: Fill in the entries of the truth table that correspond to the attractors generated in STEP1. Using the predictor set W randomly fill in the remaining entries of the truth table. If STEP4 has been repeated more than a pre-specified number of times go back to STEP2.

STEP5: Search for cycles of any length in the state transition diagram $\tilde{\Gamma}$ that is associated with the truth table generated in STEP4. If a cycle is found go back to STEP4, otherwise continue to STEP6.

STEP6: If $\tilde{\Gamma}$ has less than l or more than L level sets go back to STEP4, otherwise continue to STEP7.

STEP7: Save the generated BN and terminate the algorithm.

The second algorithm employs a state transition diagram $\tilde{\Gamma}$ that satisfies the design goals about attractor structure and level-set structure, and checks if the truth table associated with $\tilde{\Gamma}$ has a predictor set W satisfying the design goals.

b. Algorithm 2

STEP1: Randomly generate a state transition diagram $\tilde{\Gamma}$ that satisfies the design goals about the attractor structure and level set structure. If STEP1 has been repeated more than a pre-specified number of times terminate the algorithm.
STEP2: Fill in the truth table using $\tilde{\Gamma}$.
STEP3: If there is at least one W_i in the predictor set W given by the truth table that has less than m or more than M elements go back to STEP1, otherwise continue to STEP4.
STEP4: Save the generated BN and terminate the algorithm.

The following examples provide walk-through illustrations to show how algorithm 1 and 2 works in the particular case of 3 genes.

Example 3.3(Algorithm 1): Suppose that $k = 2$, $m = 1$, $M = 2$, $l = 1$, and $L = 5$. Next, suppose that the states 000 and 011 are generated by STEP1. STEP2: Suppose W is generated where $W_1 = \{x_2, x_3\}$, $W_2 = \{x_1, x_3\}$, $W_3 = \{x_1, x_2\}$. STEP3: Table I shows that the attractors generated in STEP1 are compatible with W. The remaining entries $a_1, ..., a_6$ in the truth table are filled in randomly in the next step of the algorithm. One can notice certain patterns in the entries in each one of the three columns of the table. These reflect the structure of the predictor set W, and reduce the number of the possible ways to randomly fill in the missing entries during the next step of the algorithm. On the other hand, if the attractors generated in STEP1 were 000 and 100, then for the predictor function of the first gene x_1, we must have $f_1(0, 0) = 0$, while from the second attractor, we get $f_1(0, 0) = 1$, which is a contradiction. Therefore that attractor set is not compatible with the set W generated in STEP2.
STEP4: Here we randomly fill in the remaining entries of the truth table. Suppose that this produces $a_1 = 0, a_2 = 0, a_3 = 1, a_4 = 0, a_5 = 0, a_6 = 1$. This selection produces the

following transitions in the state transition diagram: $0 \to 0; 1 \to 2; 2 \to 1; 3 \to 3; 4 \to 2; 5 \to 0; 6 \to 3; 7 \to 1$, where we have used the decimal representation of the states. It is clear that during STEP5 of the algorithm the cycle $1 \to 2; 2 \to 1$ will be discovered, which will cause the BN generated by the present truth table to be discarded, and we will be returned to STEP4.

On the other hand if we had $a_1 = 0, a_2 = 1, a_3 = 1, a_4 = 0, a_5 = 0, a_6 = 1$ produced in STEP4, then the transitions in the state transition diagram would be $0 \to 0; 1 \to 2; 2 \to 5; 3 \to 3; 4 \to 2; 5 \to 0; 6 \to 7; 7 \to 1$. Since the only cycles here are those within the attractor set, STEP5 of the algorithm will take us to STEP6. STEP6 will detect that there are 5 level sets, and this will take us to STEP7.

TABLE I

TRUTH TABLE FOR EXAMPLE 3.3

$x_1\ x_2\ x_3$	f_1	f_2	f_3
0 0 0	0	0	0
0 0 1	a_1	1	0
0 1 0	a_2	0	1
0 1 1	0	1	1
1 0 0	0	a_3	a_4
1 0 1	a_1	a_5	a_4
1 1 0	a_2	a_3	a_6
1 1 1	0	a_5	a_6

Example 3.4(Algorithm 2): Suppose that $k = 2$, $m = 1$, $M = 2$, $l = 1$ and $L = 3$.

Next, suppose that the transition diagram shown in Figure 8 was randomly generated in STEP1.

The truth table resulting from STEP2 is shown in Table II.

STEP3: It is clear from this truth table that $W_1 = \{x_1, x_2, x_3\}$, and since it has more than $M = 2$ elements the algorithm returns to STEP1. On the other hand if the transition diagram shown in Figure 9 was generated in STEP1, then STEP2 would produce the truth table shown in Table III.

Now each W_i; $i = 1, 2, 3$ has no more than $m = 2$ elements, and the algorithm successfully terminates producing a BN with the truth table shown in Table III and state transition diagram from Figure 9.

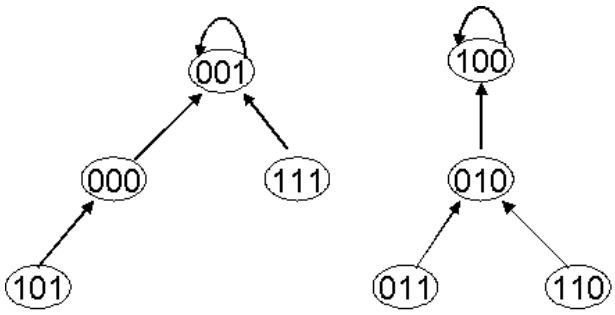


Fig. 8. First State Transition Diagram for Example 3.4 .

3. Comparison between the Two Algorithms

Several simulations were carried out to evaluate the performance of the two algorithms. Table IV shows the performance of Algorithm 1 for the case of $n = 6$, $2 \leq k \leq 4$, $m = 1$, $l = 1$, and $L = 2^6 - 1$. The number of maximum repetitions of STEP1, STEP2, and STEP4

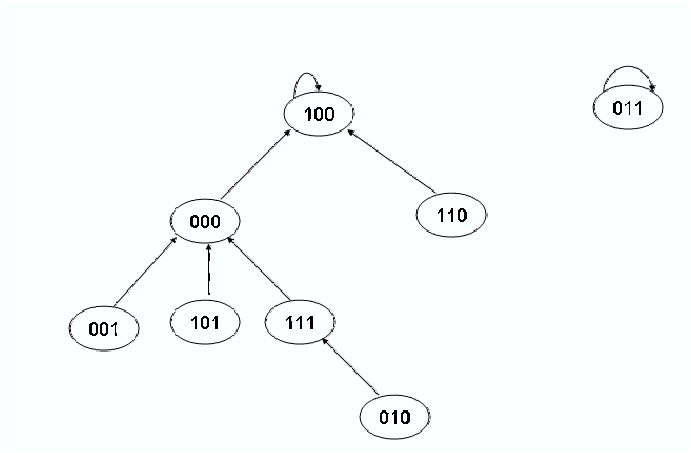


Fig. 9. Second State Transition Diagram for Example 3.4 .

were set to 10, 20 and 500 respectively. The total execution time for this simulation was 13123.875 seconds or roughly 3.5 hrs on a 2.4 GHz P4 Intel Xeon Processor.

Table V shows the performance of Algorithm 1 for the case of $n = 10$, $1 \leq k \leq 6$, $m = 1$, $l = 1$, and $L = 2^{10} - 1$. The number of maximum repetitions of STEP1, STEP2, and STEP4 were set to 10, 15 and 1000 respectively. The execution time for this simulation was 58842.5 seconds or around 16 hours on an identical machine.

The significant increase in the run time for the case of 10 genes can be attributed to two major factors: first, the NP-completeness nature of the cycle search performed in STEP5; and second, the low probability mass of k-forests in the space of all directed graphs, as was already discussed in Section 2. Table VI shows the performance of Algorithm 2 for the case $n = 3$, $1 \leq k \leq 2$, $m = 1$, $M = 2$, $l = 2$, and $L = 5$. There were 1000 BNs generated in STEP1 and the simulation time was 4.01 seconds. One can notice the low frequency of successfully generated BN even for such a small number of genes. The simulation for the case $n = 6$, $2 \leq k \leq 4$, $m = 1$, $M = 5$, $l = 4$, and $L = 15$ confirms that observation: the algorithm did not generate any BN during the first 3×10^6 iterations. It took 78329.2

TABLE II

FIRST TRUTH TABLE FOR EXAMPLE 3.4

Gene Values	f_1	f_2	f_3
0 0 0	0	0	1
0 0 1	0	0	1
0 1 0	1	0	0
0 1 1	0	1	0
1 0 0	1	0	0
1 0 1	0	0	0
1 1 0	0	1	0
1 1 1	0	0	1

seconds or approximately 21hrs to run this many iterations.

The reason for such a huge difference in the performance of the two algorithms is the fact that the state transition diagrams generated by Algorithm 1 have a very small probability mass in the space of all k-forests, $k = 1, ..., N$ on N vertices. One can easily see that when each gene predictor set W_i is required to have exactly m elements, the number of possible state transition diagrams generated by Algorithm 1 is $\binom{n}{m}^n N^{2^m}$. Using Theorem 1 one can obtain an estimate of the probability mass of the state transition diagrams generated by Algorithm 1 within the space of all k-forests, $k = 1, ..., N$:

$$\frac{\binom{n}{m}^n N^{2^m}}{(N+1)^{N-1}}$$

TABLE III

SECOND TRUTH TABLE FOR EXAMPLE 3.4

Gene Values	f_1	f_2	f_3
0 0 0	1	0	0
0 0 1	0	0	0
0 1 0	1	1	1
0 1 1	0	1	1
1 0 0	1	0	0
1 0 1	0	0	0
1 1 0	1	0	0
1 1 1	0	0	0

For the case $n = 6$, $m = 5$ this ratio is approximately 1.7911×10^{-52}.

B. Design of Probabilistic Boolean Networks

The algorithms produce many distinct networks satisfying a given set of constraints. The presence of multiple solutions allows for optimization procedures when designing PBNs from microarray data. In this section, we describe a procedure for designing a PBN from microarray data. The sizes of the basins are used to select BNs from a group of generated networks and to combine them in a PBN whose steady-state distribution closely matches the observed frequency distribution of the data. The assumption that these data correspond to the steady-state of the underlying gene regulatory system provides a reference point of

TABLE IV

SIMULATIONS FOR ALGORITHM1 WITH 6 GENES

n	M	BNs saved at STEP7	BNs searched in STEP5
6	1	1267	7670
6	2	1375	10160
6	3	2396	19124
6	4	1399	27590
6	5	1960	35060
6	6	1704	37550

how closely the dynamics of a generated PBN approximate the data. The designed PBN should have these data points as attractors – and no other attractors because there is no reason in the data for having other attractors. We focus on singleton attractors.

The design procedure begins by selecting a random number N_1 between 2 and 5, and then randomly selecting N_1 distinct states as singleton attractors from the original data according to the data frequency distribution. Repeating this procedure 10 times yields 10

TABLE V

SIMULATION FOR ALGORITHM1 WITH 10 GENES

n	M	BNs saved at STEP7	BNs searched in STEP5
10	9	80	30090

TABLE VI

SIMULATIONS FOR ALGORITHM 2 WITH 3 GENES

n	M	BNs saved at STEP4
3	1	5
3	2	43

sets, $A_1, A_2, ..., A_{10}$, of singleton attractors, with A_i possessing N_i attractors, $2 \leq N_i \leq 5$ and $i = 1, 2....10$. After that, algorithm 1 is employed to generate 100 Boolean Networks, $\mathcal{B}_{i1}, \mathcal{B}_{i2}, ..., \mathcal{B}_{i,100}$, for each of the 10 attractor sets. The generated networks have state transition diagrams satisfying two additional constraints. First, the number of their level sets range from 2 to 10. This constraint manifests the understanding that in the underlying gene regulatory network the steady-state/fixed points of the system are not achieved with too few or too many consecutive transitions. The second constraint is that all gene predictor sets have between 1 and 3 genes, the number being randomly set.

The BNs generated for each one of the 10 attractor sets are then used as a sample space for the selection of a PBN whose steady-state distribution matches closely the frequency distribution of the data in the mean-square error (MSE) sense. One BN from each group of 100 BNs is randomly selected, and the basin size of each singleton attractor is calculated. These numbers are used as estimates of the steady-state probabilities of the corresponding attractors (keeping in mind that very little time is spent in transient states and that random perturbations and switching randomly put the network in different basins). For example, if the Boolean network $\mathcal{B}_{ij}$ has attractor states $\mathbf{a}_1^{ij}$, $\mathbf{a}_2^{ij}$ ... $\mathbf{a}_{N_{ij}}^{ij}$ with corresponding basin sizes S_1^{ij}, S_2^{ij} ... $S_{N_{ij}}^{ij}$, respectively, then our estimate of the steady-state probability for

the attractror $\mathbf{a}_l^{ij}$ is given by $\pi_{ij}(\mathbf{a}_l^{ij}) = S_l^{ij} / \sum_{k=1}^{N_{ij}} S_k^{ij}$. One can take the average of these estimates of the steady-state probabilities of the singleton attractor $\mathbf{a}_l^{ij}$ over the 10 BNs comprising the PBN as an estimate of the steady-state probability of that attractor in the PBN. If a particular singleton attractor is not present in a constituent BN, then its contribution to the steady-state probability is set to 0.

Continuing in this fashion, one obtains an estimate of the steady-state probabilities of each one of the data states used in the generation of a PBN. Let us denote these states by $\mathbf{b}_1, \mathbf{b}_2, ...\mathbf{b}_m$ and their corresponding estimated steady-state probabilities by $\pi_1, \pi_2...., \pi_m$. The procedure calculates the MSE between $\pi_1, \pi_2...., \pi_m$ and $f_1, f_2, ..., f_m$, where f_i is the relative frequency of $\mathbf{b}_i$ in the sample. The designed PBN is selected as the one that minimizes the MSE among a randomly selected subset of 10000 PBNs from the set of all possible PBNs that can be generated using the BNs produced by algorithm 1 for the selected attractor sets. We settle on 10000 PBNs because an exhaustive search is prohibitive, there being a total of 100^{10} possible PBNs.

1. Melanoma PBN Design

The gene-expression profiles used in this study result from the study of 31 malignant melanoma samples [48]. For the study, messenger RNA was isolated directly from melanoma biopsies, and fluorescent cDNA from the message was prepared and hybridized to a microarray containing probes for 8150 cDNAs (representing 6971 unique genes). The 7 genes WNT5A, pirin, S100P, RET1, MART1, HADHB and STC2 used here for the model were chosen from a set of 587 genes from the data set that have been subjected to an analysis of their ability to cross predict each other's state in a multivariate setting [39]. The gene-expression profiles were binarized to arrive at 31 binary vectors with 7 columns corresponding to the selected 7 genes. The frequency distribution of the 18 distinct binary data vectors is shown in Figure 10. The assumption that these data correspond to the steady-

state of the underlying gene regulatory system implies that those and only those 18 distinct data vectors should appear as attractors in the generated PBN. This condition is guaranteed by the design procedure.

Figure 10 shows the portion of histogram (the data states only) of the steady-state distribution (after a long run) of the designed PBN, with $q = 0.001$ and $p = 0.001$, and of the frequency distribution of the data states. The steady-state distribution of the PBN closely matches (in the MSE sense) the frequency distribution observed in the data.

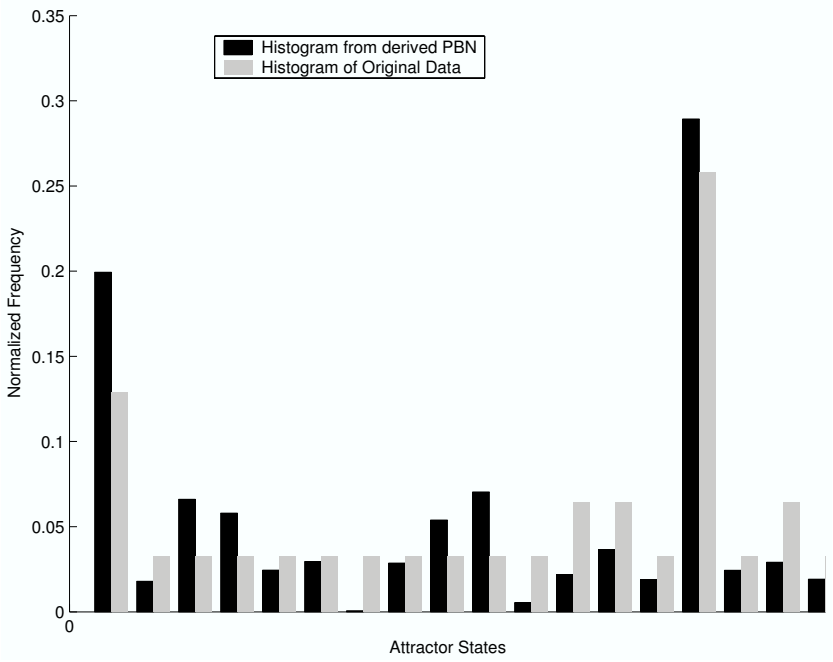


Fig. 10. Histogram for Original and Generated PBN [8].

CHAPTER IV

CONTROL OF PROBABILISTIC BOOLEAN NETWORKS *

From a translational perspective, the ultimate objective of genetic regulatory network modeling is to use the network to design different approaches for affecting network dynamics in such a way as to avoid undesirable phenotypes, for instance, cancer. In this chapter we present results on intervention in the context of *Probabilistic Boolean Networks*(PBNs). Given a PBN, the transition from one state to the next takes place in accordance with certain transition probabilities and their dynamics, and hence intervention, can be studied in the context of homogeneous Markov chains with finite state spaces.

A major goal of functional genomics is to screen for genes that determine specific cellular phenotypes (disease) and model their activity in such a way that normal and abnormal behavior can be differentiated. The pragmatic manifestation of this goal is the development of therapies based on the disruption or mitigation of aberrant gene function contributing to the pathology of a disease. Mitigation would be accomplished by the use of drugs to act on the gene products. Engineering therapeutic tools involves synthesizing nonlinear dynamical networks, analyzing these networks to characterize gene regulation, and developing intervention strategies to modify dynamical behavior. For instance, changes in network connectivity or functional relationships among the genes in a network, via mutations or rearrangements, can lead to steady-state behavior associated with tumorigenesis, and this is likely to lead to a cancerous phenotype unless corrective therapeutic intervention is applied.

