BOOLEAN MODELS FOR GENETIC REGULATORY NETWORKS

A Dissertation

by

YUFEI XIAO

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

BOOLEAN MODELS FOR GENETIC REGULATORY NETWORKS

A Dissertation

by

YUFEI XIAO

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

ABSTRACT

Boolean Models for

Genetic Regulatory Networks. (August 2007)

Yufei Xiao, B.S., Zhejiang University;

M.S., University of Virginia

Chair of Advisory Committee: Dr. Edward R. Dougherty

This dissertation attempts to answer some of the vital questions involved in the genetic regulatory networks: inference, optimization and robustness of the mathematical models. Network inference constitutes one of the central goals of genomic signal processing. When inferring rule-based Boolean models of genetic regulations, the same values of predictor genes can correspond to different values of the target gene because of inconsistencies in the data set. To resolve this issue, a consistency-based inference method is developed to model a probabilistic genetic regulatory network, which consists of a family of Boolean networks, each governed by a set of regulatory functions. The existence of alternative function outputs can be interpreted as the result of random switches between the constituent networks. This model focuses on the global behavior of genetic networks and reflects the biological determinism and stochasticity.

When inferring a network from microarray data, it is often the case that the sample size is not sufficiently large to infer the network fully, such that it is necessary to perform model selection through an optimization procedure. To this end, the network connectivity and the physical realization of the regulatory rules should be taken into consideration. Two algorithms are developed for the purpose. One algorithm finds the minimal realization of the network constrained by the connectivity, and the other algorithm is mathematically proven to provide the minimally connected network constrained by the minimal realization.

Genetic regulatory networks are subject to modeling uncertainties and perturbations, which brings the issue of robustness. From the perspective of network stability, robustness is desirable; however, from the perspective of intervention to exert influence on network behavior, it is undesirable. A theory is developed to study the impact of function perturbations in Boolean networks: It finds the exact number of affected state transitions and attractors, and predicts the new state transitions and robust/fragile attractors given a specific perturbation. Based on the theory, one algorithm is proposed to structurally alter the network to achieve a more favorable steady-state distribution, and the other is designed to identify function perturbations that have caused changes in the network behavior, respectively.

To my parents, sister and late grandparents

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Prof. Edward R. Dougherty, for his guidance in my doctoral research. He is an avid researcher and inspiring mentor, who has helped my academic progress tremendously and made research an exciting journey of discovery for me. I am grateful to have Prof. Erchin Serpedin, Prof. Aniruddha Datta, and Prof. Bani Mallick as members on my dissertation committee. During my Ph.D. studies, former and current students and researchers of the Genomic Signal Processing Laboratory (directed by Prof. Dougherty) have given me generous help and encouragement, and my special thanks are to Dr. Jianping Hua, Dr. Ivan Ivanov and Dr. Ranadip Pal. Finally, I would like to thank my parents and sister for their love and support, and they have always had faith in me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

A. Genomic Signal Processing

The Genome carries the hereditary information of an organism, and is encoded in the double-helix DNA molecules in the cells. Genome includes genes and non-coding DNA sequences. Genes encode the information for producing proteins. The non-coding DNA sequences, often called the non-coding genes, refer to the DNA sequences that do not encode any protein, and they either have no known function, or play some roles in the regulation of other genes. In the remaining text, we will use the term "genes" for both the protein-coding genes and the non-coding genes.

The size of a genome typically ranges from thousands of DNA base pairs to several billion. Human genome has about three billion DNA base pairs, and there are approximately $20,000$ genes. Among them, only a small percentage has been associated with known functionalities, and the vast majority remains to be studied. One way to investigate a genetic function is to study its expression (the amount of mRNAs actively engaged in the transcription of a certain gene, which is an indicator of gene activity level). Recent high-throughput technologies, such as cDNA microarrays, have made it possible to obtain large scale measurement of gene expressions. Two salient goals of functional genomics are [2]:

- Screening for key genes and gene combinations that account for specific cellular phenotypes (e.g., disease) and revealing the mechanism;
- Using genomic signals to classify disease on a molecular level.

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

Genomic Signal Processing is an emerging inter-disciplinary area that incorporates engineering methods in the study of the genome. It is characterized by an important feature: It studies the genome in a systematic manner, by working with several or even hundreds of genes at one time instead of considering one single gene. Genomic Signal Processing intends to obtain the information of the relationship between the genes, to identify the genes that are relevant to certain biological functions, and to reconstruct some abstract mathematical or engineering models that describe genetic regulations and make predictions.

As the genes in an organism constantly work together to perform biological functions, and regulations of their activities (such as promotion or suppression) involve not only genes, but also RNAs and proteins, altogether they form the so called *genetic regulatory networks*. Genetic regulatory networks is one of the key issues for genomic signal processing. First, we need to model the genetic regulatory network so as to capture the critical dynamics of gene activities, or describe the interactions among genes, RNAs and proteins that contribute to the regulation of gene expressions. Second, we would like to use the model for predictions; for instance, to determine the expression level of a target gene, we would like to identify a group of genes that are a good indicator of the activity of the target gene, and establish a mapping (such as a linear function or a neural network) from the group of genes to the target gene. Third, we would like to develop intervening schemes that steer a genetic regulatory network toward a desirable state, which could be applied in the treating of a disease that is linked with the malfunctioning of genetic regulations.

This dissertation concerns the modeling and optimization of genetic regulatory networks based on microarray gene expression data. Theory will be developed to analyze the robustness and sensitivity of the networks subject to perturbation, and applied to the network intervention and fault identification. Examples will be given to illustrate the ideas and proposed procedures, and demonstrate the application of the methods and theories to real genomic data.

B. Overview of Genetic Regulatory Networks

Genetic regulatory networks refer to "collections of DNA segments in a cell which interact with each other and with other substances in the cell, thereby governing the rates at which genes in the network are transcribed into mRNA. Generally speaking, a genetic network not only contains genes, but proteins, transcriptional factors, mRNAs, etc."[3]

The design of gene regulatory networks is a key issue in genomic signal processing [2, 4, 5]. There are numerous mathematical models of genetic regulatory networks, including differential equations, Bayesian networks, Boolean networks, et al. Modeling can be based on prior biological knowledge, on data obtained from traditional measuring methods such as northern blots, or on data from high-throughput methods, such as cDNA microarrays. The present main interest and challenge is inferring models from microarray gene expression data, sometimes combined with other information. Gene expression levels are continuous variables, but genes often exhibit switch-like behavior, thus the expression levels can be quantized at two (-1 and 1, or 0 and 1) or three (-1,0 and 1) levels.