---

*Parts of this chapter are reprinted, with permission, from (i) ”Intervention in context sensitive probabilistic Boolean networks”, R. Pal, A. Datta, M. L. Bittner, and E. R. Dougherty, *Bioinformatics*, vol. 21, pp. 12111218, 2005 and (ii) ©2006 IEEE. Reprinted, with permission, from ”Optimal Infinite Horizon Control for Probabilistic Boolean Networks”, Pal, R., Datta, A. and E. R. Dougherty, *IEEE Transactions on Signal Processing*, 2006, Vol. 54, no. 6, 2375:2387

To date, intervention studies using PBNs have used three different approaches: (i) resetting the state of the PBN, as necessary, to a more desirable initial state and letting the network evolve from there [10]; (ii) changing the steady-state (long-run) behavior of the network by minimally altering its rule-based structure [11]; and (iii) manipulating external (control) variables that alter the transition probabilities of the network and can, therefore, be used to desirably affect its dynamic evolution [12]. In this chapter, we extend the control-theoretic approach in two important directions. First, whereas the original control-theoretic approach has been developed in the framework of *instantaneously random* PBNs, here we design optimal intervention for *context-sensitive* PBNs [13]. This extension is significant because the latter class more closely models small biological subnetworks whose logical behavior is affected by conditions outside the genes represented in the model network. Second, the earlier finite horizon results are extended to the infinite horizon case in an effort to alter the steady-state behaviour of the genetic regulatory network. Moreover, the stationary policies obtained in case of infinite horizon control are much easier to implement than a policy that changes with time.

Probabilistic Boolean networks can be used for studying the dynamic behavior of gene regulatory networks. Once a probability distribution vector has been specified for the initial state, the probability distribution vector evolves according to Eq. 2.10. From this perspective PBNs are *descriptive* in nature. There is no mechanism for controlling the evolution of the probability distribution vector. For treatment or intervention purposes, we are interested in working with PBNs in a *prescriptive* fashion, where the transition probabilities of the associated Markov chain depend on certain external variables, whose values can be chosen to make the probability distribution vector evolve in some desirable manner.

The use of such external variables makes sense from a biological perspective. For instance, in the case of diseases like cancer, external treatment inputs such as radiation,

chemotherapy, etc. may be employed to move the state probability distribution vector away from one associated with uncontrolled cell proliferation or markedly reduced apoptosis. The variables could also include genes that serve as external master-regulators for all the genes in the network. To be consistent with the binary nature of the expression status of individual genes in a PBN, we will assume that these variables (*control inputs*) can take on only the binary values 0 or 1. The values of the individual control inputs can be changed from one time step to another in an effort to make the network behave in a desirable fashion.

A. Finite-Horizon Control in Context-Sensitive PBNs

Let L denote the number of BNs constituting the context-sensitive PBN and p denote the probability that the value of any particular gene undergoes a random perturbation and q denote the probability that the network function switches at any given time point.

For a context-sensitive PBN, the state $z(t)$ at time t could be originating from any one of the L possible networks. In order to keep track of the network emitting a particular state let us redefine the states by incorporating the network number inside the state label. Since we have L different BNs forming the PBN, the total number of states becomes $2^n L$ and let us label these states as S_0, $S_1, \cdots, S_{2^n L-1}$ where for each $k = 1, 2, \cdots, L$, states $S_{2^n(k-1)}, S_{2^n(k-1)+1}, ..., S_{2^n k-1}$ belong to network k. Equivalently $S_{2^n(k-1)+i}$ corresponds to z_{k_i} where z_{k_i} is the decimal representation of the ith state in the network k. Let the redefined state at time t be denoted by $w(t)$.

1. Transition Probabilities of Context-Sensitive PBNs

We now derive expressions for the transition probabilities in a context-sensitive PBN subject to perturbations by recognizing that the following mutually exclusive events can occur at any time point t:

(1) The current network function is applied, the PBN transitions accordingly, and the network function remains the same for the next transition.

(2) The current network function is applied, the PBN transitions accordingly, and a new network function is selected for the next transition.

(3) There is a random perturbation and the network function remains the same for the next transition.

(4) There is a random perturbation and a new network function is selected for the next transition.

Assuming that the individual genes perturb independently, and letting $mod(v, w)$ denote the remainder left over when v is divided by w, we consider two cases for determining the transition probability of going from state a to state b:

Case 1. $[a/2^n] = [b/2^n]$, meaning $2^n(k-1) \leq a, b \leq 2^n k - 1$ for the same k. This corresponds to the events (1) and (3) above and the transition probabilities are given by

$$Pr(w(t+1) = b | w(t) = a) = (1-q)(1-p)^n f_{k,a,b} + (1-q)(1-p)^{n-h} p^h s(h) \quad \textbf{(4.1)}$$

where h is the Hamming Distance between $mod(a, 2^n)$ and $mod(b, 2^n)$, i.e. the number of genes which differ between the two states,

$$f_{k,a,b} = \begin{cases} 1 & \text{if } a \text{ transitions to } b \text{ in a single step in network } k \\ 0 & \text{otherwise} \end{cases}$$

and

$$s(h) = \begin{cases} 0 & \text{if } h = 0 \\ 1 & \text{otherwise} \end{cases}$$

The first term in Eq. (4.1) corresponds to event (1) above, where $1-q$ is the probability that

the network selection does not change, $(1-p)^n$ is the probability that none of the n genes undergoes a perturbation, we assume that network selection and random gene perturbation are independent events, and $f_{k,a,b} = 1$ if that particular transition is possible in the kth Boolean network. The second term corresponds to event (3), where h genes have to be perturbed to go from state a to state b.

Case 2. $2^n(k_1-1) \leq a \leq 2^n k_1 - 1$ and $2^n(k_2-1) \leq b \leq 2^n k_2 - 1$, where $k_1 \neq k_2$. This corresponds to events (2) and (4) above and the transition probabilities are given by

$$Pr(w(t+1)=b|w(t)=a) = q\frac{c_{k_2}}{\sum_{i=1,i\neq k_1}^{L} c_i}(1-p)^n f_{k_1,a,b} + q\frac{c_{k_2}}{\sum_{i=1,i\neq k_1}^{L} c_i}(1-p)^{n-h}p^h s(h). \tag{4.2}$$

If we define

$$g(a,b) = \begin{cases} 1 & \text{if } [a/2^n] - [b/2^n] = 0 \\ 0 & \text{otherwise} \end{cases}$$

then a unified transition probability expression encompassing the two cases is given by

$$\begin{aligned} &Pr(w(t+1)=b|w(t)=a) = \\ &\quad [(1-q)(1-p)^n f_{k,a,b} + (1-q)(1-p)^{n-h}p^h s(h)]g(a,b) \\ &\quad + [q\frac{c_{k_2}}{\sum_{i=1,i\neq k_1}^{L} c_i}(1-p)^n f_{k_1,a,b} + q\frac{c_{k_2}}{\sum_{i=1,i\neq k_1}^{L} c_i}(1-p)^{n-h}p^h s(h)](1-g(a,b)). \end{aligned} \tag{4.3}$$

By letting a and b range over all integers from 0 to $2^n L - 1$ and using Eq. (4.3), we can determine all the entries of the $2^n L \times 2^n L$ matrix of transition probabilities.

In practice, it will likely be impossible to detect the Boolean network from which the current gene activity profile is being emitted. In most cases, we will only have knowledge of the states of the genes. To handle such situations, we can derive an expression for the transition probability from state s_2 to state s_1, where these states run from 0 to $2^n - 1$ and

reflect only the expression status of the n-gene state vector:

$$
\begin{aligned}
&\Pr[z(t+1) = s_1 | z(t) = s_2] \\
&\quad = \sum_{i=1}^{k} Pr[z(t+1) = s_1, s_2 \text{ belongs to network } i | z(t) = s_2] \\
&\quad = \sum_{i=1}^{k} Pr[z(t+1) = s_1 | z(t) = s_2, s_2 \text{ belongs to network } i] \\
&\qquad\qquad .Pr[s_2 \text{ belongs to network } i] \\
&\quad = \sum_{i=1}^{k} Pr[z(t+1) = s_1 | w(t) = s_2 + 2^n(i-1)].c_i \\
&\quad = \sum_{i=1}^{k} \sum_{j=1}^{k} c_i.Pr[w(t+1) = s_1 + 2^n(j-1) | w(t) = s_2 + 2^n(i-1)] \qquad (4.4)
\end{aligned}
$$

where s_1 and s_2 run from 0 to $2^n - 1$. Note that here state s_1 is equivalent to the distinct states $s_1, s_1 + 2^n, ...s_1 + (L-1)2^n$ in the previous $2^n L$ formulation. Similarly s_2 here is equivalent to $s_2, s_2 + 2^n, ...s_2 + (L-1)2^n$ in the earlier formulation. By letting s_1 and s_2 range from 0 to $2^n - 1$ and using Eq. (4.4), we can derive the $2^n \times 2^n$ transition probability matrix A corresponding to the averaged context-sensitive PBN.

Substituting Eq. 4.3 into Eq. 4.4 yields

$$
\begin{aligned}
&\Pr[z(t+1)=s_1|z(t)=s_2]= \\
&\sum_{k_1=1}^{L} c_{k_1}\Big[(1-q)(1-p)^n f_{k_1,s_2,s_1}+(1-q)(1-p)^{n-h}p^h s(h) \\
&+q\sum_{k_2=1,k_2\neq k_1}^{L}\frac{c_{k_2}}{\sum_{l=1,l\neq k_1}^{L} c_l}(1-p)^n f_{k_1,s_2,s_1}+ \\
&q\sum_{k_2=1,k_2\neq k_1}^{L}\frac{c_{k_2}}{\sum_{l=1,l\neq k_1}^{L} c_l}(1-p)^{n-h}p^h s(h)\Big] \\
&=(1-p)^n\Big\{\sum_{k_1=1}^{L} c_{k_1}\Bigg((1-q)f_{k_1,s_2,s_1}+q\frac{\sum_{k_2=1,k_2\neq k_1}^{L} c_{k_2}}{\sum_{l=1,l\neq k_1}^{L} c_l}f_{k_1,s_2,s_1}+ \\
&(1-q)\left(\frac{p}{1-p}\right)^h s(h)+q\frac{\sum_{k_2=1,k_2\neq k_1}^{L} c_{k_2}}{\sum_{l=1,l\neq k_1}^{L} c_l}\left(\frac{p}{1-p}\right)^h s(h)\Bigg)\Big\} \\
&=(1-p)^n\Big\{\sum_{k_1=1}^{L} c_{k_1}\left(f_{k_1,s_2,s_1}+\left(\frac{p}{1-p}\right)^h s(h)\right)\Big\} \\
&=(1-p)^n\Big\{\sum_{k_1=1}^{L} c_{k_1}f_{k_1,s_2,s_1}\Big\}+(1-p)^n\left(\frac{p}{1-p}\right)^h s(h) \qquad (4.5)
\end{aligned}
$$

Let us denote by $B_1, B_2, \cdots, B_L$, the transition matrices of the individual Boolean Networks. Then B_v, $v \in [1, ..., L]$ represent deterministic transition matrices and hence each B_v has a single non-zero entry of 1 in each row. The second term of Eq. 4.5 is independent of the constituent Boolean Networks or their selection probabilities and depends only on the perturbation probability p, number of genes n and the hamming distance between states h (which can be determined when n is known). Consequently, the probability transition matrix P of the averaged *context-sensitive* PBN composed of k Boolean Networks is of the form

$$P=(1-p)^n\sum_{v=1}^{L} c_v B_v + D^{n,p} \qquad (4.6)$$

where the entries of B_v are generated from the first term in Eq. 4.5 and $D^{n,p}$ represents the entries corresponding to the second term of Eq. 4.5.

Furthermore, the matrix $D^{n,p}$ has the form

$$D^{n,p} = (1-p)^n \begin{bmatrix} 0 & \frac{p}{1-p} & \frac{p}{1-p} & (\frac{p}{1-p})^2 & \cdots & (\frac{p}{1-p})^n \\ \frac{p}{1-p} & 0 & (\frac{p}{1-p})^2 & \frac{p}{1-p} & \cdots & (\frac{p}{1-p})^{n-1} \\ .. & .. & .. & .. & \cdots & .. \\ .. & .. & .. & .. & \cdots & .. \\ .. & .. & .. & .. & \cdots & .. \\ (\frac{p}{1-p})^n & (\frac{p}{1-p})^{n-1} & (\frac{p}{1-p})^{n-1} & (\frac{p}{1-p})^{n-2} & \cdots & 0 \end{bmatrix}.$$

The individual $2^n \times 2^n$ terms of $D^{n,p}$ for $i = 0, 1, ..., 2^n - 1$ and $j = 0, 1, ..., 2^n - 1$ are

$$D^{n,p}(i,j) = D^{n,p}(j,i) = (1-p)^n \begin{cases} 0 & \text{if } i = j \\ (\frac{p}{1-p})^{h(i,j)} & \text{otherwise} \end{cases}$$

where $h(i,j)$ = no. of bits different in the binary representation of i and j.

2. Optimal Control of Context-Sensitive PBNs

In this section, we consider the problem of external control in a context-sensitive PBN. Towards this end, suppose that a PBN with n genes has m control inputs, u_1, u_2, $\cdots, u_m$, each of which can take on only the binary values 0 or 1. Then at any time t, the row vector $u(t) \triangleq [u_1(t), u_2(t), \cdots, u_m(t)]$ describes the complete status of all the control inputs. $u(t)$ can take on all binary values from $[0, 0, \cdots, 0]$ to $[1, 1, \cdots, 1]$. One can equivalently represent the control input status using the decimal number

$$v(t) = \sum_{i=1}^{m} 2^{m-i} u_i(t). \tag{4.7}$$

As $u(t)$ takes on binary values from $[0, 0 \cdots, 0]$ to $[1, 1, \cdots, 1]$, the variable $v(t)$ ranges from 0 to $2^m - 1$. We can equivalently use $v(t)$ as an indicator of the complete control input status of the PBN at time t.

If a control action is applied, then the transition probability expressions will change. Suppose that our control action consists of forcibly altering the value of a single gene, g, from 0 to 1 or from 1 to 0. Thus, $m = 1$ here. Then the new transition probabilities with control, denoted by $Prc1$, are given by

$$\begin{aligned} Prc1(w(t+1) = b|w(t) = a) &= Pr(w(t+1) = b|w(t) = a + 2^{n-g}) func(a) \\ &+ Pr(w(t+1) = b \quad | \quad w(t) = a - 2^{n-g})(1 - func(a)) \end{aligned} \tag{4.8}$$

where

$$func(a) = \begin{cases} 1 & \text{if state of gene } g \text{ is } 0 \text{ for } a \\ 0 & \text{if state of gene } g \text{ is } 1 \text{ for } a \end{cases}$$

and the transition probabilities, Pr, without control are given by Eq. (4.3).

Here, a and b range over 0 through $2^n L - 1$. As shown in Chapter II, we can reduce the dimension of the state space by replacing the w's in Eq. (4.8) by z's and using Eq. (4.4) to determine the transition probabilities without the control action:

$$\begin{aligned} &Prc1(z(t+1) = b|z(t) = a) = \\ &\quad Pr(z(t+1) = b|z(t) = a + 2^{n-g}) func(a) + \\ &\quad Pr(z(t+1) = b|z(t) = a - 2^{n-g})(1 - func(a)) \end{aligned} \tag{4.9}$$

By letting a and b vary over 0 to $2^n - 1$ and making use of Eq. (4.9), we can determine the $2^n \times 2^n$ matrix $A(v(t))$ of control-dependent transition probabilities.

In the rest of this section, we formulate and solve the control problem assuming 2^n states and the availability of full state information. The same development can be carried out for the $2^n L$ state formulation if we simultaneously have the gene state information and the network labels. As shown in [12], the one-step evolution of the probability distribution vector in the case of a PBN containing 2^n states with control inputs takes place according to the equation:

$$pd(t+1) = pd(t)A(v(t)) \tag{4.10}$$

where $pd(t)$ is the 2^n dimensional state probability distribution vector and $A(v(t))$ is the $2^n \times 2^n$ matrix of control-dependent transition probabilities determined by Eq. (4.9). Since the transition probability matrix is a function of the control input $v(t)$, the evolution of the probability distribution vector of the PBN with control now depends not only on the initial distribution vector but also on the values of the control input at different time steps. Furthermore, intuitively it appears possible to make the states of the network evolve in a desirable fashion by appropriately choosing the control input at each time step.

These ideas have been formalized in [12] to arrive at the following finite horizon optimization problem. Given an initial state z_0:

$$\min_{\mu_0,\mu_1,\cdots,\mu_{M-1}} E\left[\sum_{t=0}^{M-1} C_t(z_t, \mu_t(z_t)) + C_M(z_M)\right] \tag{4.11}$$

subject to $Pr(z(t+1) = j | z(t) = i, v(t))$, given by Eq. (4.9), where

- M represents the treatment/intervention window;
- $\mu_t : [0, 1, 2, \cdots, 2^n - 1] \rightarrow [0, 1, 2, \cdots, 2^m - 1]$,

  $t = 0, 1, 2, \cdots, M-1$ are functions mapping the state space into the control space;
- $C_t(z_t, v_t)$ is the one step cost of applying the control v_t at state z_t;

- and $C_M(z_M)$ is the terminal cost associated with the state z_M.

As discussed in [12], the consideration of such an optimization problem can be naturally motivated in the context of cancer treatment applications where one must choose between a number of alternative treatments to be applied over a finite horizon of time. Once input from biologists/clinicians has been used to select an appropriate cost function and an appropriate treatment window, the control problem is essentially reduced to that of controlling a Markov Chain over a finite horizon.

The dynamic programming solution to Eq. (4.11) is given by:

$$J_M(z_M) = C_M(z_M) \tag{4.12}$$

$$\begin{aligned} &J_t(z_t) = \\ &\min_{v_t \in \{0,1,\cdots,2^m-1\}} \left[C_t(z_t, v_t) + \sum_{j=0}^{2^n-1} Pr(z_t|j, v_t).J_{t+1}(j) \right] \\ &t = 0, 1, \cdots, M-1. \end{aligned} \tag{4.13}$$

[49, 12]. If $v_t^* = \mu_t^*(z_t)$ minimizes the right hand side of Eq. (4.13) for each z_t and t, then the control law $\pi^* = \left\{\mu_0^*, \mu_1^*, \cdots, \mu_{N-1}^*\right\}$ is optimal.

The optimal control problem, Eq. (4.11), and its solution, Eqs. (4.12) and (4.13), are from a very general setting; however, in our case, the class of allowable controls is severely constrained since our control action consists of forcibly altering the expression status of only a *single* gene. This limited control objective is dictated primarily by limitations on the kind of interventions that appear to be within the realm of biological possibility.

3. Selecting the Control Gene

Given a particular target gene, there may be several genes that are good predictors for it. Among a set of predictors for a particular gene, some of them may have more impact on the value of the target gene than others. For instance, in cancer studies it has been shown that p53 has a more profound effect on the cell cycle regulator gene WAF1/p21 than other predictors of WAF1, such as AP2 or BRCA1 [50]. In view of this, one can define the *influence* of the variable x_j on the Boolean function f [5]. To do so, let D be the probability mass distribution over the states of a Boolean network and let $\frac{\partial f(x)}{\partial x_j}$ be the partial derivative of the Boolean function f with respect to the argument x_j. Then the influence of x_j on f is defined by

$$I_j(f) = E_D[\frac{\partial f(x)}{\partial x_j}] = Pr\{\frac{\partial f(x)}{\partial x_j} = 1\} = Pr\{f(x) \neq f(x^{(j)})\} \tag{4.14}$$

where $x^{(j)}$ is the same as x except that the jth component is toggled. In this dissertation, we will assume that the distribution D is uniform.

The main idea behind the influence definition is to quantify the amount by which the gene x_j affects the value of the function f. If the value of the function f changes on toggling the value of gene x_j for most gene activity profiles x, then the influence of the jth gene on f is high. For the case of PBNs, let F_i be the set of predictors for gene x_i with corresponding probabilities $c_1^{(i)}, ..., c_{l(i)}^{(i)}$. Let $I_k(f_j^{(i)})$ be the influence of variable x_k on the predictor $f_j^{(i)}$. Then the influence of gene x_k on gene x_i is given by [5]

$$I_k(x_i) = \sum_{j=1}^{l(i)} c_j^{(i)} I_k(f_j^{(i)}) \tag{4.15}$$

We can use the *influence* to select the control gene. For example, suppose we have treatments $d_1, d_2...., d_r$ that can affect genes $g_1, g_2.....g_r$, respectively. Biological or economic considerations may constrain us to use only one treatment at a time. Then we can use the gene that has the highest influence on the target gene g_t. The influence can be directly calculated from the PBN as given by the previous formula or it can be approximated from the observed gene activity profiles. The hope is that by selecting a gene with high influence as the control gene, we will be able to carry out a more cost-effective intervention. The simulation results presented in the next section show that such an expectation is met.

4. Melanoma Application

In this section, we apply the results of the previous section to a context-sensitive PBN derived from gene expression data collected in a study of metastatic melanoma [48]. In this study, the abundance of mRNA for the gene WNT5A was found to be highly discriminating between cells with properties typically associated with high versus low metastatic competence. These findings were validated and expanded in a second study in which experimentally increasing the levels of the Wnt5a protein secreted by a melanoma cell line via genetic engineering methods directly altered the metastatic competence of that cell as measured by the standard *in vitro* assays for metastasis [51]. Furthermore, it was found that an intervention that blocked the Wnt5a protein from activating its receptor, the use of an antibody that binds the Wnt5a protein, could substantially reduce Wnt5a's ability to induce a metastatic phenotype. This suggests that a reasonable control strategy would be to use an intervention that reduces the WNT5A gene's action in affecting biological regulation, since the available data suggests that disruption of this influence could reduce the chance of a melanoma metastasizing, a desirable outcome. Instantaneously random PBNs derived from the same expression data have been used in [12, 52] for demonstrating earlier intervention strategies.

Here, we consider a 7 gene network containing the genes WNT5A, pirin, S100P, RET1, MART1, HADHB and STC2. To obtain the PBN, we have used the Bayesian connectivity-based approach of [9] to construct four highly probable Boolean networks that are used as the constituent Boolean networks in the PBN, with their selection probabilities based on their Bayesian scores. The four generated Boolean networks are shown in Figs. 11, 12 ,13, and 14, where the states are labeled from 0 to $127 = 2^7 - 1$. Each constituent network is assumed to be derived from steady-state gene-expression data, and the attractor states and the level sets are shown in the figures. Observe that in each of these networks, the state enters an attractor cycle in a small number of steps (at most nine), which is consistent with what is expected in real networks [9].

The control strategy of the previous section has been applied to the designed PBN with pirin chosen as the control gene and $p = q = 0.01$. Figure 15 shows the expected cost for a finite horizon problem of length 5 originating from each of the 128 states. In these simulations, the problem formulation for 2^n states has been used. The cost of control is assumed to be 0.5 and the states are assigned a terminal penalty of 5 if WNT5A is 1 and 0 if WNT5A is 0. The control objective is to down-regulate the WNT5A gene. From Fig. 15, it is clear that the expected cost with control is much lower than that without control, which agrees with our objective. If the length of the control horizon is increased, then Fig. 16 shows that all the initial states start yielding almost the same expected cost. This may be due to the fact that the maximum level of the constituent networks is 9 and the Markov chain is ergodic. If, on the other hand, the $2^n L$ formulation is used, then the expected costs for different initial states become almost equal after a larger number of time steps (data not shown). This is possibly due to the fact that no averaging is used in that formulation.

Next we consider the relationship between the influence of a control gene and its effectiveness in carrying out the intervention. The influences of the other six genes on WNT5a are as follows: pirin $= 1$, S100P $= 0.75$, RET1 $= 0$, MART1 $= 0$, HADHB $= 1$,

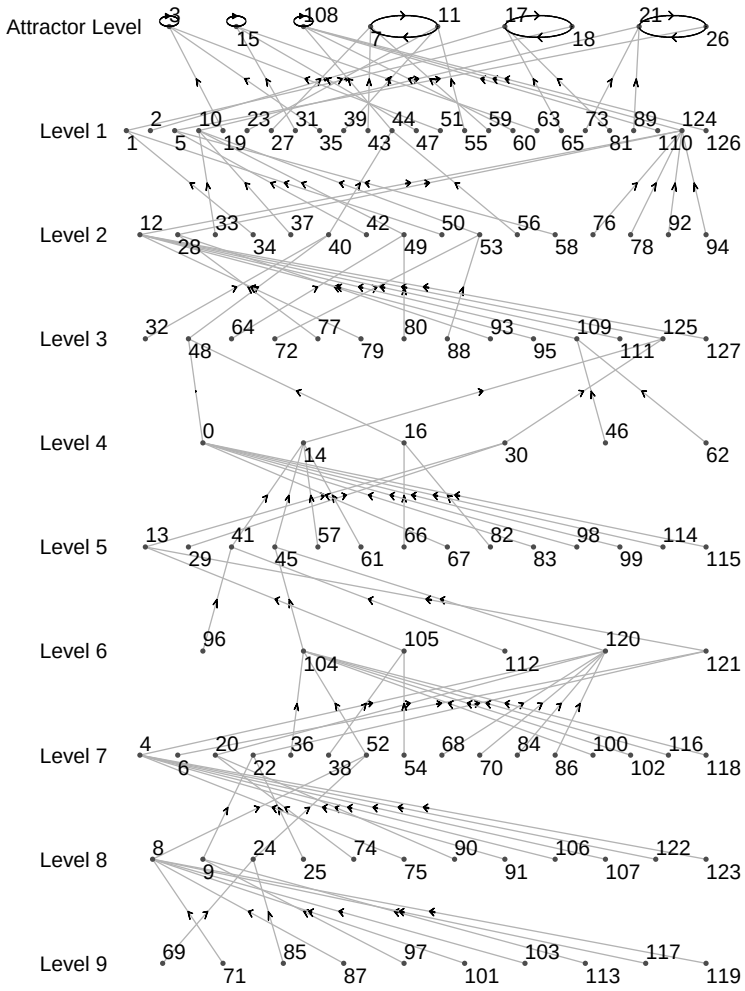


Fig. 11. Network 1 [13].

and STC2 $= 1$. The influence has been calculated from the influences of the genes in the four constituent Boolean networks, assuming equal probabilities for each network. These influence values (GI) are tabulated alongside the control genes (CG) in Table VII. The perturbation probability p is not taken into account for the influence calculations because it has a very low value. If the starting gene activity profile is pirin $= 0$, S100P $= 0$, RET1 $= 0$, MART1 $= 0$, HADHB $= 1$, STC2 $= 0$, and WNT5A $= 1$, then the expected costs for finite horizon control problems of lengths (Ln) 5 and 30 are shown in Table VII. Here,

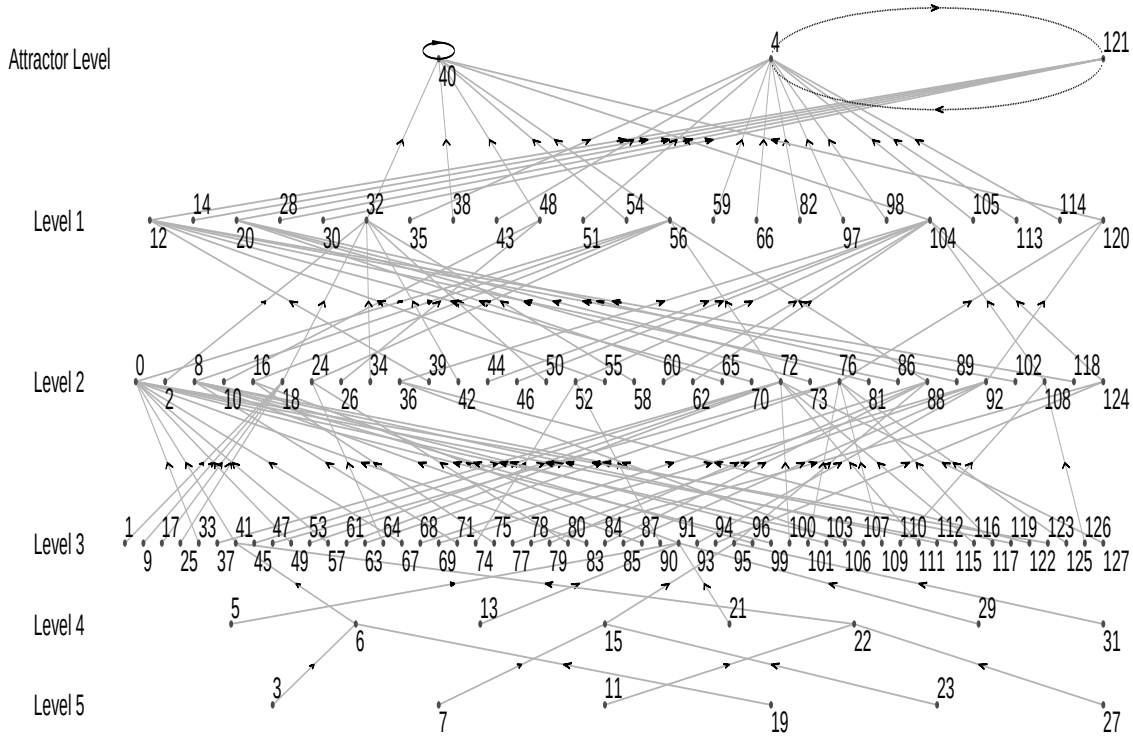


Fig. 12. Network 2 [13].

$Ec1$ represents the expected cost when the 2^n state formulation is used, $Ec2$ represents the expected cost when the $2^n L$ state formulation is used, the suffix wc denotes with control, and the suffix woc denotes without control. The table shows that the expected cost is much lower (0.35 and 0.39) when the high-influence genes pirin and HADHB are used, as compared to the expected cost (0.56) obtained when the low-influence gene MART1 is used to control the network.

B. Infinite Horizon Control for Context-Sensitive PBNs: Problem Formulation

In this section, we formulate and solve the infinite horizon control problem for context-sensitive PBNs. The problem formulation and results summarized in the last section for the finite horizon case serve to motivate the developments here. Consider the finite horizon

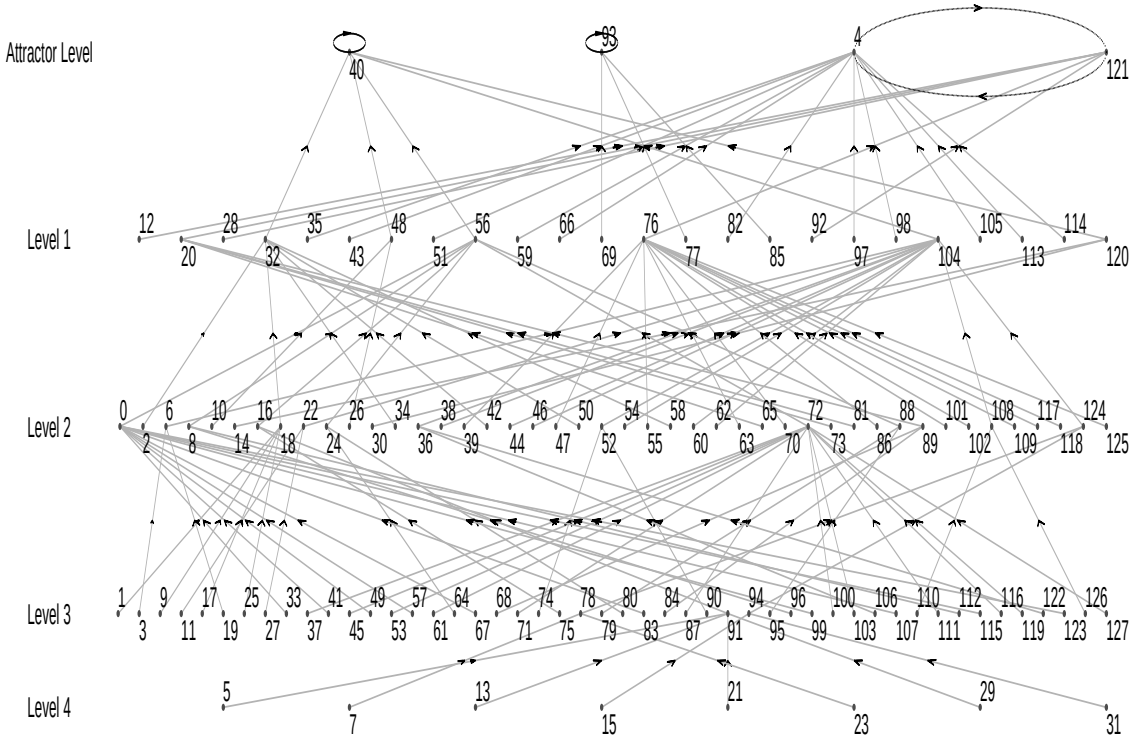


Fig. 13. Network 3 [13].

cost function being minimized in (4.11) and suppose that the control horizon characterized by M is made larger and larger and in the limit we would like for it to tend to infinity. In trying to do so, we immediately encounter a number of potential obstacles that did not arise in the finite horizon case.

First, in the finite horizon case, since there is a terminal state which is being separately penalized, the cost per stage $g_t(z_t, u_t)$ is assumed to only depend on the control applied and the current state. In the infinite horizon problem, the control horizon is infinite and, therefore there is no terminal state or its associated terminal penalty. Consequently, for the infinite horizon case, the cost per stage should depend on the origin i, the destination j and the applied control input u. In other words, $g_t(i, u)$ of the finite horizon problem should now be replaced by $\tilde{g}(i, u, j)$ so that the per stage cost takes into account the origin, the

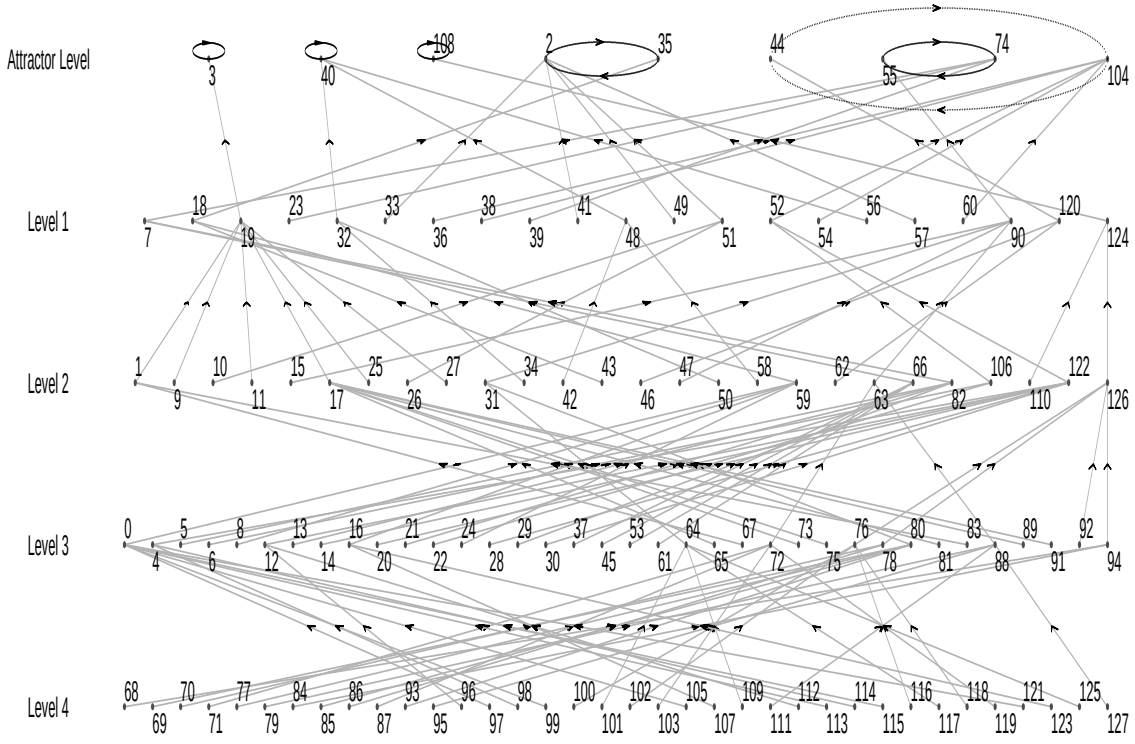


Fig. 14. Network 4 [13].

destination and the control.[1]

Second, in the finite horizon problem, the summation in (4.11) is a finite one and so the quantity being minimized is finite. If we let the control horizon go to infinity, there is a possibility that the summation of the one stage costs may go to infinity (for all controls) leading to an ill-posed optimization problem. To make the optimization problem well posed, the cost considered in (4.11) has to be modified before letting the length M of the control horizon tend to infinity. We will consider two such modifications that have been

[1]Note that while finite horizon control problems in the literature allow for cost-per-stage functions that vary from one stage to another, infinite horizon control problems in the literature have typically been derived assuming that the same cost per stage function is used for all stages. For PBNs (both context-sensitive and otherwise), this is not of any consequence since all of our earlier finite horizon results also used the same cost per stage function for all stages.

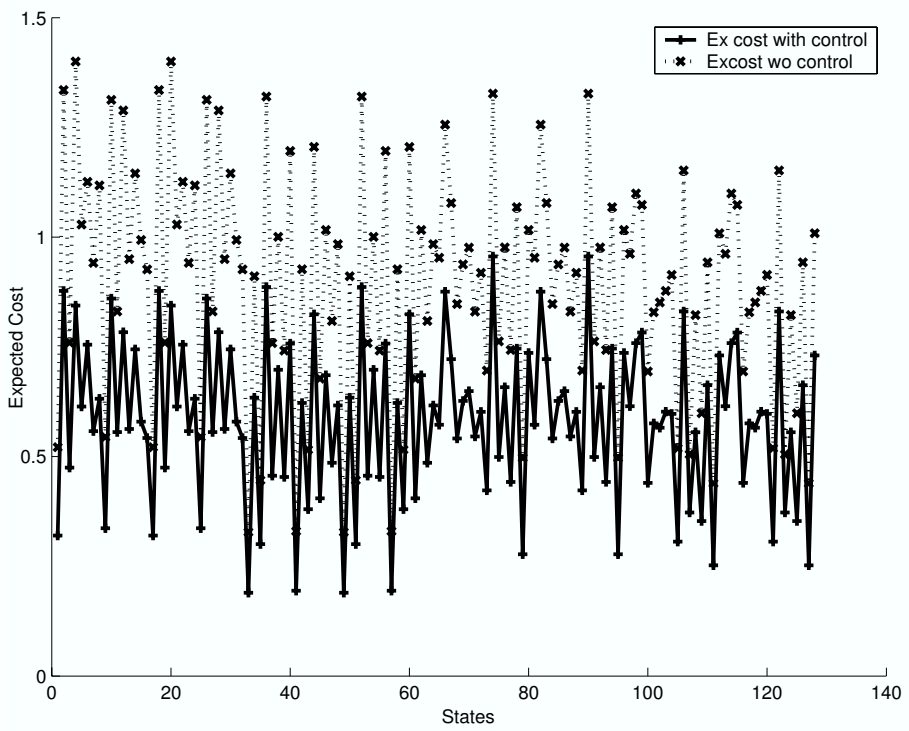


Fig. 15. Expected Cost for a Finite Horizon Problem of Length 5 Originating from the Different Initial States [13].

extensively studied in the literature.