From gene expression data to genetic regulatory model is a reverse engineering problem. The biggest challenge is, as a biological system, gene regulatory network is an open system, subject to latent variables. Any gene regulatory network that we study is a subnetwork of the whole genome, not free of interactions with factors outside of subnet. As a result, the gene networks demonstrate both biological determinism and stochasticity. The next challenge is the gene expression data. One key interest in this area is on human functional genomics, and due to the complexity of human genome, the microarray data sets often contain thousands of genes, which makes modeling very difficult. Also, due to various factors, such as cost, ethics, legal issues, accessibility, etc, time series data are not always available, thus some important properties, such as causality of gene regulations, may not be properly inferred. Another difficulty is, sample size from human studies is usually small, setting restrictions to modeling accuracy. Moreover, to model genetic regulatory networks, it is important to make measurements under various conditions and perturbations, so as to reveal as much of the critical dynamics as possible, as already being done in the studies of some simple organisms (*E. Coli*, yeast, etc). However, perturbations may not be implemented in human genetic studies, which poses an ethical problem. Fortunately, the situation can be partly mitigated by measuring gene expressions from diverse phenotypes, which is a feasible solution.

Little consent has been reached on what is the best model. There are arguments on continuous versus discrete models, deterministic versus stochastic models, fine-scale versus coarse models, etc. The primary reason lies in that, no single model can fully describe all the facades of gene regulatory activities, which manifest themselves at multiple levels. Each model can only provide insight into some specific aspects of the regulatory network, not the whole picture. The secondary reason has to do with the complexity of each model: According Occam's Razor, one should choose a model that is just complex enough to explain the data. Thus different models should be chosen for different data sets and different purposes.

To evaluate goodness of inference results, we must take into consideration of our goals and the difference between models: e.g., if we are concerned with the successful prediction of a target gene, the correctness of causality of gene regulation may not matter. We must also keep in mind that, a good inference result does not only "fit" the data well, but should possess the power to predict new outcomes.

At present, differential equation model is the most accurate when giving the dynamic detail of expression level changes. Its inference can also be successful, with a very small curve-fitting error. However, its application is very limited because: it relies on some highly intensive computational methods (e.g., genetic programming) to search for parameters, and cannot be applied to large-scale networks; it needs high quality time-series data for inference, and the timing of sampling is very important; it needs a fairly good prior knowledge of the gene network, such that the chosen genes are tightly related to each other.

The most widely applied models are graphic models, including Bayesian networks, dynamic Bayesian networks, Boolean networks and probabilistic Boolean networks. Bayesian network and dynamic Bayesian network can be used for both continuous and quantized gene expressions, and can well capture the probabilistic nature of gene regulations. The former is usually used to model static dependency among genes, but it rules out the feedback of genes, which is a drawback. The latter can model dynamic networks, but requires time series expression data. Both require that the number of samples be not too small, and partial knowledge of some genetic interactions is desirable, otherwise the search space is huge and the results can be unreliable. Boolean networks and probabilistic Boolean networks are coarse-grained models, using binary (sometimes tertiary) quantized gene expressions, and describing regulatory rules by logic functions. They can model both the static and dynamic networks. Probabilistic Boolean networks are derived from Boolean networks, by combining the deterministic nature of Boolean networks and probabilistic nature of Bayesian networks. The advantages of Boolean models are that the gene regulatory rules are explicit, bearing physical meanings, unlike Bayesian models which are purely statistical.

Right now, it is hard to evaluate the soundness of all inferred models, either due to lack of prior biological knowledge for validation, or because there are seldom follow-up experiments conducted by biologists to verify a new prediction. However, some good results are available, either because some inferred relations are previously reported or a predicted outcome of the network is verified experimentally. In 2003, Eran Segal et al. [6] inferred module networks in *Saccharomyces cerevisiae* from expression data set, with much of the results being validated, and three new novel predictions are supported by a follow-up microarray experiment. One of the good results with probabilistic Boolean model is by Huai Li and Ming Zhan [7] in the study of BCR-ABL and insulin/IGF pathways, which correctly identified the leukemia drug target and genes important for longevity. One good result on human genetic networks is achieved by Paula Sebastiani et al. [8], by using Bayesian networks, and the constructed network can predict the occurrence of stroke in 114 sickle cell anemia patients with 98.2% accuracy. Another successful work is by Katia Basso, et al. [9], modeling regulatory networks in human B cells, where in the constructed MYC subnetwork, 29 of the 56 predicted first neighbors were previously reported.

CHAPTER II

RULE-BASED MODELS FOR GENETIC REGULATORY NETWORKS*

It has long been discovered that cells exhibit switch-like behavior in their functional regulations. For instance, the logical character of gene regulation has been recognized for some time [11, 12, 13] and the dynamical behavior of Boolean networks can be used to model many biologically phenomena, such as cellular state dynamics possessing switch-like behavior, stability, and hysteresis [14]. From almost the inception of microarray analysis, logical relations among genes have been constructed from the data [15, 16] and recently the manifestation of logical relations in the continuous data has been analyzed [17]. Besides the fact that we are concerned in many of our applications with ON-OFF type behavior, an important practical reason for working in the binary setting, or at least in the context of a very coarse quantization, is the exponentially increasing complexity (and therefore data requirement) with finer quantization. The general question as to whether certain genes, when quantized as binary switches, can be informative in separating phenotype classes such as tumors and normal tissue, as well as different stages of tumor development, depends on the bi-modality of their behavior. The potential for binary discrimination has been shown for clustering [18] and classification [19]. The former has a good discussion of binarization. With the use of microarray data, which integrates expression over a

*© 2006 IEEE. Reprinted from *IEEE Transactions on Signal Processing*, see [10] for complete publication information.

collection of cells, it should be recognized that we are modeling global behavior, not the activity of individual cells, so that binarization corresponds to global bi-modality.

A. Boolean Networks

Boolean network (BN) is a rule-based model of genetic regulatory networks introduced by Kauffman [20, 21, 13]. It can be formally denoted as $G(U, \mathbf{f})$ [22], where a set U consists of n binary-valued nodes $x_1, x_2, \cdots, x_n$ that represent n genes in a genetic regulatory network. Each node takes on one of the two possible values, 1 or 0 (ON or OFF), to represent the active (expressed) and inactive (not expressed) status of a gene. There are k_i inputs (also known as *predictors* or *parents*) to the ith node reflecting regulatory mechanisms. The value of node x_i at a discrete time step $t+1$ is determined by its k_i input nodes at time t through a Boolean function,

$$x_i(t+1) = f_i(x_{i1}(t), x_{i2}(t), \cdots, x_{ik_i}(t)), \quad i = 1, 2, \cdots, n. \tag{2.1}$$

The number k_i is called the *connectivity* to node x_i and $K = \max_i k_i$ is the maximum connectivity of a Boolean network.

The *state* of the BN is denoted by a binary vector $\mathbf{x}(t)$ consisting of the values all the n nodes at t ($t = 0, 1, 2, \cdots$), written as $\mathbf{x}(t) = (x_1(t), x_2(t), \cdots, x_n(t))$. The *state transition* $\mathbf{x}(t) \to \mathbf{x}(t+1)$ is governed by the n Boolean functions. The n Boolean functions $f_1, f_2, \cdots, f_n$ in the BN constitute a vectorized function $\mathbf{f}$, called the *network function*, and $\mathbf{f} = (f_1, f_2, \cdots, f_n)$. A BN is said to be *homogeneous* if its network function does not vary with time and the state transitions are therefore deterministic. Assuming the Boolean functions are homogeneous (time-invariant) and the nodes are updated synchronously, we can write the Boolean functions in the simplified form by dropping $t+1$ and t in Equation 2.1. In the remaining text, we refer to BNs as both homogenous and synchronous.

Given a current state $\mathbf{x}(t)$, one can obtain the state at the next time step $\mathbf{x}(t+1)$ by evaluating the n Boolean functions. If one allows $\mathbf{x}(t)$ to be any one of the 2^n possible states, from $00\cdots0$ to $11\cdots1$, and computes their respective next states, then a list of 2^n one-step state transition pairs can be obtained. The 2^n state transitions can fully characterize a Boolean network's dynamics. For instance, if a Boolean network has 3 nodes, its one-step state transitions will have 8 pairs of states, where the first pair consists of state 000 and its next state, the second pair consists of 001 and its next state, etc.

Given all the one-step state transitions, a directed graph $T(V, E)$, known as the *state transition diagram*, can thus be constructed for the Boolean network. V is a set of 2^n vertices, each vertex being a state of the Boolean network. E is a set of 2^n edges, each pointing from a state to its next state in the pair-wise state transitions. If a state's next state is itself, then the edge is a loop.

An important characteristic of a Boolean network is, it possesses attractors. Starting from any initial state, after a finite number of state transitions, the network will reach a state or a set of states and remain there. The state or set of states that the network eventually settles into is called an *attractor*. The set of all states that eventually evolve into the same attractor constitutes the *basin of attraction* (BOA) for that attractor. A Boolean network may have more than one attractor. Different basins of attraction are depicted in the state transition diagram as disjoint subgraphs. An attractor consisting of one single state is called a *singleton attractor* (where we see a loop that points from a node to itself in the state transition diagram), and otherwise it is called an *attractor cycle* (a closed path in the state transition diagram). When Boolean networks are used to model genetic regulatory networks, attractors are often identified with phenotypes [13]. Real biological systems are typically assumed to have short attractor cycles, with singleton attractors being of special import.

An example of a state transition diagram with three attractors is shown in Fig. 1, where the graph is composed of three disjoint subgraphs, $T_1(V_1, E_1)$, $T_2(V_2, E_2)$ and $T_3(V_3, E_3)$. 110 and 101 are singleton attractors, while 100 and 111 constitute a cycle. Their respective basins of attraction are $V_1 = \{000, 010, 110\}$, $V_2 = \{101\}$ and $V_3 = \{001, 011, 100, 111\}$.

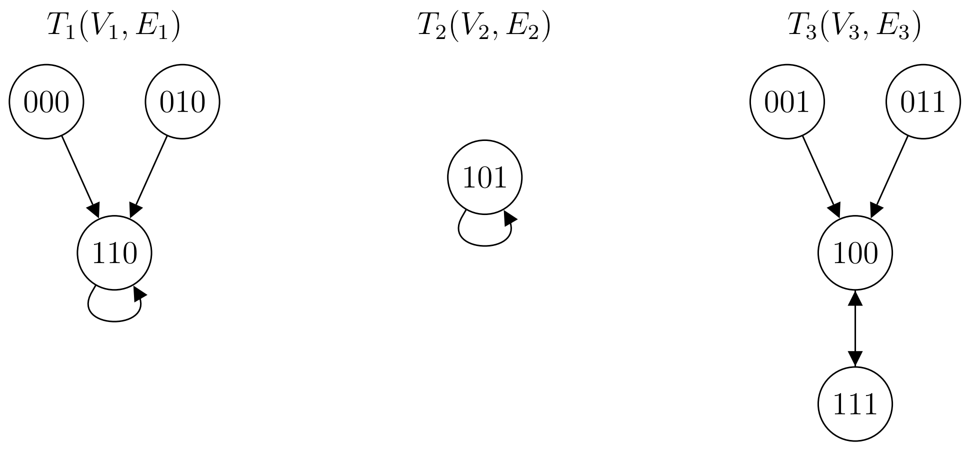


Fig. 1. State transition diagram of a Boolean network with 3 nodes.

B. Probabilistic Genetic Regulatory Networks

As is seen in the last section, Boolean network is a deterministic model. When stochasticity is introduced and genes are allowed to be quantized at more than two levels, a probabilistic genetic regulatory network model comes into the picture. A *probabilistic genetic regulatory network* $G_p(U, F, p, q)$ is defined on a set of U consisting of n nodes (genes), $x_1, x_2, \cdots, x_n$, each takeing values in a finite set M that contains d values ($d \leq 2$). The function set F consists of r network functions that govern the state transitions, $F = (\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_r)$. Each network function $\mathbf{f}_j$ is composed of n

functions $\psi_{j1}, \psi_{j2}, \cdots, \psi_{jn}$, and the value of the ith gene at time $t+1$ is given by

$$x_i(t+1) = \psi_{ji}(x_{i1}(t), x_{i2}(t), \cdots, x_{ik_i}(t)), \quad i = 1, 2, \cdots, n. \tag{2.2}$$

The choice of network function $\mathbf{f}_j$ is controlled by a selection procedure. Specifically, at each time point a random decision is made as to whether to switch the network function for the next transition, and the switch probability is q. If a decision is made to change the network function, then a new function is chosen from among $\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_r$, with the corresponding selection probabilities $c_1, c_2, \cdots, c_n$. A final aspect of the system is that at each time point there is a probability p of any gene changing its value uniformly randomly to another value in M. Since there are n genes, the probability of there being a random perturbation at any time point is $1-(1-p)^n$. 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 $(\mathbf{x}^i, \mathbf{f}_j)$. The random perturbation models random mutations in genes and makes the Markov chain ergodic, meaning that it has the possibility of reaching any state from another state and that it possesses a steady-state distribution.