In the first case, we assume that the cost per stage $\tilde{g}(i,u,j)$ is *bounded* $\forall\ i,j \in S$ and $u \in C$ and a *discounting factor* $\alpha \in (0,1)$ is introduced in the cost to make sure that the limit of the finite sums converges as the horizon length goes to infinity. More specifically, our objective is to find a policy $\pi = \{\mu_0, \mu_1.....\}$, where $\mu_t : S \rightarrow C$, $t = 0, 1....$, that minimizes the cost function[2]

$$J_\pi(z_0) = \lim_{M\rightarrow\infty} E\{\sum_{t=0}^{M-1} \alpha^t \tilde{g}(z_t, \mu_t(z_t), w_t)\}, \tag{4.16}$$

[2]Note that a Markov Chain can be modeled by $z_{t+1} = w_t$ [49]. Hence the destination state is the same as the disturbance.

TABLE VII

EXPECTED COST TABLE [13]

CG	GI	Ln	$Ec1wc$	$Ec1woc$	$Ec2wc$	$Ec2woc$
pirin	1	30	.355352	.5784	.566017	.949586
mart1	0	30	.568611	.5784	.743938	.949586
hadhb	1	30	.398291	.5784	.300602	.949586
stc2	1	30	.413105	.5784	.569817	.949586
pirin	1	5	.652455	.974544	.396288	.61994
mart1	0	5	.963684	.974544	.53374	.61994
hadhb	1	5	.762097	.974544	.304567	.61994
stc2	1	5	.830185	.974544	.398155	.61994

where the cost per stage $\tilde{g} : S \times C \times D \rightarrow \Re$ is given. This problem is referred to in the literature as the problem of *minimizing the total cost over an infinite number of stages with discounted and bounded cost per stage*. In the general formulation, the inclusion of α in the cost captures the fact that costs incurred at a later time are less significant. In the case of cancer treatment, $\alpha < 1$ signifies that the condition of the patient in the initial stages of treatment is more important than the condition at a later stage, or in other words, the reward for improving the condition of the patient in the present is more significant than the reward obtained from similar improvement at a later stage. This approach is reasonable if we keep in mind the expected life-span of the patient.

In the second case, one avoids the problem of a possibly infinite total cost by consid-

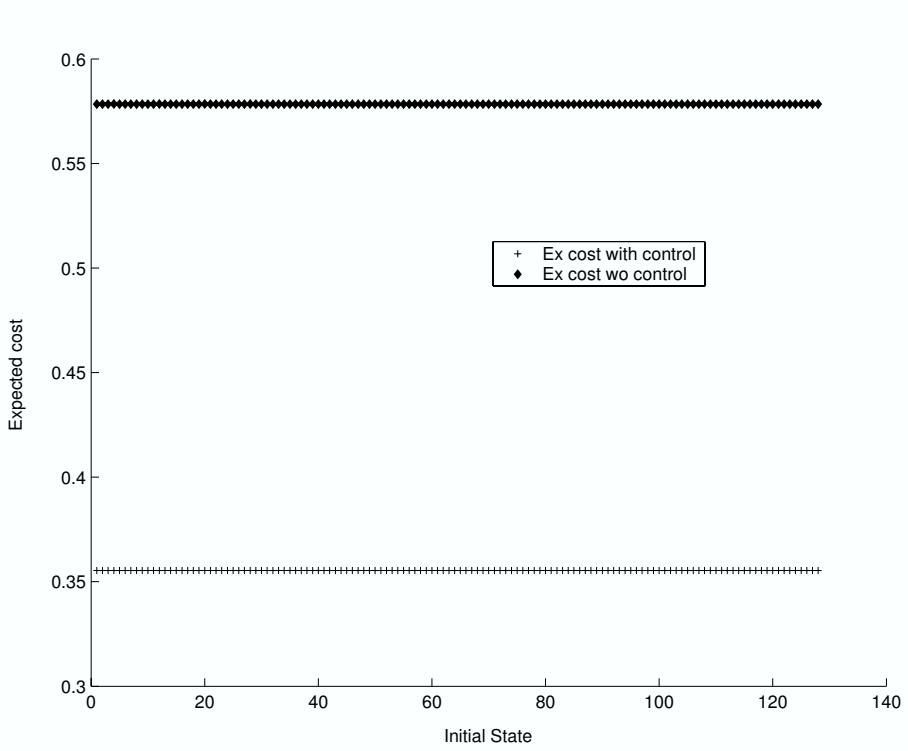


Fig. 16. Expected Cost for a Finite Horizon Problem of Length 30 Originating from the Different Initial States [13].

ering the *average cost per stage* which is defined by

$$J_\pi(z_0) = \lim_{M\to\infty} \frac{1}{M} E\{\sum_{t=0}^{M-1} \tilde{g}(z_t, \mu_t(z_t), w_t)\}. \tag{4.17}$$

In this formulation, a control policy $\pi = \{\mu_0, \mu_1, \cdots\}$ is chosen to minimize the above cost and the problem is referred to as the *average cost per stage problem*. Minimization of the total cost is feasible if $J_\pi(z_0)$ is finite for at least some admissible policies π and some admissible states z_0. If we consider no discounting, i.e. a *discount factor* of 1, and there is no zero-cost absorbing state (which is the case in context-sensitive PBNs with perturbation), then the total cost will frequently go to ∞. Hence the *average cost per stage*

formulation is essential when we are interested in the condition of the patient in the long run and equal importance is given to the patient's condition in all stages.

For reasons already discussed, the cost per stage $\tilde{g}(z_t, u_t, w_t)$ depends on z_t, u_t and w_t. However, since in Eqns (4.16) and (4.17), the cost is obtained only after taking the expectation with respect to the disturbances, it is possible to replace $\tilde{g}(z_t, u_t, w_t)$ by an equivalent cost per stage that does not depend on the disturbance w. This amounts to using the expected cost per stage in all calculations. More specifically, if $\tilde{g}(i, u, j)$ is the cost of using u at state i and moving to state j, we use as cost per stage the expected cost $g(i, u)$ given by [49]:

$$g(i,u) = \sum_{j=0}^{2^n-1} p_{ij}(u)\tilde{g}(i,u,j). \tag{4.18}$$

Now, the cost $\tilde{g}(i, u, j)$ of moving from state i to state j under control u may depend on the starting state i. However, in the case of PBNs, we have no obvious basis for assigning different costs based on different initial states. Accordingly, we assume that the penalty $\tilde{g}(i, u, j)$ is independent of the starting state i and its value is based on the control effort and the terminal state j. The penalty is high if the end state is a bad state regardless of the starting state, and vice-versa. Hence $\tilde{g}(i, u, j) = \tilde{g}(u, j)$ and Eq. (4.18) becomes

$$g(i,u) = \sum_{j=0}^{2^n-1} p_{ij}(u)\tilde{g}(u,j). \tag{4.19}$$

We are now ready to present the solutions to the infinite horizon optimal control problems in the two cases where the performance indices are (i) total cost with discounted and bounded cost per stage; and (ii) average cost per stage. In either case, we denote by Π the set of all *admissible* policies π, i.e., the set of all sequences of functions $\pi = \mu_0, \mu_1, ...$ with $\mu_t(z) : S \to C, t = 0, 1, ...$ The optimal cost function J^* is defined by

$$J^*(z) = \min_{\pi \in \Pi} J_\pi(z), z \in S. \tag{4.20}$$

A *stationary policy* is an admissible policy of the form $\pi = \mu, \mu, ...$, and its corresponding cost function is denoted by J_μ. We say that the stationary policy $\pi = \mu, \mu....$ is optimal if $J_\mu(z) = J^*(z)$ for all states z.

1. Optimal Control Solution: Total Cost with Discounted and Bounded Cost per Stage

In this section, we solve the problem of minimizing the cost (4.16) under the assumption that the cost per stage $\tilde{g}(i, u, w)$ is bounded, i.e. $\exists\ B\ >\ 0$ such that $\tilde{g}$ satisfies $|\tilde{g}(z, u, w)| \leq B$, for all $(z, u, w) \in S \times C \times D$. In the case of context-sensitive PBNs, this assumption holds since the expected cost, $g(i, u)$, for state i is given by Eq. (4.19), $\sum_{j=0}^{2^n-1} p_{ij}(u) = 1$, and $\tilde{g}(u, j)$ is bounded since the control and disturbance spaces are finite.

Observe that if we set $g_M(z_M) = 0\ \forall\ z_M \in S$ and $g_t(z_t, u_t) = \alpha^t g(z_t, u_t)$ in the finite horizon problem of Eq. (4.11) and let $M \rightarrow \infty$, then we obtain the infinite horizon cost function considered in Eq. (4.16). Thus it seems reasonable that the finite horizon solution described by Eqs. (4.12) and (4.13) in the last section could provide a basis for arriving at the solution of the optimization problem (4.20) where J_π is given by Eq. (4.16). A formal derivation of this connection is given in [49]. Here we simply state the result and present an intuitive justification for it.

Towards this end, note that Eq. (4.13) in the dynamic programming algorithm basically describes how the optimal cost J_{t+1} propagates backwards in time to the optimal cost J_t in the finite horizon problem (4.11). For the cost function considered in Eq. (4.16), it is clear that the cost J_{t+1} must be discounted by the factor α while being propagated to the previous stage. Consequently, for the optimal control problem of this section, Eq. (4.13) will have to be replaced by

$$J_t(i) = \min_{u \in C} \left[g(i, u) + \alpha \sum_{j=0}^{2^n-1} p_{ij}(u) J_{t+1}(j) \right]. \tag{4.21}$$

The above equation motivates the introduction of the following two mappings:

For any cost function $J : S \rightarrow \Re$, define the mapping $TJ : S \rightarrow \Re$ by

$$(TJ)(i) = \min_{u \in C}[g(i,u) + \alpha \sum_{j=0}^{2^n-1} p_{ij}(u)J(j)],\ i \in S. \tag{4.22}$$

Note that TJ is the optimal cost function for the one-stage (finite horizon) problem that has stage cost g and terminal cost αJ.

Similarly for any cost function $J : S \rightarrow \Re$ and control function $\mu : S \rightarrow C$, define the mapping $T_\mu J : S \rightarrow \Re$ by

$$(T_\mu J)(i) = g(i,\mu(i)) + \alpha \sum_{j=0}^{2^n-1} p_{ij}(\mu(i))J(j),\ i \in S. \tag{4.23}$$

$T_\mu J$ can be viewed as the cost function associated with the policy μ for the one-stage problem that has stage cost function g and terminal cost αJ. Since the mappings T and T_μ map functions $J : S \rightarrow \Re$ into new functions mapping S to $\Re$, one can define the composition of T with itself and T_μ with itself as follows:

$$(T^k J)(i) = (T(T^{k-1}J))(i), i \in S,\ k = 1, 2, \cdots, \tag{4.24}$$

$$(T^0 J)(i) = J(i),\ i \in S, \tag{4.25}$$

and

$$(T_\mu^k J)(i) = (T_\mu(T_\mu^{k-1}J))(i), i \in S,\ k = 1, 2, \cdots, \tag{4.26}$$

$$(T_\mu^0 J)(i) = J(i),\ i \in S. \tag{4.27}$$

The mappings T and T_μ play an important role in the solution of the optimal control problem of this section. Specifically, it can be shown that (i) the optimal cost function J^* is the unique fixed point of the map T; (ii) the iteration $J_{t+1} = TJ_t$ converges to J^* as $t \rightarrow \infty$; and (iii) the mapping T_μ can be used to characterize the conditions under

which a given stationary policy μ is optimal. These ideas are formalized in the following three theorems adapted from [49]. To make the dissertation self-contained, the proofs are included in the Appendix B.

Theorem B.1. ***Convergence of the discounted-cost algorithm****: For any bounded cost function $J : S \to \Re$, the optimal cost function J^* satisfies*

$$J^*(i) = \lim_{M\to\infty} (T^M J)(i), \textit{ for all } i \in S. \tag{4.28}$$

Theorem B.2. ***Bellman's Equation****: The optimal cost function J^* satisfies*

$$J^*(i) = \min_{u\in C}[g(i,u) + \alpha \sum_{j=0}^{2^n-1} p_{ij}(u)J^*(j))], \textit{ for all } i \in S. \tag{4.29}$$

or, equivalently, $J^* = TJ^*$. Furthermore, J^* is the unique solution of this equation within the class of bounded functions.

Theorem B.3. ***Necessary and Sufficient Condition for Optimality****: A stationary policy μ is optimal if and only if $\mu(i)$ attains the minimum in Bellman's equation (4.29) for each $i \in S$; i.e.,*

$$TJ^* = T_\mu J^* \tag{4.30}$$

The three theorems above provide the basis for coming up with computational algorithms for determining the optimal policy. Theorem B.2 asserts that the optimal cost function satisfies Bellman's equation while Theorem B.1 states that the optimal cost function can be iteratively determined by running the recursion

$$J_{t+1} = TJ_t, \ t = 0, 1, 2, \cdots \tag{4.31}$$

for any bounded initial cost function $J_0 : S \to \Re$. Since this iteration is guaranteed to converge to J^*, one can keep on running this iteration until some stopping criterion is reached. The resulting policy is a stationary one which, by Theorem B.3, must be optimal.

The iteration described in (4.31) above is referred to as the *Value Iteration* procedure since, at every stage we are iterating on the values of the cost function and the optimal policy simply falls out as a by product when the iteration converges to the optimal value of the cost function.

An alternative approach for solving the optimal control problem of this section is referred to as *Policy Iteration.* Before presenting this approach, we introduce the following matrix and vector notations.

$$J = \begin{pmatrix} J(0) \\ . \\ . \\ J(2^n - 1) \end{pmatrix},$$

$$J_\mu = \begin{pmatrix} J_\mu(0) \\ . \\ . \\ J_\mu(2^n - 1) \end{pmatrix},$$

$$TJ = \begin{pmatrix} (TJ)(0) \\ . \\ . \\ (TJ)(2^n - 1) \end{pmatrix},$$

$$T_\mu J = \begin{pmatrix} (T_\mu J)(0) \\ . \\ . \\ (T_\mu J)(2^n - 1) \end{pmatrix}.$$

The transition probability matrix corresponding to the stationary policy μ is represented as

$$P_\mu = \begin{pmatrix} p_{00}(\mu(0)) & ... & p_{0,2^n-1}(\mu(0)) \\ . & . & . \\ . & . & . \\ p_{2^n-1,0}(\mu(2^n-1)) & ... & p_{2^n-1,2^n-1}(\mu(2^n-1)) \end{pmatrix}$$

and g_μ represents the cost vector

$$g_\mu = \begin{pmatrix} g(0,\mu(0)) \\ . \\ . \\ g(2^n-1,\mu(2^n-1)) \end{pmatrix}.$$

Using the above notation, it is clear that for any stationary policy μ, (4.23) can be rewritten as

$$T_\mu J = g_\mu + \alpha P_\mu J.$$

Furthermore, it can be shown (reasoning similar to proof of Theorem B.2 given in the Appendix B) that the cost J_μ corresponding to the policy μ satisfies

$$J_\mu = g_\mu + \alpha P_\mu J_\mu$$

or

$$[I - \alpha P_\mu] J_\mu = g_\mu. \tag{4.32}$$

Equation (4.32) above is a system of linear equations that can be solved to calculate the cost J_μ corresponding to a given stationary policy μ. In the policy iteration algorithm, one starts with a given stationary policy, evaluates the corresponding cost using (4.32) and tries to find a policy that yields a smaller cost. The process is terminated when we arrive at a fixed point of the mapping T. We next formally present the steps involved in the policy iteration algorithm.

Step 1: (Initialization) An initial policy μ^0 is selected.

Step 2: (Policy Evaluation) Given a stationary policy μ^k, we compute the corresponding cost function J_{μ^k} from the linear system of equations

$$(I - \alpha P_{\mu^k}) J_{\mu^k} = g_{\mu^k}. \tag{4.33}$$

P_{μ^k} is the probability transition matrix obtained using control policy μ^k.

Step 3: (Policy Improvement) An improved (in terms of the cost J) stationary policy μ^{k+1} satisfying $T_{\mu^{k+1}} J_{\mu^k} = T J_{\mu^k}$ is obtained.

The iterations are stopped if $J_{\mu^k} = T J_{\mu^k}$, else we return to Step 2 and repeat the process.

2. Optimal Control Solution: Average Cost per Stage

In this section, we solve the problem of minimizing the cost (4.17). In this case, there is no discounting i.e. $\alpha = 1$ and we are interested in determining the policy that minimizes the limit of $\frac{J_M}{M}$ as $M \to \infty$ where J_M is the optimal finite horizon cost over an interval of length M. The same reasoning used in the last section can be used to derive the counterparts of Eqns. (4.21),(4.22) and (4.23) which for this case become

$$J_t(i) = \min_{u \in C} [g(i,u) + \sum_{j=0}^{2^n-1} p_{ij}(u) J_{t+1}(j)], i \in S. \tag{4.34}$$

$$(TJ)(i) = \min_{u \in C} [g(i,u) + \sum_{j=0}^{2^n-1} p_{ij}(u) J(j)], i \in S. \tag{4.35}$$

$$(T_\mu J)(i) = g(i,\mu(i)) + \sum_{j=0}^{2^n-1} p_{ij}(\mu(i)) J(j), i \in S. \tag{4.36}$$

However in this case, the value iteration

$$J_{t+1}(i) = TJ_t(i) \tag{4.37}$$

considered in the last section cannot be directly used since, in the absence of the discounting factor, it may diverge to infinity. Thus calculating the average cost by taking $\lim_{M\to\infty} \frac{J_M}{M}$ is not feasible.