When confined to the binary setting, the preceding description characterizes a probabilistic Boolean network (PBN). The state space in the probabilistic Boolean network is a n-dimensional binary vector space $S = \{0,1\}^n$, and each network function consists of a set of r Boolean functions that can be represented by truth tables. The Boolean setting simplifies the analysis but it is not restrictive since the analysis goes through for any finite valuation set. One can view a PBN as a collection of Boolean networks, each defined by a network function $\mathbf{f}_j$, with the switch probability q and the selection probabilities $c_1, c_2, \cdots, c_n$ controlling how the PBN switches between Boolean networks.

By definition, the attractors of a PBN are the attractors of its constituent BNs. When free of random gene flipping (p=0), the network will reach an attractor and remain there until a random switch of BN forces it to transit to another state outside of the current attractor. When $0 < q < 1$, the PBN is said to be context-sensitive, since it can switch between its constituent Boolean networks while being able to remain in a Boolean network for a period of time.

CHAPTER III

MODELING GENETIC REGULATORY NETWORKS VIA CONTEXT-SENSITIVE PROBABILISTIC BOOLEAN NETWORKS*

A. Introduction

The modeling of genetic regulatory networks holds potential for gaining a deep understanding of biological processes and for developing effective therapeutic intervention in human diseases such as cancer. This inevitably entails using computational and formal methods to understand general principles governing the system under study and to make useful predictions about system behavior in the presence of known conditions. A number of modeling approaches have been considered. Here we are interested in graphical models, of which, two of the most studied are Boolean networks [20, 21, 23, 13, 14, 24] and Bayesian networks [25, 26, 27].

Probabilistic Boolean networks represent an interface between the absolute determinism of Boolean networks and the probabilistic nature of Bayesian networks, in that they incorporate rule-based uncertainty [22, 28]. This compromise is important because rule-based dependencies between genes are biologically meaningful, while mechanisms for handling uncertainty are conceptually and empirically necessary. The binary (Boolean) nature of PBNs has been assumed so as to model ON-OFF

*© 2006 IEEE. Reprinted from *IEEE Transactions on Signal Processing*, see [10] for complete publication information.

switching behavior, but their structure extends easily to any discrete (multi-valued) setting, thereby yielding a general framework for probabilistic genetic regulatory networks (PGRNs) — in particular, for ternary cDNA data [29, 30]. The dynamics of these networks can be studied in the probabilistic context of Markov chains, thereby facilitating steady-state analysis [31]. The dynamical properties of PBNs have been studied to consider the effect of individual genes on global dynamical network behavior, both from the view of random gene perturbation as well as intervention to elicit desired network behavior [32, 33]. It has been shown that PBNs offer the potential to design treatment strategies based on the application of external control variables to drive network dynamics [34, 35].

A key issue in network modeling is design (inference) of the network from data [36, 37, 38, 39, 40, 41, 42]. Network connectivity and transition rules must be inferred from the data, with perhaps the imposition of biological constraints [43]. When building function-based gene networks from expression data, the functions are typically derived via some optimization-based criterion. This requires determining, for each gene g, the genes that will serve as input to the function giving the value of g and the structure of the function. The basic method proposed for PBNs is based on the coefficient of determination [44, 28], and has been the one most used. Two other methods have been proposed for PGRN design based on multivariate nonlinear prediction and Markov chain Monte Carlo predictor design. One utilizes information-theoretic gene clustering to find input genes and a two-layer perceptron [30], and the other employs Bayesian gene selection and multinomial probit regression [45].

Except in rare circumstances, the optimal function for a gene will not be a perfect predictor because there will be inconsistencies in the data. This means that a specific vector of values for a set of regulatory genes will not necessarily correspond to a single value of the target gene. Thus, network design is inherently probabilistic. In this chapter we wish to model these inconsistencies in a way that reflects context changes in genomic regulation. The network can be in any of a number of contexts. Within a context, the network behaves deterministically and the generated data is consistent. If a regulatory set takes on a specific vector of values, then the target gene associated with the regulatory set must take on a single value, and this is reflected in the data.

Overall, we propose an inference procedure for PBNs whose contexts model the data in such a way that they are consistent for each context, the intent being to view data inconsistencies as being due to latent variables. Separate sections are dedicated to data-consistent inference, data-consistent operator design, and data-consistent PBN design. We follow these with a discussion of the relationship between standard and data-consistent designs, the role of data filtering, application to a melanoma-related network, and some concluding remarks.

B. Inference and the Issue of Data Consistency

For the most part, PBN inference has been based on classical binary optimization, where the predictor variables for each target gene have been selected using the coefficient of determination (CoD). The CoD measures the degree to which the best estimate for the value (transcriptional activity) of a target gene can be improved using the knowledge of the values of a set of predictor genes, relative to the best estimate in the absence of any knowledge of the predictors:

$$CoD = \frac{\epsilon_0 - \epsilon_{opt}}{\epsilon_0}. \tag{3.1}$$

In 3.1, ϵ_0 is the error arising when using the best estimate of the target-gene expression level given only statistics relating to the target gene itself, without using any information concerning other genes, and ϵ_{opt} is the error arising using the best estimate of the target-gene expression level using the expression levels of a set of predictor genes. In general, $0 \leq CoD \leq 1$. If a predictor set can perfectly predict a target gene, then $\epsilon_{opt} = 0$ and $CoD = 1$; at the other extreme, if a predictor set provides no information about the target gene, then $\epsilon_{opt} = \epsilon_0$ and $CoD = 0$.