Instead we consider a *differential cost* h_t which is obtained by subtracting a fixed component of J_t say $J_t(n_1)$ from each element of J_t i.e.

$$h_t(i) = J_t(i) - J_t(n_1)\ \forall i \in S. \tag{4.38}$$

Clearly $h_t(n_1) = 0$. Also defining $e = [1, 1, ..., 1]^T$, the above relationship can be rewritten as

$$h_t = J_t - J_t(n_1)e \tag{4.39}$$

Similarly

$$h_{t+1} = J_{t+1} - J_{t+1}(n_1)e \tag{4.40}$$

Now substituting for J_t and J_{t+1} into Eq. (4.37), we have

$$\begin{aligned} h_{t+1} + J_{t+1}(n_1)e &= T(h_t + J_t(n_1)e) && (4.41)\\ \implies h_{t+1} + J_{t+1}(n_1)e &= Th_t + J_t(n_1)e \\ \implies h_{t+1} &= Th_t - (J_{t+1}(n_1)e - J_t(n_1)e) && (4.42) \end{aligned}$$

From Eq. (4.37),

$$\begin{aligned} J_{t+1}(n_1) &= T(J_t)(n_1) \\ &= T(h_t + J_t(n_1)e)(n_1) \text{ from Eq.}(4.39) \\ &= Th_t(n_1) + J_t(n_1). \end{aligned} \tag{4.43}$$

Hence, it follows that

$$J_{t+1}(n_1)e - J_t(n_1)e = Th_t(n_1)e$$

so that Eq. (4.42) yields

$$h_{t+1} = Th_t - (Th_t)(n_1)e \tag{4.44}$$

as the *value iteration algorithm* for the differential cost.

We next state two theorems adapted from [49] which form the basis for the solution to the average cost per stage optimal control problem. The proofs are given in Appendix B. The first theorem formalizes something which appears to be intuitively reasonable – since the *average* optimal cost is calculated over an *infinite* horizon, its value should be independent of the starting state. Note also that the required assumption about the ergodicity of the Markov Chain is satisfied by the context-sensitive PBNs considered in this chapter.

Theorem B.4. *For ergodic Markov Chains, the optimal average cost per stage is independent of the initial state.*

The next theorem formalizes the fact that if the value iteration (4.44) for the differential cost converges to some vector h, i.e. $(Th)(n_1)e + h = Th$ then $Th(n_1)$ is the optimal average cost per stage (which is the same for all initial states).

Theorem B.5. *If a scalar λ and a 2^n-dimensional vector h satisfy*

$$\lambda + h(i) = \min_{u \in C}[g(i,u) + \sum_{j=0}^{2^n-1} p_{ij}(u)h(j)],\ i \in S, \tag{4.45}$$

or equivalently, $\lambda e + h = Th$, where e is the unitary vector $[1111...1]^T$ and h = $[h(0), h(1)$ .... $h(2^n-1)]^T$, then λ is the optimal average cost per stage $J^*(i)$ for all i, i.e.

$$\lambda = \min_{\pi} J_\pi(i) = J^*(i), i \in S, \tag{4.46}$$

Furthermore, if $\mu^*(i)$ attains the minimum in Eq. (4.45) for each i, then the stationary policy μ^* is optimal, i.e., $J_{\mu^*}(i) = \lambda$ for all $i \in S$.

We note that for the average cost per stage problem, Eq. (4.45) plays the same role as Bellman's Equation (4.29) in the solution of the problem of the last section. Consequently, we can immediately arrive at the following *policy iteration* algorithm for this case :

Step 1: (Initialization) An initial policy μ^0 is selected.

Step 2: (Policy Evaluation) Given a stationary policy μ^k, we obtain the corresponding average and differential costs λ^k and $h^k(i)$ satisfying

$$\lambda^k + h^k(i) = g(i, \mu^k(i)) + \sum_{j=0}^{2^n-1} p_{ij}(\mu^k(i))h^k(j), i \in S. \tag{4.47}$$

This linear system of equations can be solved utilizing the fact that $h^k(n_1) = 0$, where $n_1 \in S$ is any particular reference state.

Step 3: (Policy Improvement) An improved stationary policy μ^{k+1} satisfying

$$g(i, \mu^{k+1}(i)) + \sum_{j=0}^{2^n-1} p_{ij}(\mu^{k+1}(i))h^k(j) = \min_{u \in C}[g(i,u) + \sum_{j=0}^{2^n-1} p_{ij}(u)h^k(j)], \tag{4.48}$$

or equivalently, $T_{\mu^{k+1}}h^k = Th^k$, is obtained.

The iterations are stopped if $\mu^{k+1} = \mu^k$, else we return to Step 2 and repeat the process.

3. Melanoma Application

In this section, we apply the results of the previous section to a context-sensitive PBN derived from the same gene expression data used earler. Finite Horizon Control on instantaneously random and context-sensitive PBNs derived from this expression data have been used in [12, 52, 13](see also Section IV.A.4 in this dissertation) for demonstrating earlier intervention strategies.

We consider a 7 gene network containing the genes WNT5A, pirin, S100P, RET1, MART1, HADHB and STC2. To obtain the PBN, we have used the algorithms described in [8] to construct four Boolean networks to use as the constituent Boolean networks in the PBN. Each constituent network is assumed to be derived from steady-state gene-expression data (a common assumption – see [8]). The states are ordered as WNT5A, pirin, S100P, RET1, MART1, HADHB and STC2, with WNT5A as the most significant bit (MSB) and STC2 as the least significant bit (LSB).

The control strategies of the previous sections have been applied to the designed PBN with pirin chosen as the control gene ($u = 1$ signifying the state of pirin is reversed and $u = 0$ signifying no intervention) and $p = q = 0.01$.

The cost of control is assumed to be 1 and the states are assigned penalties as follows:

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

The penalty assignment is based on the fact that for infinite-horizon problems, there is no terminal penalty; instead, the cost per stage g contains the penalties of each state. Since our objective is to down-regulate the WNT5A gene, a higher penalty is assigned for destination states having WNT5a up-regulated. Also for a given WNT5A status for the destination

state, a higher penalty is assigned when the control is active versus when it is not.

a. Discounted Cost Problem

Fig. 17 shows the Total Cost for the discounted cost function with bounded cost per stage originating from each of the 128 states after the iterations have converged, with the discount factor α chosen to be 0.9.

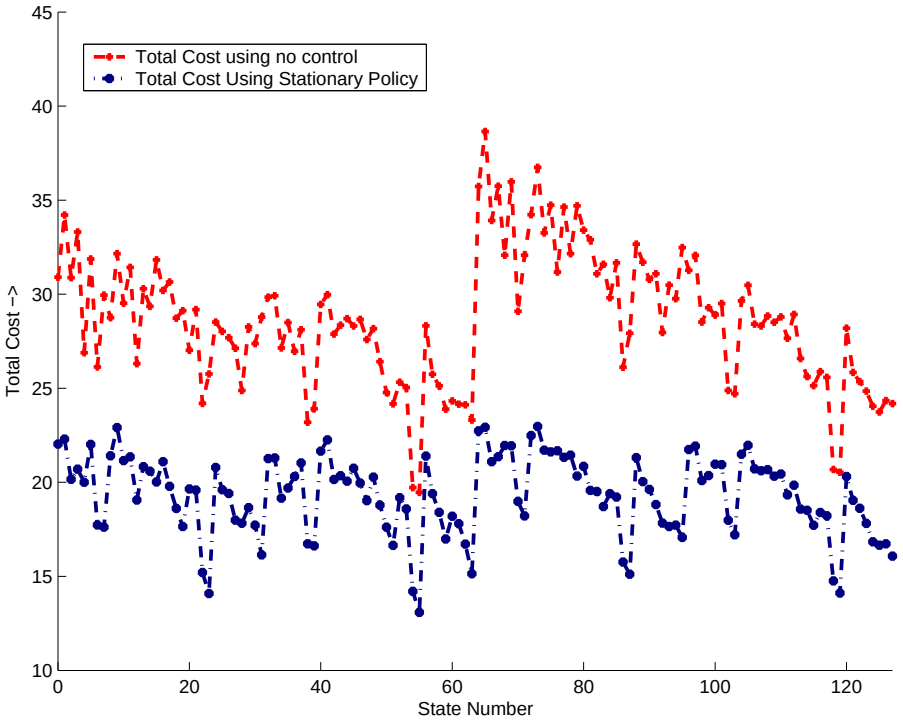


Fig. 17. Total Cost Originating from the Different Initial States [53].

The control objective is to down-regulate the WNT5A gene. From Fig. 17, it is clear that the Total Cost with an optimal stationary policy is much lower than that without control, which agrees with our objective. Fig. 18 shows the stationary policy obtained from the solution of the discounted cost problem. Fig. 19 shows the average total cost per state for each iteration. The stationary policy has been obtained using value iteration and policy iteration. The starting policy for the policy iterations was $\mu = 0, 0, 0....$, i.e. no control, and

hence the initial cost for the policy iteration is the same as the eventual total uncontrolled cost (Fig. 19). We should note that the policy iteration provides us the optimal policy in a small number of steps as compared to value iteration. Moreover, as the collection of stationary policies is finite (in this particular case, it is 2^{128}), the policy iteration is bound to give us an optimal stationary policy in a finite number of steps, whereas value iteration may converge in an infinite number of steps. On the other hand, the problem with policy iteration is solving the system of linear equations $(I - \alpha P_{\mu^k})J_{\mu^k} = g_{\mu^k}$, which becomes very complicated as the number of states increases.

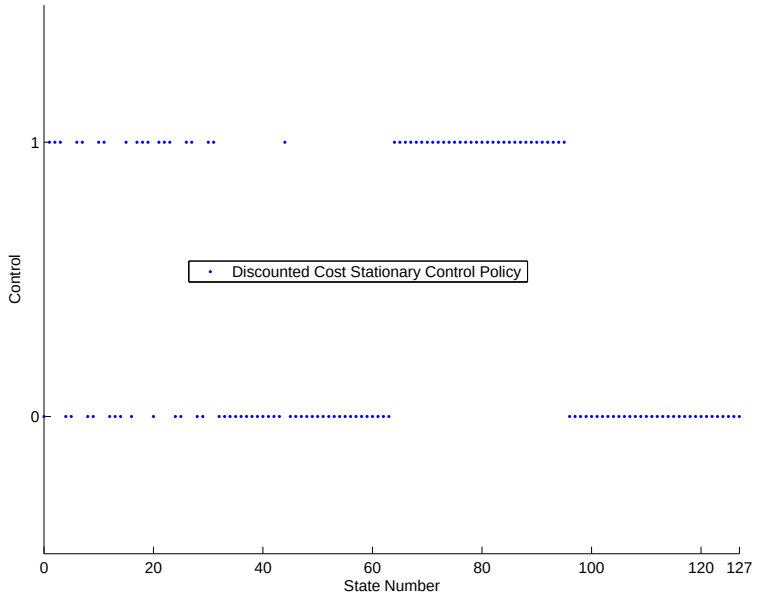


Fig. 18. Stationary Policy Obtained Using Discounted Cost Formulation [53].

Fig. 20 shows the steady-state distributions of the PBN using the obtained stationary policy (Fig. 18) and Fig. 21 shows the original PBN steady state for comparison. We should note that the states from 0 to 63 have WNT5A 0 and hence are desirable states, as compared to states 64 to 127 that have WNT5A 1 and hence are undesirable. The steady-state distribution Figures 20 and 21 show that the stationary policy has enabled us to shift the probability mass from the bad states to states with lower metastatic competence. For ex-

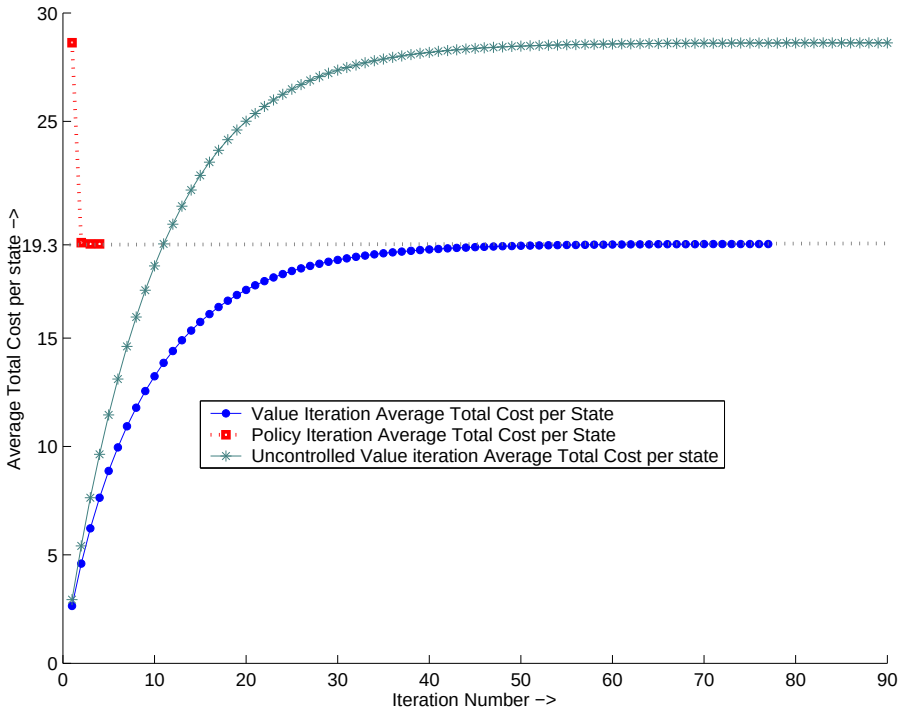


Fig. 19. Average Cost per State Using Discounted Total Cost [53].

ample, state 65 (WNT5A is 1) has a high probability mass (0.15) in the original steady state but stationary control has reduced its steady-state mass to 0.04. Similarly, the probability mass of state 63 (desirable state) is high when using the stationary policy. To numerically quantify the change, we multiply the stationary distribution with the cost vector. For the original PBN the cost vector is 0 for states 0 to 63 and 5 for states 64 to 127. For the stationary policy the cost vector is $\tilde{g}(\mu(z), z),\ z \in [0, 1, 2, ...127]$. The value for the stationary policy using discounted cost formulation is 1.7465 as compared to 2.9830 for no control.

b. Average Cost per Stage Problem

In this section we use the average-cost-per-stage formulation to design our optimal stationary policy. Both the value iteration and policy iteration algorithms are used to calculate the optimal policy. The optimal policy obtained using the two iteration methods are the same.

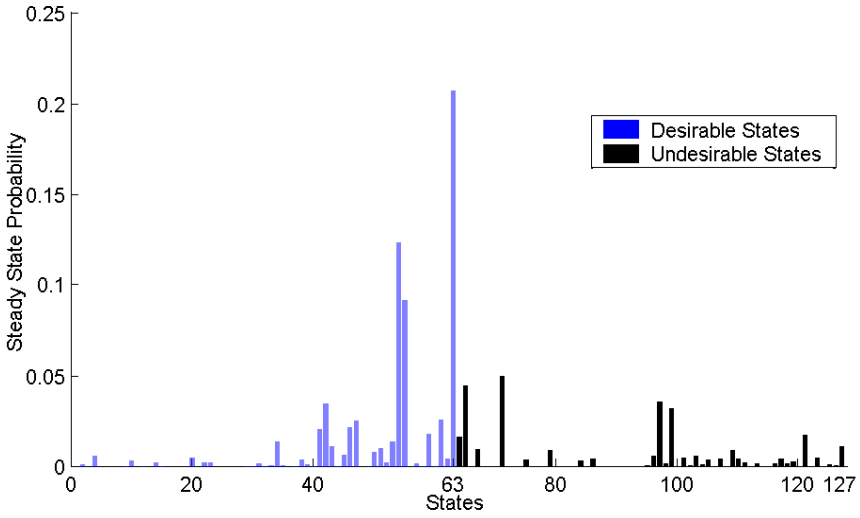


Fig. 20. Steady State Using Discounted Cost Stationary Policy [53].

This policy is shown in Fig. 22. The average cost, λ, for the optimal policy of Fig. 22 is 1.746302, whereas the average cost for the uncontrolled policy is 2.9829707. We have used the same cost of control and penalties as used for the discounted cost simulations. The evolution of λ with each iteration for the two methods are shown in Fig. 23. The steady-state distribution is shown in Fig. 24 and is very similar to the steady-state distribution obtained using the previous total-cost formulation. Comparison of Figs. 21 and 24 indicates that application of the stationary policy has been successful in shifting the steady-state distribution from undesirable to desirable states. The numerical value for the multiplication of the steady-state distribution with the cost vector is 1.7463 for the stationary policy, whereas for the uncontrolled PBN it is 2.9830.

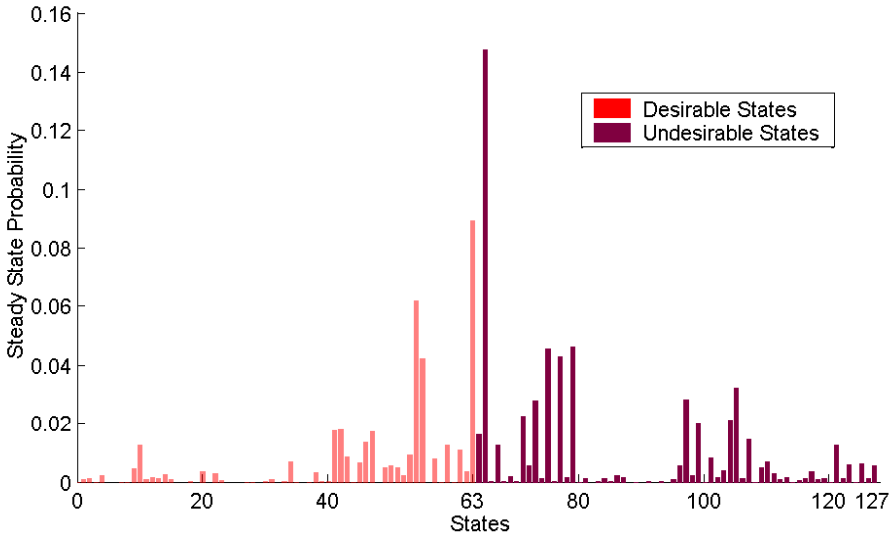


Fig. 21. Original Steady State [53].

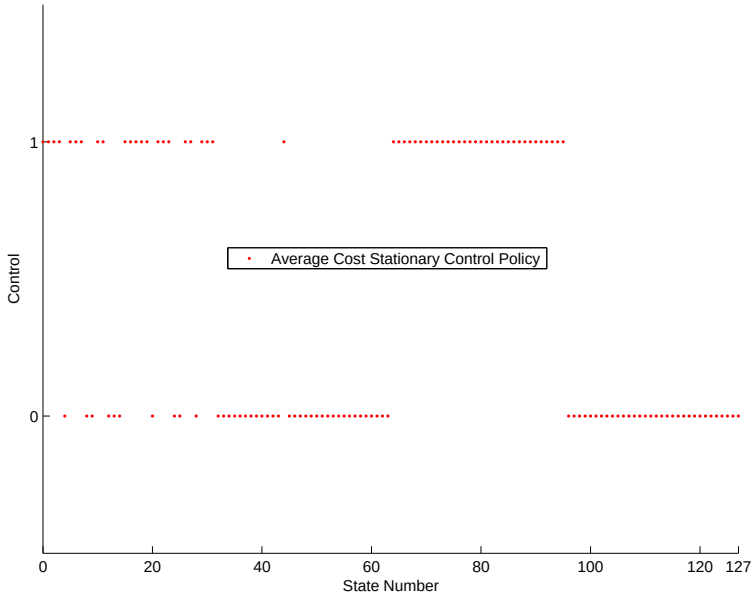


Fig. 22. Stationary Policy Obtained Using Average Cost Formulation [53].