If we fix ahead the number of predictor genes (referred to as the *connectivity*) that can compose a regulatory set for a target gene, then the design method is to choose the regulatory set with the largest CoD and then define a binary regulatory function based on the genes in the regulatory set. For instance, suppose genes g_1, g_2, and g_3 have the highest collective CoD among all triples for predicting gene g. Let $wxyz$ denote a binary vector of values for (g_1, g_2, g_3, g). If 0001 appears more often in the data than 0000, then for $xyz = 000$ the predictor function is defined by $\psi(000) = 1$; otherwise, it is defined by $\psi(000) = 0$ (ties being broken either by convention or randomly). If both 0000 and 0001 appear in the data, then the data is inconsistent relative to predicting g via g_1, g_2, and g_3.

Inconsistency means that the data is interpreted in such a way that the predictor is a random function: the same values of the predictors can yield different values of the target. This interpretation is problematic under the assumption that biological regulation is deterministically encoded in the genes. There are possible reasons for inconsistent data that are not inherent to the network. First, the data could be noisy. We will not consider this issue here, but plan to address it in a subsequent study in the context of a noise model, which is the only way to address it in a rigorous mathematical framework. Second, it could be that the predictor set is incomplete. For instance we might have 0000 and 0001 in the data for genes g_1, g_2, g_3, and g, but had we considered genes g_0, g_1, g_2, g_3, and g, the observations would have been 00000 and 10001, which would have eliminated the inconsistency in predicting g. In practice, we limit the number of predictors owing to the exponentially increasing data demand as the number of predictors is increased. Here, however, to avoid such dimensionality inconsistencies, we will assume that all genes other than the target compose the regulatory set for the target gene. Once the predictor is designed, we can drop nonessential variables. For instance, if for $\psi(wxy) = z$, $\psi(0xy) = \psi(1xy)$ for all xy, then w can be dropped as a predictor gene for z.

In this chapter, we will address an inherent problem that leads to inconsistency. Consider a network with two contexts, C_0 and C_1. If the regulatory genes g_1, g_2, and g_3 form the vector 001 in context C_0, then their target gene g must take on a specific value, say 0, in C_0. This uniqueness condition holds for all vectors of values for g_1, g_2, and g_3. It may well be that in context C_1 the regulatory genes take the vector 001 while gene g has value 1, but the data is consistent so long as a single context is maintained. Unless the contexts are known when data from the network is sampled, it would appear that the network is not operating consistently. Since the context is generally not known, an experiment to predict g when $g_1g_2g_3 = 001$ is likely to yield n_0 and n_1 observations of 0 and 1, respectively, meaning that 0010 and 0011 for genes g_1, g_2, g_3 and g have been observed n_0 and n_1 times in contexts C_0 and C_1, respectively. The regulatory function ψ for g would then be defined for 001 by $\psi(001) = 0$ if $n_0 > n_1$ and $\psi(001) = 1$ if $n_1 > n_0$, with some convention determining $\psi(001)$ if $n_1 = n_0$.

Here we take a different approach. If the data reveal two values for a target gene for a single vector for the regulatory set, then we will construct the network in such a way that there are two distinct functions, ψ_0 and ψ_1, such that $\psi_0(001) = 0$ and $\psi_1(001) = 1$. The two functions represent two different network contexts. The probabilities of the two functions being selected for regulation will be in agreement with the context probabilities (in a manner shortly to be defined).

Conceptually, the regulatory action is viewed as a system with inputs corresponding to the regulating genes for the target gene; however, the system is not fully described by the input gene values alone, but by these inputs in conjunction with the context. Biologically, the context is determined by the manner in which the genes are responding to *latent variables* external to the model network. Together, the latent variables act in a manner as to *select* a network (system) context. One can imagine a set of input lines entering the overall system, within the system there being a family of subsystems (contexts), and the system output being a single line whose information is selected from among the subsystems. This would be the structure of a computer system whose output is determined by a multiplexor, with the multiplexor's decision being determined by a selection input to it. Biologically, only a single subsystem may be operative at any given time, but mathematically it is irrelevant whether we assume that a single regulatory function operates in a given selected context or that all regulatory functions operate and a single output is selected from among these.

From an engineering perspective, we are not concerned with the actual mechanisms of a system, but only the manner in which it transforms input signals to output signals. A similar statement applies to subsystems. Hence, by definition, a context is represented by a subsystem, which is itself a collection of mathematical functions. Since the context is selected by external variables, we cannot know deterministically when the system is in a certain context, but we can infer the probability of the system being in a particular context from the data. Our basic criterion for network design is that the distribution of expected state observations for the system, if it is observed over a long period of time, agrees with the observed distribution of states for the data. As for consistency, that holds *ipso facto* because the system behaves deterministically so long as it remains in a fixed context (i.e., it is determined by the unique set of functions defining that context).

Recalling the definition of a PGRN, one sees that such networks are defined in accordance with context changes, each context being characterized by a network function; however, heretofore PGRNs have been defined through the construction of network functions by choosing several strong predictor functions for each gene and forming each network function by choosing a strong predictor for each gene. Such an approach is not in accord with an assumption of data consistency. The inference methodology discussed in this chapter is in accord with it.

Owing to their significance, it is especially important that attractors are properly modeled in an inferred PBN. If the switching and perturbation probabilities are very small, which is typical if the network is sufficiently self-contained not to be subject to frequent latent-variable effects, then it behaves as a single Boolean network for long periods of time. As a result, it spends the vast majority of its time in attractors. In most experimental situations, unless a situation has been created where time-course gene expression measurements are taken following some stimulus to the system that drives it out of its steady-state behavior, the typical assumption is that measurements (or at least almost all of them) are taken in the steady state [29]. This assumption has two immediate implications for inference. First and most importantly, since data states are, with probability near one, attractor states, we would like them to be attractors in the model. According to Proposition 1 (to be seen later in this chapter), this is fully accomplished with the proposed inference procedure. For small samples, it is very possible that sampling misses biological attractor states in the data; however, with large samples the likelihood grows for observing biological attractor states in the data, and therefore incorporating them in the model. This is precisely what one would expect in a learning paradigm. As for the converse of the first implication, while network design can result in non-data states being attractor states in the model, Propositions 2 and 4 show that in a number of cases a non-data state will not be an attractor. In addition, as might be expected in a learning environment, avoiding non-data states as attractors depends on design generalization beyond that immediately implied by the data.