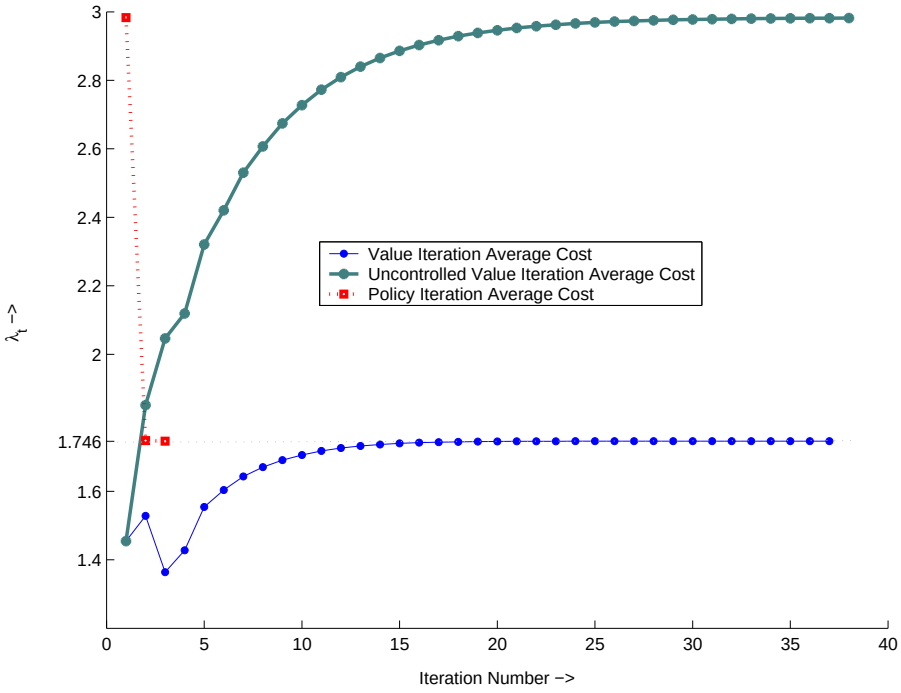


Fig. 23. Evolution of λ with Each Iteration [53].

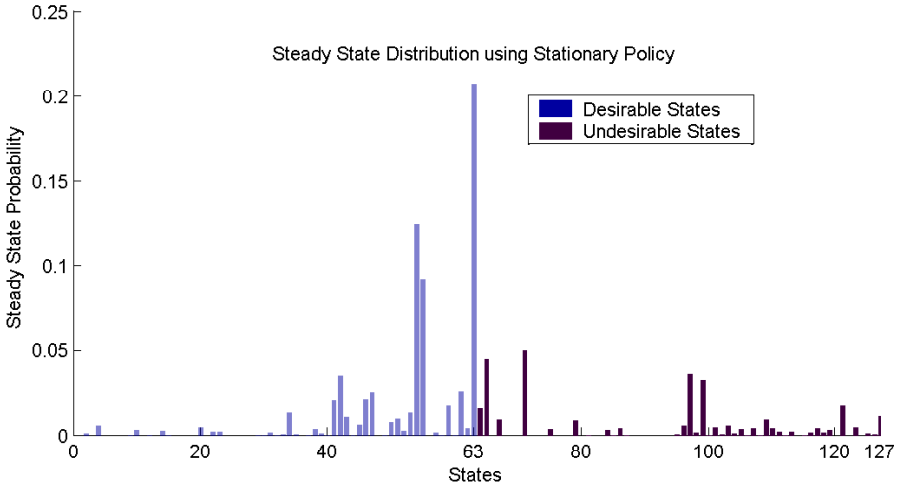


Fig. 24. Steady State Using Average Cost Stationary Policy [53].

CHAPTER V

ROBUSTNESS OF INTERVENTION STRATEGIES

If we revisit Fig. 1, it is clear that errors made during data extraction, discretization, gene selection and network generation will all propagate downstream and impact the actual success of the designed intervention strategy. Indeed, if the designed intervention approach is to have any hope of succeeding in practice, its outcome must possess some degree of "robustness" or insensitivity to the errors that will invariably propagate down to the intervention design stage from steps further upstream. The study of the effect on intervention outcome of the errors propagating from the different upstream steps is an important open problem in translational genomics. In this chapter, we focus on a special subproblem where it is assumed that the combined effect of the errors propagating from the different stages manifests itself as uncertainty in the transition probabilities of the network, and the robustness of the intervention strategies is to be studied with respect to this uncertainty. With respect to Fig. 1, this corresponds to determining how the uncertainties in Step C impact the outcome of the intervention strategy designed in Step D. Besides error propagation, uncertainties arise due to the inverse problem of system identification being an ill-posed problem.

The intervention approach proposed in Chapter IV Section B is of particular relevance for translational genomics since it seeks to shift the steady-state mass of the PBN from undesirable states to desirable ones. Since it is believed that the steady-state behaviour of a PBN is indicative of the phenotype [3], it is likely that alterations in the steady-state behaviour of the PBN would translate into changes at the phenotypic level. Moreover, the scheme of Section B makes use of stationary policies which are more easily impementable. In Chapter IV Section $B.3$, we used gene expression data from melanoma studies to demon-

strate the feasibility of altering the steady-state distribution of a PBN in desirable ways. However, such alteration in steady-state behaviour was achieved under the assumption that the transition probability matrix of the PBN is known. Such an assumption will not be satisfied for reasons that we have already articulated. Instead, while the intervention strategy would have to be designed based on an estimated network with a transition probability matrix P, in practice it would be applied to the actual network whose transition probability $\tilde{P}$ differs from P. The goal of this chapter is to examine how, for a given intervention policy, the mismatch between P and $\tilde{P}$ affects the steady-state distribution of the *controlled* PBN.

A. Perturbations for the Steady-State Distribution of a Controlled PBN

Before trying to derive any perturbation bounds, let us make a simple observation concerning the probability transition matrix of a controlled PBN. Note that the only kind of interventions that have been proposed to date in the literature [11, 10, 12, 52, 13, 54, 53] are restricted to flipping the expression status of one or more control genes. For such intervention strategies, it is always possible to relate the transition probability matrices of the controlled and uncontrolled PBNs, via a linear transformation, as we explain next.

Let P denote the estimated probability transition matrix corresponding to the PBN of interest and suppose this PBN has m binary control inputs $a_1, a_2 \cdots a_m$ where a_i refers to the status of the ith control gene with $a_i = 1$ signifying that the ith control gene is to be flipped. If we apply a stationary policy, i.e. a policy dependent only on the current state and not on the time, to the Markov Chain P, the rows of the controlled transition probability matrix P_c will be a collection of selected rows from P. This is due to the fact that the flipping of genes actually forces the Markov Chain to start from another initial state. To clearly understand this, let us look at a concrete example.

Example A.1. *Suppose we have a network with* 7 *genes, three of which, namely genes*

1, 2 *and* 3 *are control genes. This means that* $m = 3$ *here. Suppose that the stationary policy for state* 0000001 *(corresponding to the decimal number* 1*) is* 101*, i.e. flip gene 1, leave gene 2 as is, and flip gene 3. This implies that if we are currently at state* 0000001*, application of the stationary policy will reinitialize the state to* 1010001 *(corresponding to the decimal number* 81*). Therefore, in the controlled transition probability matrix* P_c*, the transition probabilities of going from state* 1 *to each of the other states will be the same as the transition probabilities of going from state* 81 *to each of those states in the original uncontrolled network with transition probability matrix* P*.*

From the above example, it is clear that when the class of allowed interventions is restricted to the flipping of genes, the application of a stationary policy converts the uncontrolled transition probability matrix P to a controlled transition probability matrix P_c where P_c and P are related by $P_c = TP$ and T represents a matrix which has only one non-zero entry of 1 in each row. If the stationary policy is of no control, then clearly $T = I$, the identity matrix.

Let π and π_c denote the stationary distribution vectors corresponding to the transition matrices P and P_c respectively. Since the probability transition matrix P has been estimated from data, there can be some errors in estimation. Let $\tilde{P}$ denote the actual transition matrix of the genetic network and let $\tilde{P}_c$ denote the controlled transition probability matrix that results from the application of the stationary policy T on $\tilde{P}$. Let $\tilde{\pi}$ and $\tilde{\pi_c}$ denote the stationary distributions of $\tilde{P}$ and $\tilde{P}_c$ respectively. Our goal is to study the change $\tilde{\pi}_c - \pi_c$ based on the knowledge of P and some characterization of the estimation error $E \triangleq P - \tilde{P}$. Let us summarize the notation and relationships introduced so far:

(i) $\pi P = \pi$

(ii) $\tilde{\pi}\tilde{P} = \tilde{\pi}$

(iii) $\pi_c P_c = \pi_c$

(iv) $\tilde{\pi}_c \tilde{P}_c = \tilde{\pi}_c$

(v) $E = P - \tilde{P}$

(vi) $E_c \stackrel{\Delta}{=} P_c - \tilde{P}_c$.

For two Markov Chains with transition probabilities P and $\tilde{P}$ and sharing a common state space, the difference between the two stationary distributions can be bounded by $|\pi - \tilde{\pi}|_q \leq K\|E\|_\infty$ where $q = 1$ or ∞ and $K > 0$ are some constants and $|\pi - \tilde{\pi}|_q$ refers to the qth norm of the vector $\pi - \tilde{\pi}$ and $\|E\|_\infty$ denotes the ∞ norm of the error matrix E which is equivalent to the maximum row sum of E. The constants K are usually referred to as *condition numbers* and several of them have been studied in the literature. Obviously, some of the condition numbers will yield tighter bounds than the others and [55] gives a nice comparison of the available bounds. Initial studies of steady-state distributions of PBNs using condition numbers were carried out in [10] but steady-state distributions under control were not considered in that reference. Here, we will prove a theorem for a particular condition number studied by Seneta [56]. For a given transition probability matrix P, this condition number called the *ergodicity coefficient* $\tau_1(P)$ is defined by

$$\tau_1(P) = \sup_{\substack{|x^T|_1=1 \\ x^T \mathbf{1}_n = 0}} |x^T P|_1 \tag{5.1}$$

where $\mathbf{1}_n$ denotes the n-dimensional column vector having all entries equal to one. As shown in Appendix C, equivalent definitions are

$$\tau_1(P) = \frac{1}{2} \max_{i,j} \sum_{s=1}^{n} |p_{is} - p_{js}| \tag{5.2}$$

$$\text{and } \tau_1(P) = 1 - \min_{i,j} \sum_{s=1}^{n} \min(p_{is}, p_{js}) \tag{5.3}$$

where p_{ij} refers to the ith row and jth column entry of matrix P. These two definitions are more useful for the purpose of computational evaluation. In [56], the ergodicity coef-

ficient was used to obtain a bound on the perturbation in the steady-state distribution due to perturbations in the transition probability matrix. More specifically, it was shown that if $\tau_1(P) \neq 1$, then

$$|\pi - \tilde{\pi}|_1 \leq \frac{1}{1-\tau_1(P)} \|E\|_\infty. \tag{5.4}$$

Here, we will use the above result to obtain an analytical bound on the perturbations in the *controlled* steady-state distributions that could result from perturbations in the uncontrolled probability transition matrix.

1. Analytical Result Involving the Ergodicity Coefficient

Theorem A.2. *Let P and $\tilde{P}$ be two compatible probability transition matrices with $\tau_1(P) \neq 1$. Then*

$$|\pi_c - \tilde{\pi}_c|_1 \leq \frac{1}{1-\tau_1(P)} \|E\|_\infty. \tag{5.5}$$

Proof. The proof is accomplished by showing

(i) if $\tau_1(P) \neq 1$ then $\frac{1}{1-\tau_1(P_c)} \leq \frac{1}{1-\tau_1(P)}$ i.e. $\tau_1(P_c) \leq \tau_1(P)$; and

(ii) $\|E_c\|_\infty \leq \|E\|_\infty$.

From our earlier discussion, for the class of interventions that have been used for PBNs, we can write $P_c = TP$ where T is a stochastic matrix with each row containing only a single non-zero entry of 1. According to [57],

$$\tau_1(P_1 P_2) \leq \tau_1(P_1)\tau_1(P_2). \tag{5.6}$$

Thus, in our case $\tau_1(P_c) \leq \tau_1(T)\tau_1(P)$. From Eq. 5.3, it is clear that ergodicity coefficient of a stochastic matrix is less than or equal to 1 and hence

$$\tau_1(P_c) \leq \tau_1(P). \tag{5.7}$$

Thus from Eq. 5.4, it follows that

$$|\pi_c - \tilde{\pi}_c|_1 \leq \frac{1}{1 - \tau_1(P)} \|E_c\|_\infty. \tag{5.8}$$

To prove the second part, we consider

$$\begin{aligned} E_c &= P_c - \tilde{P}_c && (5.9)\\ &= TP - T\tilde{P} && (5.10)\\ &= T(P - \tilde{P}) && (5.11)\\ &= TE && (5.12) \end{aligned}$$

In view of Eq. 5.12, it follows that the rows of E_c are selected from the rows of E and hence $\|E_c\|_\infty$ (maximum absolute row sum of E_c) $\leq \|E\|_\infty$. Thus, from Eq. 5.8, it follows that Eq. 5.5 holds, and this completes the proof. □

There are other available perturbation bounds in the literature and some of them are tighter than the ergodicity coefficient bound. The reason for emphasizing the ergodicity coefficient bound here is that the kind of analytical result proved in the above theorem can be derived only for this bound. We will show with the help of simulations that the most effective perturbation bound (to be defined shortly) for the steady-state distribution of the controlled probability transition matrix can sometimes be greater than the corresponding perturbation bound for the steady-state distribution of the original uncontrolled probability transition matrix. The inequality in Eq. 5.7 implies that if the Markov Chain corresponding to an uncontrolled genetic network has a small ergodicity coefficient bound, then the corresponding controlled Markov Chain will also have an ergodicity coefficient that is bounded by the same bound. Consequently, if a stationary policy is designed from an estimated Markov Chain that is "close" to the actual one for the network, then this policy when applied to the actual network will produce results that are close to the desired outcome, as far

as the steady-state behaviour is concerned.

As already mentioned, there are several condition numbers other than the ergodicity coefficient that can be found in the literature. These perturbation bounds are mostly stated in terms of the *fundamental matrix* or the *group inverse* of $A := I - P$. The fundamental matrix of the Markov Chain with transition probability matrix P is defined by

$$Z = (A + e\pi^T)^{-1} \tag{5.13}$$

where $e = [1\ 1\ 1\ \cdots 1]^T$. The group inverse of A is the unique square matrix $A^{\#}$ satisfying the relationships

$$AA^{\#}A = A, A^{\#}AA^{\#} = A^{\#}, \text{ and } AA^{\#} = A^{\#}A. \tag{5.14}$$

The currently available condition numbers for bounding the 1 and ∞ norms of the perturbations in the steady-state distributions are [55]:

$$k_1 = \quad ||Z||_\infty \quad q = 1 \tag{5.15}$$

$$k_2 = \quad ||A^{\#}||_\infty \quad q = 1 \tag{5.16}$$

$$k_3 = \quad \frac{max_j(a^{\#}_{jj} - \min_i a^{\#}_{ij})}{2} \quad q = \infty \tag{5.17}$$

$$k_4 = \quad \max_{i,j} |a^{\#}_{ij}| \quad q = \infty \tag{5.18}$$

$$k_5 = \quad \frac{1}{1-\tau_1(P)} \quad q = 1 \tag{5.19}$$

$$k_6 = \quad \tau_1(A^{\#}) = \tau_1(Z) \quad q = 1 \tag{5.20}$$

$$k_7 = \quad \frac{\min_j ||A^{-1}_{(j)}||_\infty}{2} \quad q = \infty. \tag{5.21}$$

Here the bound k_5 involves the ergodicity coefficient.

2. Simulation Studies for Different Perturbation Bounds

Some of the large transition matrices encountered in genomics tend to be sparse and the perturbation bound k_5 based on the ergodicity coefficient is not very sharp for them. Accordingly, we will first report some simulation results for perturbation bounds using smaller networks of 4 genes (i.e. networks having $2^4 = 16$ states). For generating the networks for these simulations we have used the data from melanoma cell lines which were previously used in several papers e.g. [12, 52, 13, 53]. The 7 gene networks considered in those references were reduced to 4 gene networks using the reduction mapping algorithm given in [58].

For the simulations we generated a number of PBNs consisting of 4 genes and calculated their perturbation bounds. The PBNs were then operated upon by a random stationary policy matrix T and the new perturbation bounds were calculated. In all the cases, the perturbation bound k_5 was found to be smaller for the controlled transition matrices as compared to that for the original uncontrolled transition matrix. In Figure 25, we show the ergodicity coefficient perturbation bounds (k_5) of the original uncontrolled PBNs as gray bars for 10 different generated PBNs. The blue stars represent the ergodicity coefficient perturbation bounds for TP where T is a randomly generated stationary policy matrix.

As shown in [55], the perturbation bound k_3 is one of the tighest bounds. Figure 26 shows the perturbation bound k_3 for different simulations. From Figure 26, it is clear that in this case, the perturbation bound for some randomly generated stationary policy matrices (TP) can exceed the corresponding bound for the original uncontrolled probability transition matrix (P), although this situation is not very common.

The simulation studies for validating a number of control approaches that we proposed earlier were performed on a network of 7 genes containing WNT5A. To maintain uniformity, we built PBNs for the same 7 genes and Fig. 27 shows the perturbation bound

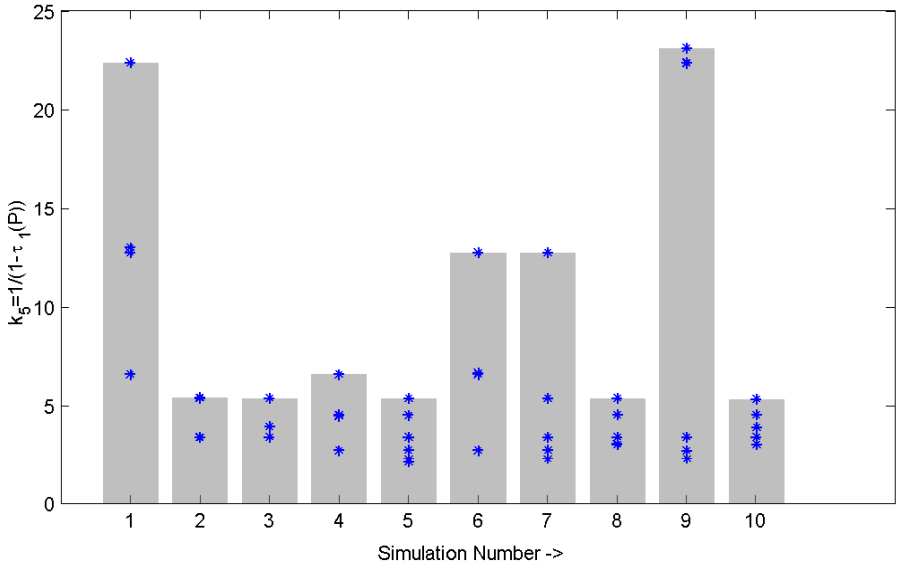


Fig. 25. Perturbation Bound k_5 for 10 Different PBNs (represented as bars) and the Stars Represent Perturbation Bounds for Random Stationary Policies Applied to the PBNs.