1. Operator Design under the Requirement of Data Consistency

Since the key to network design is designing the functions, we begin by treating data consistency in the more general framework of designing a single Boolean operator on random inputs. Let $S = \{\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^m\}$ be the set of $m = 2^n$ vectors associated with the binary-valued observation variables $X_1, X_2, \cdots, X_n$ and let Y be a target binary random variable to be predicted via $X_1, X_2, \cdots, X_n$. A data set D composed of observations of the form $(\mathbf{x}^k, y)$ is said to be *consistent* if $(\mathbf{x}^k, 0)$ and $(\mathbf{x}^k, 1)$ are not both in D. Going the other way, a random predictor-target pair $(\mathbf{X}, Y)$ is said to be consistent with the data set D if D is consistent relative to the observation pairs resulting from $(\mathbf{X}, Y)$. In such as case, there exists a predictor ψ for Y via $\mathbf{X}$, defined on S, possessing zero error on the data. ψ is said to be consistent relative to D. ψ may not be unique, since for any vector $\mathbf{x}^k$ for which neither $(\mathbf{x}^k, 0)$ nor $(\mathbf{x}^k, 1)$ appears in the data, ψ can be defined arbitrarily.

Consider a random operator Ψ on S. Every realization ψ of Ψ defines a function on the random vector $\mathbf{X}$, or, equivalently, on the state space S endowed with the probability measure corresponding to $\mathbf{X}$. If D is any data set generated by ψ, then, *ipso facto*, ψ is consistent relative to D. The number of observations in the data corresponding to any vector $\mathbf{x}^k$ is related to the probability of $\mathbf{x}^k$ in S, not ψ. Specifically, letting $\nu(\mathbf{x}^k)$ denote the number of observations of $\mathbf{x}^k$ in an arbitrary data set of size N, then $E[\nu(\mathbf{x}^k)] = N\pi(\mathbf{x}^k)$, where $\pi(\mathbf{x}^k)$ is the probability of $\mathbf{x}^k$ in S. In accordance with the empirical distribution of $\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^m$ for D, define the probability measure π_D on S by $\pi_D(\mathbf{x}^k) = \nu_D(\mathbf{x}^k)/N$, where $\nu_D(\mathbf{x}^k)$ is the number of observations of $\mathbf{x}^k$ in D. Then $\pi_D(\mathbf{x}^k)$ is an estimate of $\pi(\mathbf{x}^k)$. A key to operator design is the following observation: if ψ_0 and ψ_1 are two realizations of Ψ and they agree on all vectors except $\mathbf{x}^i$, for which $\psi_0(\mathbf{x}^i) = 0$ and $\psi_1(\mathbf{x}^i) = 1$, then both pairs $(\mathbf{x}^i, 0)$ and $(\mathbf{x}^i, 1)$ may lie in a data set generated by ψ_0 and ψ_1, but the data will be consistent for all $\mathbf{x}^k \neq \mathbf{x}^i$.

Case 1 : Suppose the data set D has the property that there is a single vector, $\mathbf{x}^i$, possessing different Y values, and all other vectors possess a single Y value in D. Suppose there are $\nu_D(\mathbf{x}^i, 0)$ and $\nu_D(\mathbf{x}^i, 1)$ pairs $(\mathbf{x}^i, 0)$ and $(\mathbf{x}^i, 1)$, respectively. Define two functions, ψ_0 and ψ_1, that agree on all vectors except $\mathbf{x}^i$, and are thereon defined by $\psi_0(\mathbf{x}^i) = 0$ and $\psi_1(\mathbf{x}^i) = 1$. Let Ψ_D be a random function possessing two realizations ψ_0 and ψ_1. We define the probability structure for Ψ_D by

$$P(\Psi_D = \psi_a) = \frac{\nu_D(\mathbf{x}^i, a)}{\nu_D(\mathbf{x}^i)}$$

with $a = 0, 1$. ψ_0 is consistent relative to the data set $D_i(0)$ consisting of the original data set D with all pairs $(\mathbf{x}^i, 1)$ removed, and ψ_1 is consistent relative to the data set $D_i(1)$ consisting of D with all pairs $(\mathbf{x}^i, 0)$ removed.

Case 2 : Suppose the data set D has the property that there exist two vectors, $\mathbf{x}^i$ and $\mathbf{x}^j$, possessing different Y values, and all other vectors possess a single Y value in D. Let there be $\nu_D(\mathbf{x}^i, 0)$, $\nu_D(\mathbf{x}^i, 1)$, $\nu_D(\mathbf{x}^j, 0)$, and $\nu_D(\mathbf{x}^j, 1)$ pairs of $(\mathbf{x}^i, 0)$, $(\mathbf{x}^i, 1)$, $(\mathbf{x}^j, 0)$, and $(\mathbf{x}^j, 1)$, respectively. Define four functions ψ_{00}, ψ_{01}, ψ_{10}, and ψ_{11} that agree on all vectors except $\mathbf{x}^i$ and $\mathbf{x}^j$, and are thereon defined by $\psi_{00}(\mathbf{x}^i) = 0$, $\psi_{00}(\mathbf{x}^j) = 0$, $\psi_{01}(\mathbf{x}^i) = 0$, $\psi_{01}(\mathbf{x}^j) = 1$, $\psi_{10}(\mathbf{x}^i) = 1$, $\psi_{10}(\mathbf{x}^j) = 0$, $\psi_{11}(\mathbf{x}^i) = 1$, and $\psi_{11}(\mathbf{x}^j) = 1$. Define the following probability structure for Ψ:

$$P(\Psi_D = \psi_{ab}) = \frac{\nu_D(\mathbf{x}^i, a)\nu_D(\mathbf{x}^j, b)}{\nu_D(\mathbf{x}^i)\nu_D(\mathbf{x}^j)}, \tag{3.2}$$

for $a, b = 0, 1$. For instance, $P(\Psi_D = \psi_{00}) = \nu_D(\mathbf{x}^i, 0)\nu_D(\mathbf{x}^j, 0)/\nu_D(\mathbf{x}^i)\nu_D(\mathbf{x}^j)$. ψ_{ab} is consistent relative to the data set $D_{ij}(ab)$ consisting of the original data set D with all pairs $(\mathbf{x}^i, 1-a)$ and $(\mathbf{x}^j, 1-b)$ removed.

Case k : The preceding definition and probability structure can be inductively defined for any k vectors possessing different Y values, with all the other vectors possessing a single Y value. We say that the resulting random function is *order-k consistent* relative to the data set D.

We now state the basic theorem for consistent-data operator design.