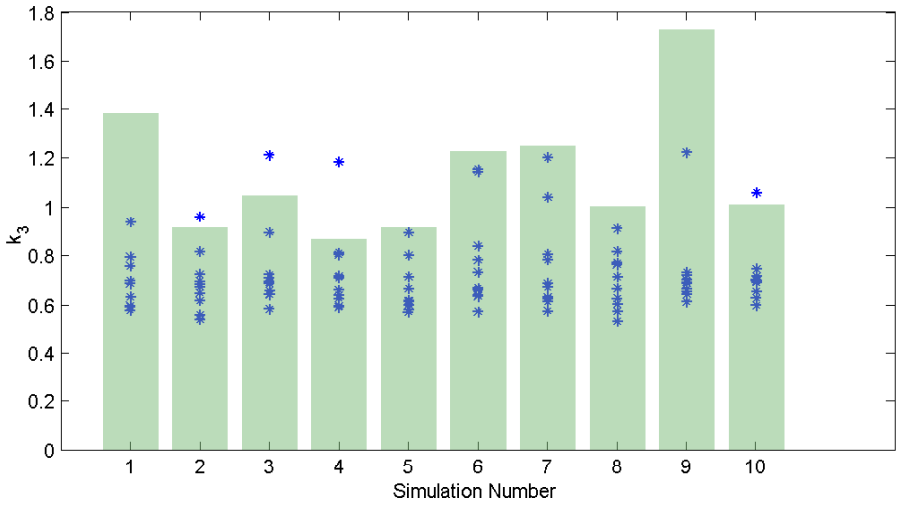


Fig. 26. Perturbation Bound k_3 for 10 Different PBNs (represented as bars) and the Stars Represent Perturbation Bounds for Random Stationary Policies Applied to the PBNs.

k_3 for a particular set of 80 simulations. The bars represent the perturbation bounds for the uncontrolled transition matrix P and the stars represent the perturbation bounds for the controlled transition matrix TP. The stationary policy corresponding to T represents the same objective as in [53], i.e. down-regulating the gene WNT5A and using a discounted cost infinite horizon approach. We should note that the perturbation bounds k_3 for the uncontrolled PBN with transition probability matrix P and the controlled PBN with transition probability matrix TP are quite similar. We performed a number of other simulations and all of them led to the same conclusions.

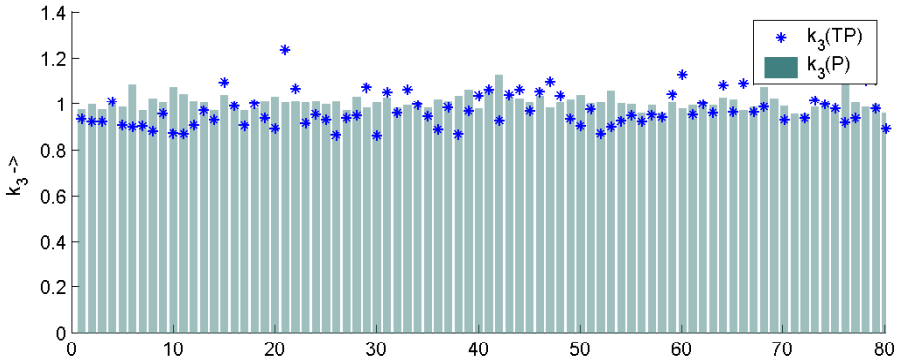


Fig. 27. Perturbation Bound k_3 for 80 Different PBNs (represented as bars) and the Stars Represent Perturbation Bounds for Stationary Policies Applied to the PBNs. Here the Number of Genes Is 7.

The reason behind highlighting these perturbation bounds is that they give us a measure of the maximum change in the steady-state distributions. If for instance, the estimated transition matrix (P) of a gene regulatory network has a small perturbation bound k_5, then we can rest assured that the steady-state ($\tilde{\pi}_c$) of the actual gene regulatory network ($\tilde{P}$) controlled by a stationary policy T will be close to the steady-state (π_c) of the gene regulatory network (P) controlled by the same stationary policy T . Thus for intervention strategies where the steady-state distribution is a measure of the effectiveness of the intervention, the perturbation bounds can provide a good estimate of the outcome of the control strategy. In

the case of a PBN whose perturbation bound k_5 is high, the perturbation bound k_3 can be used to give us some idea of the uncertainty involved. As shown by simulations, the difference between the values of k_3 for the original uncontrolled probability transition matrix P and the controlled probability transition matrix TP is quite small, and for this reason the uncertainty in the steady-state distribution after application of the stationary policy will be approximately the same as the uncertainty in the steady-state distribution of the original uncontrolled PBN.

Furthermore, the perturbation bounds can be used as a kind of measure for network selection. In general, genetic networks are quite stable or, in other words, robust to small perturbations. Hence a transition probability matrix representing a genetic network should necessarily be robust to perturbations and the alteration in its steady state for small changes in the transition probabilities should be minimal. In this context, it is appropriate to mention that in the field of genomics, it is still not clear as to what metric should be used to carry out network selection. The available data in genomics studies are quite limited and this can give rise to a number of possible networks that fit the data. The selection among these different networks is a very important issue and some initial approaches for doing this have been proposed in the literature, e.g. [8, 58]. The perturbation bound combined with other metrics could provide yet an alternative approach for doing so.

CHAPTER VI

CONCLUSIONS

This dissertation attempts to formulate the treatment of genetic diseases from the systems and control theoretic point of view. Four significant contributions in the context of inference and control of Boolean and Probabilistic Boolean networks are provided. In chapter III, we provided algorithms for inference of Boolean Networks from steady-state data. In pattern recognition, it is important to constrain the solution space when making inferences from limited data. We have applied that principle to Boolean networks by making assumptions on the dynamical structure of the network, assumptions that can be made in accordance with biological understanding. Since the algorithms generate networks in the constrained solution space, they can be used to provide synthetic networks to test proposed inference algorithms, for both Boolean and probabilistic Boolean networks. Two important points should be noted. First, the algorithms, both programmed in C, can easily be parallelized for supercomputer implementation to synthesize larger networks under the assumption that larger data sets will become available. Second, while this report has focused on binary-valued networks, there is nothing inherently binary in the algorithms and they can be directly applied to more finely quantized networks, albeit, at the cost of much larger solution spaces.

Chapter IV extends earlier results on intervention in instantaneously random PBNs without perturbation to context-sensitive PBNs with perturbation. The extension is significant because the latter class more closely models small biological subnetworks whose logical behavior is affected by conditions outside the genes represented in the model network. The results show that the expected cost with control is much lower than without control. In addition, the results indicate that we can achieve a much better control outcome

if a gene with high influence is selected as the control gene.

In Chapter IV Section B, we formulated the optimal infinite-horizon control problem and its solution for context-sensitive PBNs. The stationary policies obtained are much easier to implement than a policy that changes with time. Depending on which is more vital to us: current condition of the patient or the condition over a long length of time, we can utilize the discounted-cost or average-cost formulation. The melanoma application shows that we can shift the stationary distribution towards states with lower metastatic competence using the stationary control policy.

Finally, we have studied the robustness of the infinite horizon intervention and examined how uncertainties in the transition probability matrix of the uncontrolled PBN show up in the steady-state distribution of the controlled PBN. Since the steady-state distribution of a PBN is thought to characterize the phenotype, our studies essentially seek to examine the effect of network uncertainty on the phenotype that would result from the application of intervention strategies. Through analytical derivation and simulation studies, we demonstrated that the stationary infinite horizon optimal control policies proposed in Chapter IV are quite robust with respect to network uncertainty. The intervention strategies for PBNs that have been proposed thus far are all limited to flipping the expression status of one or more genes in the network, and this is dictated by what interventions are implementable with the currently available biological techniques. This limited class of interventions ensures that the controlled probability transition matrix is related to the uncontrolled probability transition matrix via a linear transformation, and this is what made it possible to establish the robustness results.

REFERENCES

[1] S. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *Journal of Theoretical Biology*, vol. 22, pp. 437–467, 1969.

[2] ——, "Homeostasis and differentiation in random genetic control networks," *Nature*, vol. 224, pp. 177–178, 1969.

[3] ——, *The Origins of Order: Self-Organization and Selection in Evolution*. New York: Oxford University Press, 1993.

[4] I. Shmulevich, E. R. Dougherty, and W. Zhang, "From boolean to probabilistic boolean networks as models of genetic regulatory networks," *Proceedings of IEEE*, vol. 90, pp. 1778–1792, 2002.

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

[6] N. Friedman, M. Linial, I. Nachman, and D. Pe'er, "Bayesian networks to analyze expression data," in *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology*, 2000, pp. 127–135.

[7] R. F. Hashimoto, K. Seungchan, I. Shmulevich, W. Zhang, M. L. Bittner, and E. R. Dougherty, "Growing genetic regulatory networks from seed genes," *Bioinformatics*, vol. 20, pp. 1241–1247, 2004.

[8] R. Pal, I. Ivanov, A. Datta, M. L. Bittner, and E. R. Dougherty, "Generating boolean networks with a prescribed attractor structure," *Bioinformatics*, vol. 21, pp. 4021–4025, 2005.

[9] X. Zhou, X. Wang, R. Pal, I. Ivanov, M. L. Bittner, and E. R. Dougherty, “A bayesian connectivity-based approach to constructing probabilistic gene regulatory networks,” *Bioinformatics*, vol. 20, pp. 2918–2927, 2004.

[10] I. Shmulevich, E. R. Dougherty, and W. Zhang, “Gene perturbation and intervention in probabilistic boolean networks,” *Bioinformatics*, vol. 18, pp. 1319–1331, 2002.

[11] ——, “Control of stationary behaviour in probabilistic boolean networks by means of structural intervention,” *Biological Systems*, vol. 10, pp. 431–446, 2002.

[12] A. Datta, A. Choudhary, M. L. Bittner, and E. R. Dougherty, “External control in markovian genetic regulatory networks,” *Machine Learning*, vol. 52, pp. 169–191, 2003.

[13] R. Pal, A. Datta, M. L. Bittner, and E. R. Dougherty, “Intervention in context-sensitive probabilistic boolean networks,” *Bioinformatics*, vol. 21, pp. 1211–1218, 2005.

[14] D. Endy and R. Brent, “Modelling cellular behaviour,” *Nature*, vol. 409, pp. 391–395, 2001.

[15] T. Ideker, T. Galitski, and L. Hood, “A new approach to decoding life: Systems biology,” *Annual Review of Genomics and Human Genetics*, vol. 2, pp. 343–372, 2001.

[16] J. Hasty, D. McMillen, F. Issacs, and J. J. Collins, “Computational studies of gene regulatory networks: in numero molecular biology,” *Nature Reviews Genetics*, vol. 2, pp. 268–279, 2001.

[17] M. Karamouzis, V. Gorgoulis, and A. Papavassiliou, “Transcription factors and neoplasia: vistas in novel drug design,” *Clinical Cancer Research*, vol. 8, pp. 949–961, 2002.

[18] D. M. Wolf and F. H. Eckman, “On the relationship between genomic regulatory element organization and gene regulatory dynamics,” *Journal of Theoretical Biology*, vol. 195, pp. 167–186, 1998.

[19] T. Ideker, V. Thorsson, J. A. Ranish, R. Christmas, J. Buhler, J. K. Eng, R. Bumgarner, D. R. Goodlett, R. Aebersold, and L. Hood, “Integrated genomic and proteomic analyses of a systematically perturbed metabolic network,” *Science*, vol. 292, pp. 929–934, 2001.

[20] K. Murphy and S. Mian, “Modelling gene expression data using dynamic bayesian networks,” Computer Science Division, University of California, Berkeley, CA, Tech. Rep., 1999.

[21] A. Hartemink, D. Gifford, T. Jaakkola, and R. Young, “Maximum likelihood estimation of optimal scaling factors for expression array normalization,” in *International Symposium on Biomedical Optics (BiOS 2001)*, 2001, pp. 132–140.

[22] D. C. Weaver, C. T. Workman, and G. D. Stormo, “Modeling regulatory networks with weight matrices,” in *Pac. Symp. Biocomputing, Hawaii*, 1999, pp. 112–123.

[23] M. Wahde and J. Hertz, “Coarse-grained reverse engineering of genetic regulatory networks,” *BioSystems*, vol. 55, pp. 129–136, 2000.

[24] T. Mestl, E. Plahte, and S. W. Omholt, “A mathematical framework for describing and analysing gene regulatory networks,” *Journal of Theoretical Biology*, vol. 176, pp. 291–300, 1995.

[25] H. De Jong, “Modeling and simulation of genetic regulatory systems: A literature review,” *Journal of Computational Biology*, vol. 9, pp. 67–103, 2001.

[26] I. Ivanov and E. R. Dougherty, "Modeling genetic regulatory networks: Continuous or discrete?" *Journal of Biological Systems*, vol. 14, pp. 219–229, 2006.

[27] L. Glass and S. Kauffman, "The logical analysis of continuous, nonlinear biochemical control networks," *Journal of Theoretical Biology*, vol. 39, pp. 103–129, 1973.

[28] I. Shmulevich and W. Zhang, "Binary analysis and optimization-based normalization of gene expression data," *Bioinformatics*, vol. 18, pp. 555–565, 2002.

[29] X. Zhou, X. Wang, and E. R. Dougherty, "Binarization of microarray data based on a mixture model," *Molecular Cancer Therapeutics*, vol. 2, pp. 679–684, 2003.

[30] R. Pal, A. Datta, A. Fornace, M. Bittner, and E. Dougherty, "Boolean relationships among genes responsive to ionizing radiation in the nci 60 acds," *Bioinformatics*, vol. 21, pp. 1542–1549, 2005.

[31] C.-H. Yuh, H. Bolouri, and E. H. Davidson, "Genomic cis-regulatory logic: Experimental and computational analysis of a sea urchin gene," *Science*, vol. 279, pp. 1896 – 1902, 1998.

[32] E. H. Davidson, J. P. Rast, P. Oliveri, A. Ransick, C. Calestani, C.-H. Yuh, T. Minokawa, G. Amore, V. Hinman, C. Arenas-Mena, O. Otim, C. T. Brown, C. B. Livi, P. Y. Lee, R. Revilla, A. G. Rust, Z. jun Pan, M. J. Schilstra, P. J. C. Clarke, M. I. Arnone, L. Rowen, R. A. Cameron, D. R. McClay, L. Hood, and H. Bolouri, "A genomic regulatory network for development," *Science*, vol. 295, pp. 1669–1678, 2002.

[33] S. Huang, "Gene expression profiling, genetic networks, and cellular states: An integrating concept for tumorigenesis and drug discovery," *Journal of Molecular Medicine*, vol. 77, pp. 469–480, 1999.

[34] E. Dougherty, S. Kim, and Y. Chen, “Coefficient of determination in nonlinear signal processing,” *Signal Processing*, vol. 80, pp. 2219–2235, 2000.

[35] S. Kim, E. R. Dougherty, M. L. Bittner, Y. Chen, K. Sivakumar, P. Meltzer, and J. M. Trent, “General nonlinear framework for the analysis of gene interaction via multivariate expression arrays,” *Journal of Biomedical Optics*, vol. 5, pp. 411–424, 2000.

[36] M. Brun, E. R. Dougherty, and I. Shmulevich, “Steady-state probabilities for attractors in probabilistic boolean networks,” *Signal Processing*, vol. 85, pp. 1993–2013, 2005.

[37] H. Lahdesmaki, I. Shmulevich, and O. Yli-Harja, “On learning gene regulatory networks under the boolean network model,” *Machine Learning*, vol. 52, pp. 147–167, 2003.

[38] X. Zhou, X. Wang, and E. R. Dougherty, “Construction of genomic networks using mutual-information clustering and reversible-jump markov-chain-monte-carlo predictor design,” *Signal Processing*, vol. 83, pp. 745–761, 2003.

[39] S. Kim, H. Li, E. R. Dougherty, N. Chao, Y. Chen, M. L. Bittner, and E. B. Suh, “Can markov chain models mimic biological regulation ?” *Biological Systems*, vol. 10, pp. 337–357, 2002.

[40] E. R. Dougherty and Y. Xiao, “Design of probabilistic boolean networks under the requirement of contextual data consistency,” *IEEE Transactions on Signal Processing*, vol. 54, pp. 3603–13, 2006.

[41] D. Peer, A. Regev, G. Elidan, and N. Friedman, “Inferring subnetworks from perturbed expression profiles,” *Bioinformatics*, vol. 17, pp. 215–224, 2001.

[42] D. Husmeier, "Reverse engineering of genetic networks with bayesian networks," *Biochemical Society Transactions*, vol. 31, pp. 1516–1518, 2003.

[43] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufman Publishers, 1988.

[44] S. L. Lauritzen, *Graphical Models*. Oxford: Oxford University Press, 1996.

[45] F. V. Jensen, *Bayesian Networks and Decision Graphs*. New York: Springer-Verlag, 2001.

[46] H. Lahdesmaki, O. Yli-Harja, W. Zhang, and I. Shmulevich, "Intrinsic dimensionality in gene expression analysis," in *IEEE International Workshop on Genomic Signal Processing and Statistics*, 2005.

[47] N. Deo, *Graph Theory with Applications to Engineering and Computer Science*. Englewood Cliffs, NJ: Prentice-Hall, 1974.

[48] M. Bittner, P. Meltzer, Y. Chen, Y. Jiang, E. Seftor, M. Hendrix, M. Radmacher, R. Simon, Z. Yakhini, A. Ben-Dor, N. Sampas, E. Dougherty, E. Wang, F. Marincola, C. Gooden, J. Lueders, A. Glatfelter, P. Pollock, J. Carpten, E. Gillanders, D. Leja, K. Dietrich, C. Beaudry, M. Berens, D. Alberts, and V. Sondak, "Molecular classification of cutaneous malignant melanoma by gene expression profiling," *Nature*, vol. 406, pp. 536–540, 2000.

[49] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 2nd ed. Belmont, MA: Athena Scientific, 2001.

[50] A. L. Gartel and A. L. Tyner, "Transcriptional regulation of p21 (waf1/cip1) gene," *Experimental Cell Research*, vol. 246, pp. 280–289, 1999.

[51] A. T. Weeraratna, Y. Jiang, G. Hostetter, K. Rosenblatt, P. Duray, M. Bittner, and T. J. M., "Wnt5a signalling directly affects cell motility and invasion of metastatic melanoma," *Cancer Cell*, vol. 1, pp. 279–288, 2002.

[52] A. Datta, A. Choudhary, M. L. Bittner, and E. R. Dougherty, "External control in markovian genetic regulatory networks: The imperfect information case," *Bioinformatics*, vol. 20, pp. 924–930, 2004.

[53] R. Pal, A. Datta, and E. R. Dougherty, "Optimal infinite horizon control for probabilistic boolean networks," *IEEE Transactions on Signal Processing*, vol. 54, pp. 2375– 2387, 2006.

[54] A. Choudhary, A. Datta, M. L. Bittner, and E. R. Dougherty, "Intervention in a family of boolean networks," *Bioinformatics*, vol. 22, pp. 226–232, 2006.

[55] G. E. Cho and C. D. Meyer, "Comparison of perturbation bounds for the stationary distribution of a markov chain," *Linear Algebra and Its Applications*, vol. 335, pp. 137–150, 2001.

[56] E. Seneta, "Perturbation of the stationary distribution measured by ergodicity coefficients," *Advances in Applied Probability*, pp. 228–230, 1988.

[57] ——, "Sensitivity of finite markov chains under perturbation,," *Statistics and Probability Letters*, vol. 17, pp. 163–168, 1993.

[58] I. Ivanov, R. Pal, and E. Dougherty, "Dynamics-preserving size-reduction mappings for probabilistic boolean networks," *IEEE Transactions on Signal Processing*, vol. 55, pp. 2310–2322, 2007.

APPENDIX A

BN algorithm adaptation for multiple-state cyclic attractors

Algorithm 1 extension

STEP1: Randomly generate a set of k attractor states and their connections [1]. If the connections generate an attractor cycle with length $>$ Max Cycle Length, then repeat STEP1. If STEP1 has been repeated more than a pre-specified number of times, then terminate the algorithm.

STEP2: Randomly pick up a predictor set W, where each W_i has not less than m and not more than M elements. If STEP2 has been repeated more than a pre-specified number of times go back to STEP1.

STEP3: Check if the selected attractor set is compatible with W, i.e. the attractor set transitions [2] of the state transition diagram are checked for compatibility against W. If the attractor set is not compatible with W, then go back to STEP2; otherwise continue to STEP4.

---

[1] If Max Cycle Length is given to be 1, then we connect each attractor to itself. Otherwise we include random connections between attractors. Let the random attractors selected be a_1, a_2, a_3 and a_4. If Max Cycle Length is 1, then the connections are $a_1 \rightarrow a_1$, $a_2 \rightarrow a_2$, $a_3 \rightarrow a_3$, $a_4 \rightarrow a_4$. Otherwise, there are 4 attractors and hence we choose a random permutation of numbers 1 to 4. Say the random permutation is 1 ,3, 2 and 4. Then the connections between the attractors a_1, a_2, a_3 and a_4 will be $a_1 \rightarrow a_1$, $a_2 \rightarrow a_3$, $a_3 \rightarrow a_2$, $a_4 \rightarrow a_4$ (Figure 28). If the random permutation is 4, 3, 1 and 2, then the transitions between the attractors will be $a_1 \rightarrow a_4$, $a_2 \rightarrow a_3$, $a_3 \rightarrow a_1$, $a_4 \rightarrow a_2$.

[2] For the first random permutation 1 , 3, 2 , 4, the transitions are $a_1 \rightarrow a_1$, $a_2 \rightarrow a_3$, $a_3 \rightarrow a_2$, $a_4 \rightarrow a_4$; for the random permutation 4, 3, 1, 2 the transitions are $a_1 \rightarrow a_4$, $a_2 \rightarrow a_3$, $a_3 \rightarrow a_1$, $a_4 \rightarrow a_2$.

STEP4: Fill in the entries of the truth table that correspond to the attractor set transitions generated in STEP1. Using the predictor set W , randomly fill in the remaining entries of the truth table. If STEP4 has been repeated more than a pre-specified number of times go back to STEP2.

STEP5: Search for cycles of length $>$ Max Cycle Length in the state transition diagram $\tilde{\Gamma}$ that is associated with the truth table generated in STEP4. If a cycle is found, then go back to STEP4; otherwise continue to STEP6.

STEP6: If $\tilde{\Gamma}$ has less than l or more than L level sets, go back to STEP4; otherwise continue to STEP7.

STEP7: Save the generated BN and terminate the algorithm.

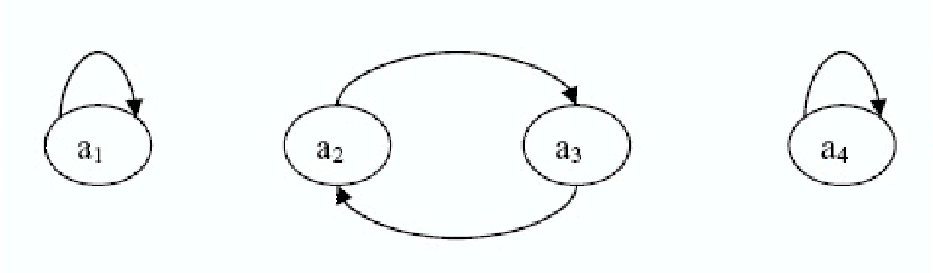


Fig. 28. Connections among Attractors.

Algorithm 2 extension

STEP1: Randomly generate a state transition diagram $\tilde{\Gamma}$ that satisfies the design goals about the attractor structure [3] and level set structure. If STEP1 has been repeated more than a pre-specified number of times, then terminate the algorithm.

STEP2: Fill in the truth table using $\tilde{\Gamma}$.

[3]When attractor cycles of length $\leq$ Max Cycle Length are allowed, then connections between attractors are permited and only those state transition diagram are selected for the subsequent steps whose cycle lengths are $\leq$ Max Cycle Length.

STEP3: If there is at least one W_i in the predictor set W given by the truth table that has less than m or more than M, then elements go back to STEP1; otherwise continue to STEP4.

STEP4: Save the generated BN and terminate the algorithm.

APPENDIX B

Proof of Theorem B.1

We want to show that

$$J^*(z) = \lim_{M\to\infty}(T^M J)(z), \text{for all } z \in S$$

Now

$$\begin{aligned} J_\pi(z_0) &= \lim_{M\to\infty} E\{\sum_{t=0}^{M-1} \alpha^t g(z_t, \mu_t(z_t))\} \\ &= E\{\sum_{t=0}^{K-1} \alpha^t g(z_t, \mu_t(z_t))\} \\ &+ \lim_{M\to\infty} E\{\sum_{t=K}^{M-1} \alpha^t g(z_t, \mu_t(z_t))\}. \end{aligned}$$

Using $|g(z,u)| \le B$, we obtain

$$|\lim_{M\to\infty} E\{\sum_{t=K}^{M-1} \alpha^t g(z_t, \mu_t(z_t))\}| \le B\sum_{t=K}^{\infty}\alpha^t = \frac{\alpha^K B}{1-\alpha}$$

Using these relations, we can write the inequalities

$$\begin{aligned} J_\pi(z_0) &- \frac{\alpha^K B}{1-\alpha} - \alpha^K \max_{z\in S}|J(z)| \\ &\le E[\alpha^K J(z_K) + \sum_{t=0}^{K-1}\alpha^t g(z_t, \mu_t(z_t))] \\ &\le J_\pi(z_0) + \frac{\alpha^K B}{1-\alpha} + \alpha^K \max_{z\in S}|J(z)| \end{aligned} \tag{B.1}$$

Minimization over π gives us

$$\begin{aligned} J^*(z_0) - \frac{\alpha^K B}{1-\alpha} &- \alpha^K \max_{z \in S} |J(z)| \\ &\leq (T^K J)(z_0) \\ &\leq J^*(z_0) + \frac{\alpha^K B}{1-\alpha} + \alpha^K \max_{z \in S} |J(z)| \end{aligned} \tag{B.2}$$

If we take the limit as $K \to \infty$ and utilize the fact that $\alpha < 1$, the result follows.

Proof of Theorem B.2

From Eq. (B.2), we have for all $z \in S$ and M

$$J^*(z) - \frac{\alpha^M B}{1-\alpha} \leq (T^M J_0)(z) \leq J^*(z) + \frac{\alpha^M B}{1-\alpha}$$

where J_0 is the zero function [$J_0(z) = 0$ for all $z \in S$]. Applying the mapping T to this relation and using the Monotonicity property of mapping T, we obtain for all $z \in S$ and M

$$(TJ^*)(z) - \frac{\alpha^{M+1} B}{1-\alpha} \leq (T^{M+1} J_0)(z) \leq (TJ^*)(z) + \frac{\alpha^{M+1} B}{1-\alpha}$$

Since $(T^{M+1} J_0)(z)$ converges to $J^*(z)$ (from convergence of DP algorithm), by taking the limit as $M \to \infty$ in the previous inequality, we obtain $J^* = TJ^*$. Uniqueness of the solution can be proved as follows: if J is bounded and satisfies $J = TJ$, then $J = \lim_{M \to \infty} T^M J$ and by Convergence of DP algorithm, we have $J = J^*$.

Proof of Theorem B.3

If $TJ^* = T_\mu J^*$, then using Bellman's equation ($J^* = TJ^*$), we have $J^* = T_\mu J^*$, so by the uniqueness property of optimal solution , we obtain $J^* = J_\mu$; i.e., μ is optimal. On the other hand, if the stationary policy μ is optimal, we have $J^* = J_\mu$, which yields $J^* = T_\mu J^*$. Combining this with Bellman's equation ($J^* = TJ^*$), we obtain $TJ^* = T_\mu J^*$.

Proof of Theorem B.4

First, let us assume some reference state rs is recurrent for the optimal stationary policy π. Then the average cost from state i ($\in$ communicating class containing rs), defined by

$$J_\pi(z_0 = i) = \lim_{M\to\infty} \frac{1}{M} E\{\sum_{t=0}^{M-1} g(z_t, \mu_t(z_t))\}$$

can be written as

$$J_\pi(z_0 = i) = \lim_{M\to\infty} \frac{1}{M} [E\{\sum_{t=0}^{r_i-1} g(z_t, \mu_t(z_t))\} + \sum_{t=r_i}^{M-1} g(z_t, \mu_t(z_t))\}],$$

where r_i is the smallest integer such that z_{r_i}=rs. Recurrence of the state rs guarantees the finiteness of r_i. Hence, when $M \to \infty$, the first term becomes negligible and we have

$$J_\pi(z_0 = i) = J_\pi(z_0 = rs)$$

thereby showing that the optimal average cost per stage is independent of the starting state.

Next, if the application of the Stationary policy breaks up the states into separate communicating classes (C_1,..C_s) then each communicating class will have some average cost ($co_1, ...co_s$). The optimal stationary policy will always drive the states to that communicating class which has the lowest average cost. Hence for the optimal stationary policy, we can always find a state rs which will be approachable from all other states in a finite number of time steps. This state rs can be used in the preceding arguments to show that the optimal average cost per stage is independent of the starting state, even in this case.

Proof of Theorem B.5

Let $\pi = \{\mu_0, \mu_1....\}$ be any admissible policy and let M be a positive Integer. By the proposition of Eq. (4.45), h satisfies, $\lambda e + h = Th$. Therefore, $T_{\mu_{M-1}} h \geq Th = \lambda e + h$. By applying $T_{\mu_{M-2}}$ to both sides of this relation, and by using the monotonicity of $T_{\mu_{M-2}}$,

we have $T_{\mu_{M-2}} T_{\mu_{M-1}} h \geq T_{\mu_{M-2}}(\lambda e + h) = \lambda e + T_{\mu_{M-2}} h \geq 2\lambda e + h$.

Continuing in the same manner, we finally obtain

$$T_{\mu_0} T_{\mu_1} .... T_{\mu_{M-1}} h \geq M\lambda e + h \tag{B.3}$$

with equality if each μ_t, $t = 0, 1, ..., M-1$, attains the minimum in Eq. (4.45). $T_{\mu_0} T_{\mu_1} .... T_{\mu_{M-1}} h(i)$ is equal to the M-stage cost corresponding to initial state i, policy $\mu_0, \mu_1, ...\mu_{M-1}$, and terminal cost function h; i.e.,

$$T_{\mu_0} T_{\mu_1} .... T_{\mu_{M-1}} h(i) = E\{h(z_M) + \sum_{t=0}^{M-1} g(z_t, \mu_t(z_t)) | z_0 = i, \pi\} \tag{B.4}$$

Using this relation in Eq. (B.3) and dividing by M, we obtain for all $i \in S$

$$\frac{1}{M} E\{h(z_M) | z_0 = i, \pi\} + \frac{1}{M} E\{\sum_{t=0}^{M-1} g(z_t, \mu_t(z_t)) | z_0 = i, \pi\} \geq \lambda + \frac{1}{M} h(i). \tag{B.5}$$

If we look back at Eq. (4.17), then we realise that the second term in the above equation is in fact the average cost per stage i.e. $J_\pi(i)$ for large M. By taking the limit as $M \to \infty$, we have $J_\pi(i) \geq \lambda$, i=1,....,n, with equality if $\mu_t(i)$, $t = 0, 1, ...$ attains the minimum in Eq. (4.45) for each i. Hence λ is the optimal average cost per stage and by Theorem B.4, λ is the same for every initial state i.

APPENDIX C

Derivation of Alternative Expressions for the Ergodicity Coefficient

Theorem .3.

$$\sup_{\substack{|x^T|_1=1 \\ x^T\mathbf{1}_n=0}} |x^TP|_1 = \frac{1}{2}\max_{i,j}\sum_{s=1}^{n}|p_{is}-p_{js}| = 1-\min_{i,j}\sum_{s=1}^{n}\min(p_{is},p_{js}).$$

Proof. First of all, we will show that $\sup_{\substack{|x^T|_1=1 \\ x^T\mathbf{1}_n=0}} |x^TP|_1$ can achieve the value $\frac{1}{2}\max_{i,j}$ $\sum_{s=1}^{n}|p_{is}-p_{js}|$. Let the maximum on the right hand side be achieved for indices i' and j'. Choose $x^T = [0,..0,1/2,0,..0,-1/2,0,..0]$, where all the entries of x are zero except for $x(i') = 1/2$ and $x(j') = -1/2$. Clearly, x satisfies the constraints $x^T\mathbf{1}_n = 0$ and $|x^T|_1 = 1$, and for this x, we have $|x^TP|_1 = \frac{1}{2}\sum_{s=1}^{n}|p_{i's}-p_{j's}|$. Hence,

$$\sup_{\substack{|x^T|_1=1 \\ x^T\mathbf{1}_n=0}} |x^TP|_1 \geq \frac{1}{2}\max_{i,j}\sum_{s=1}^{n}|p_{is}-p_{js}|. \tag{C.1}$$

The second part of the proof will consist of showing that $\sup_{\substack{|x^T|_1=1 \\ x^T\mathbf{1}_n=0}} |x^TP|_1 \leq$ $\frac{1}{2}\max_{i,j}\sum_{s=1}^{n}|p_{is}-p_{js}|$. To that end, for any $x \in R^n$, define $I_+ = \{i : x_i \geq 0\}$, $I_- = \{i : x_i < 0\}$, $u_+ = \sum_{i\in I_+}|x_i|$, $u_- = \sum_{i\in I_-}|x_i|$.

As $x^T\mathbf{1}_n = 0$ and $|x^T|_1 = 1$, it follows that $u_+ = 1/2$ and $u_- = 1/2$.

$$
\begin{aligned}
\text{Now } |x^T P|_1 &= \sum_{j=1}^{n} |\sum_{i=1}^{n} x_i p_{ij}| \\
&= \sum_{j=1}^{n} |\{\sum_{i\in I_+} x_i p_{ij} - \sum_{k\in I_-} |x_k| p_{kj}\}| \\
&= \sum_{j=1}^{n} |\{\frac{1}{2}(2\sum_{k\in I_-} |x_k|) \sum_{i\in I_+} 2x_i p_{ij} - \frac{1}{2}(2\sum_{i\in I_+} x_i) \sum_{k\in I_-} 2|x_k| p_{kj}\}| \\
&\qquad (\text{ since } u_+ = u_- = 1/2) \\
&= \sum_{j=1}^{n} |\{\sum_{k\in I_-}\sum_{i\in I_+} 4|x_k| x_i \frac{1}{2} p_{ij} - \sum_{k\in I_-}\sum_{i\in I_+} 4|x_k| x_i \frac{1}{2} p_{kj}\}| \\
&= \sum_{j=1}^{n} |\{4\sum_{k\in I_-}\sum_{i\in I_+} |x_k| x_i \frac{1}{2}(p_{ij} - p_{kj})\}| \\
&\leq \sum_{k\in I_-}\sum_{i\in I_+} \frac{|x_k|}{1/2} \frac{x_i}{1/2} \frac{1}{2} \sum_{j=1}^{n} |p_{ij} - p_{kj}|
\end{aligned}
$$

$$
\leq \frac{1}{2} \max_{i,k} \sum_{j=1}^{n} |p_{ij} - p_{kj}| \qquad \text{(C.2)}
$$

$$
(\text{since } \sum_{i\in I_+} |x_i| = 1/2 \text{ and } \sum_{i\in I_-} |x_i| = 1/2).
$$

From Eqs. C.1 and C.2, it follows that $\sup_{\substack{|x^T|_1=1 \\ x^T \mathbf{1}_n = 0}} |x^T P|_1 = \frac{1}{2} \max_{i,j} \sum_{s=1}^{n} |p_{is} - p_{js}|$.

To prove the next equality, let us focus on the term $\frac{1}{2}\max_{i,j}\sum_{s=1}^{n}|p_{is}-p_{js}|$.

$$\begin{aligned}
\text{Now } \frac{1}{2}\max_{i,j}\sum_{s=1}^{n}|p_{is}-p_{js}| &= \frac{1}{2}\max_{i,j}\sum_{s=1}^{n}(p_{is}+p_{js}-2\min(p_{is},p_{js})) \\
&= \frac{1}{2}(\max_{i,j}\{\sum_{s=1}^{n}p_{is}+\sum_{s=1}^{n}p_{js}-2\sum_{s=1}^{n}\min(p_{is},p_{js})\}) \\
&= \frac{1}{2}\max_{i,j}\{1+1-2\sum_{s=1}^{n}\min(p_{is},p_{js})\} \\
&= 1+\max_{i,j}\{-\sum_{s=1}^{n}\min(p_{is},p_{js})\} \\
&= 1-\min_{i,j}\sum_{s=1}^{n}\min(p_{is},p_{js}).
\end{aligned}$$

This completes the proof.

□

VITA

Ranadip Pal received the B.Tech. degree in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur, India, in 2002 and the M.S. degree in electrical engineering from Texas A&M University, College Station, in 2004. Pal earned his PhD degree in electrical engineering from Texas A&M University, College Station, in the summer of 2007.

His research areas are computational biology, genomic signal processing, and control of genetic regulatory networks.

Honors include the Ebensberger/Fouraker Fellowship, a Distinguished Graduate Student Master's Research Award and a National Instruments Fellowship. Pal also was an Indian National Math Olympiad Awardee and was ranked 1st in the Regional Math Olympiad, West Bengal, India. He can be reached at Zachry 09, Department of electrical and computer engineering, Texas A&M University, College Station, TX, 77843.

The typist for this dissertation was Ranadip Pal.