Theorem 1 *If the random function Ψ_D is order-k consistent relative to the set D, then (1) when restricted to any of its realizations, Ψ_D produces consistent data, (2) the estimate of the expected distribution of the data generated by Ψ_D using π_D in place of π agrees with the distribution of the data in D, and (3) the latter condition cannot be accomplished with less than 2^k functions, the number of realizations of Ψ_D.*

Proof. We first prove the case 1. For a random data set $\mathcal{D}$ of size N generated by the random function Ψ_D, let $\eta(\mathbf{x}^i, 0)$ and $\eta(\mathbf{x}^i, 1)$ be the random variables giving the number of times $\mathbf{x}^i$ is 0 and 1, respectively, in $\mathcal{D}$. Since Ψ_D is designed from the given data set D and thereafter applied to random data sets, the probability $P(\Psi_D = \psi_0)$ is fixed upon the design of Ψ_D and is independent of the probability of observing any particular state vector in $\mathcal{D}$. Thus,

$$\begin{aligned}
E[\eta(\mathbf{x}^i, 0)] &= NP(\Psi_D(\mathbf{x}^i) = 0)\pi(\mathbf{x}^i) \\
&= N\pi(\mathbf{x}^i)[P(\Psi_D = \psi_0)P(\psi_0(\mathbf{x}^i) = 0) + P(\Psi_D = \psi_1)P(\psi_1(\mathbf{x}^i) = 0)] \\
&= NP(\Psi_D = \psi_0)\pi(\mathbf{x}^i) \\
&= N\frac{\nu_D(\mathbf{x}^i, 0)}{\nu_D(\mathbf{x}^i)}\pi(\mathbf{x}^i).
\end{aligned}$$

If we replace $\pi(\mathbf{x}^i)$ by its estimate $\pi_D(\mathbf{x}^i)$ based upon the data set D, then we obtain the estimate

$$\hat{E}[\eta(\mathbf{x}^i, 0)] = \nu_D(\mathbf{x}^i, 0). \tag{3.3}$$

of the expectation $E[\eta(\mathbf{x}^i, 0)]$. Equation 3.3 states that the estimate of the expectation of the number of times that $\mathbf{x}^i$ has the label 0, based on the estimate π_D equals the number of times $\mathbf{x}^i$ has the label 0 in the data. Similarly,

$$\hat{E}[\eta(\mathbf{x}^i, 1)] = \nu_D(\mathbf{x}^i, 1). \tag{3.4}$$

For $j \neq i$, $\hat{E}[\eta(\mathbf{x}^j, 0)]$ is either 0 or $\nu_D(\mathbf{x}^j)$, depending on the common value of $\psi_0(\mathbf{x}^j)$ and $\psi_1(\mathbf{x}^j)$. In sum, when restricted to either ψ_0 or ψ_1, the random function produces consistent data, and the expected distribution of the data agrees perfectly with the empirical distribution π_D. Clearly, this could not have been accomplished by a single realization.

For case 2, for an arbitrary data set $\mathcal{D}$ of size N generated by Ψ_D, let $\eta(\mathbf{x}^i, 0)$, $\eta(\mathbf{x}^i, 1)$, $\eta(\mathbf{x}^j, 0)$, and $\eta(\mathbf{x}^j, 1)$ be random variables giving the number of times $\mathbf{x}^i$ is 0, $\mathbf{x}^i$ is 1, $\mathbf{x}^j$ is 0, and $\mathbf{x}^j$ is 1, respectively, in $\mathcal{D}$. Then

$$\begin{aligned}
E[\eta(\mathbf{x}^i, 0)] &= NP(\Psi_D(\mathbf{x}^i) = 0)\pi(\mathbf{x}^i) \\
&= N\pi(\mathbf{x}^i)[P(\Psi_D = \psi_{00})P(\psi_{00}(\mathbf{x}^i) = 0) + P(\Psi_D = \psi_{01})P(\psi_{01}(\mathbf{x}^i) = 0) \\
&\quad + P(\Psi_D = \psi_{10})P(\psi_{10}(\mathbf{x}^i) = 0) + P(\Psi_D = \psi_{11})P(\psi_{11}(\mathbf{x}^i) = 0)] \\
&= N\pi(\mathbf{x}^i)[P(\Psi_D = \psi_{00}) + P(\Psi_D = \psi_{01})] \\
&= N\pi(\mathbf{x}^i)\left(\frac{\nu_D(\mathbf{x}^i, 0)\nu_D(\mathbf{x}^j, 0)}{\nu_D(\mathbf{x}^i)\nu_D(\mathbf{x}^j)} + \frac{\nu_D(\mathbf{x}^i, 0)\nu_D(\mathbf{x}^j, 1)}{\nu_D(\mathbf{x}^i)\nu_D(\mathbf{x}^j)}\right).
\end{aligned} \tag{3.5}$$

If we replace $\pi(\mathbf{x}^i)$ by its estimate $\pi_D(\mathbf{x}^i)$ based on the data set D, then we obtain the estimate of the expectation $E[\eta(\mathbf{x}^i, 0)]$,

$$\hat{E}[\eta(\mathbf{x}^i, 0)] = N\pi(\mathbf{x}^i, 0)\left(\frac{\nu_D(\mathbf{x}^j, 0)}{\nu_D(\mathbf{x}^j)} + \frac{\nu_D(\mathbf{x}^j, 1)}{\nu_D(\mathbf{x}^j)}\right) = \nu_D(\mathbf{x}^i, 0). \tag{3.6}$$

Similarly, $\hat{E}[\eta(\mathbf{x}^i, 1)] = \nu_D(\mathbf{x}^i, 1)$, $\hat{E}[\eta(\mathbf{x}^j, 0)] = \nu_D(\mathbf{x}^j, 0)$, and $\hat{E}[\eta(\mathbf{x}^j, 1)] = \nu_D(\mathbf{x}^j, 1)$. For $l \notin \{i, j\}$, $\hat{E}[\eta(\mathbf{x}^l, 0)]$ is either 0 or $\nu_D(\mathbf{x}^l)$, depending on the common value of $\psi_0(\mathbf{x}^l)$ and $\psi_1(\mathbf{x}^l)$ for l. In sum, when restricted to either ψ_{00}, ψ_{01}, ψ_{10}, or ψ_{11}, the estimate of the expected distribution of the data, using the estimate π_D, agrees with the data distribution. This cannot be accomplished with less than four functions. Indeed, since any function must agree with the single value for vectors other than $\mathbf{x}^i$ and $\mathbf{x}^j$, were there only three functions, these would be a subset of $\{\psi_{00}, \psi_{01}, \psi_{10}, \psi_{11}\}$ and there would still be four equations of the kind in Eq. 3.5. These would require solution with only three variables of kind $P(\Psi_D = \psi_{ab})$ instead of the four variables $P(\Psi = \psi_{00})$, $P(\Psi = \psi_{01})$, $P(\Psi_D = \psi_{10})$, and $P(\Psi_D = \psi_{11})$.

The proof for case 2 extends directly to any order k, albeit, with increased notational complexity. ∎

The third part of the theorem is critical because it says that the constructed random function solves the problem with which we are concerned in an optimal way relative to minimizing the number of its realizations. By addressing data inconsistency under the assumption that inconsistencies result from the data arising from a random function of the state space, optimal operator design becomes one of finding the realizations of a random function and the probability mass on those realizations so that the resulting random operator best fits the data relative to the expectation of its output and does so using a minimal number of randomizations. In effect, we have presented an algorithm to solve this optimization problem.

To illustrate the design methodology, we consider two predictor variables, X and Y, the target variable Z, and the data in Table I(a), where *count* is the number of

Table I. Data Set and Predictor Functions

(a) Data set

xyz	000	001	010	011	100	101	110	111
Count	4	0	6	6	2	6	0	4

(b) Function for z

xy	ψ^z_{00}	ψ^z_{01}	ψ^z_{10}	ψ^z_{11}
00	0	0	0	0
01	0	0	1	1
10	0	1	0	1
11	1	1	1	1

(c) Function for y

xz	ψ^y_{00}	ψ^y_{01}	ψ^y_{10}	ψ^y_{11}
00	0	0	1	1
01	1	1	1	1
10	0	0	0	0
11	0	1	0	1

(d) Function for x

yz	ψ^z_{00}	ψ^z_{01}	ψ^z_{10}	ψ^z_{11}
00	0	0	1	1
01	1	1	1	1
10	0	0	0	0
11	0	1	0	1

times xyz is observed in the data. In the data, the observations 00 and 11 of the predictor variables are consistent, whereas 01 and 10 of the predictor variables are inconsistent. Hence, four functions are required to predict Z, as shown in Table I(b). The selection probabilities are $P(\Psi = \psi_{00}^z) = 1/8$, $P(\Psi = \psi_{01}^z) = 3/8$, $P(\Psi = \psi_{10}^z) = 1/8$, and $P(\Psi = \psi_{11}^z) = 3/8$. Notice what happens if we change the count of 111 to 0. The number of functions remains 4; however, the data does not provide inference of $\psi_{ab}^z(11)$. Therefore, it must be decided by some form of *generalization*. We will return to this question in the next chapter.

It is important in understanding Theorem 1 to recognize that the third part of the theorem refers to the second part, that is, the number of realizations required to accomplish the distributional requirement is 2^k. If we were not concerned with the expected concordance between the expected distribution of data generated by the random function Ψ_D and the distribution of the data in D, then we would need only two realizations to achieve consistent design. To see this, suppose in D there exist m vectors, $\mathbf{x}^{i_1}, \mathbf{x}^{i_2}, \cdots, \mathbf{x}^{i_m}$ possessing different Y values and for any other vector $\mathbf{x}$ there is a single observed Y value $a_{\mathbf{x}}$. Define $\psi_0(\mathbf{x}^{i_j}) = 0$ and $\psi_1(\mathbf{x}^{i_j}) = 1$ for $j = 1, 2, \cdots, m$, and $\psi_0(\mathbf{x}) = \psi_1(\mathbf{x}) = a_{\mathbf{x}}$ for any other $\mathbf{x}$. These two realizations can account for all of the inconsistencies; however, the expected distribution of data generated by Ψ_D will not be concordant with the data distribution in D.

2. Data-Consistent Design of Probabilistic Boolean Networks

Adaptation of the consistent-data predictor design to PBNs is straightforward, and the details of algorithm can be found in Appendix A. There are some issues regarding generalization and attractors that need to be addressed. Consider designing a PBN from a data set for the set $S = \{\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^m\}$ of $m = 2^n$ binary vectors. For a PBN, each gene is taken in turn as the target to be predicted via the remaining genes by a predictor function. This means that the consistent-data design procedure is applied to each gene in turn to derive its predictor set. A network function for the PBN is defined by taking one predictor function for each gene. For a network with n genes, if there are m_k predictors for gene k, then there are $m_1 m_2 \cdots m_n$ network functions. Each network function defines a *context* of the network in which the data are consistent. This means that, so long as a network is in the context of a network function, it will generate consistent data. Each context defines a standard (constituent) Boolean network. The selection probability of a network function is the product of the selection probabilities for the individual functions composing the network function.

To illustrate, for the data of Table I(a), we have three function sets shown in parts (b), (c), and (d). For both xz and yz, the observations 01 and 10 are consistent, whereas 00 and 11 are inconsistent. The PBN has 64 network functions determining the same number of contexts. The number of contexts is determined by the manner in which inconsistencies appear in the data.

Attractors are important to understanding a PBN. Each context corresponds to a Boolean network, and by definition the attractors of the PBN are the attractors of its constituent Boolean networks. Relative to attractors, there is a fundamental difference between data states and non-data states. Before giving formal definitions, we consider some possible situations.

For the PBN resulting from the data of Table I(a), consider the data state 000. It is a singleton attractor for any context $\{\psi^x_{ab}, \psi^y_{cd}, \psi^z_{ef}\}$ in which $\psi^x_{ab}(00) = \psi^y_{cd}(00) = \psi^z_{ef}(00) = 0$. There are $2 \times 2 \times 4 = 16$ such contexts (out of a total of 64 contexts). Running through the six data states, we see that each is a singleton attractor for some number of contexts. On the contrary, consider the non-data state 001. Since $\psi^x_{ab}(01) = 1$, $\psi^y_{cd}(01) = 1$, and $\psi^z_{ef}(00) = 0$ for any ab, cd, and ef, $001 \rightarrow 110$,

Table II. Predictor Functions

yz	ψ^x
00	0
01	x
10	x
11	x

xz	ψ^y
00	0
01	x
10	x
11	x

yz	ψ^z
00	0
01	x
10	x
11	x

its complement, in every context. 110 is also a non-data state, and 110 $\rightarrow$ 001, its complement, in every context. Hence, $\{110, 001\}$ is a two-state attractor cycle in every context. Note that if 110 were a data state, then it would be a singleton attractor in some contexts and the non-data state 001 would not be an attractor (in an attractor cycle) in those contexts.

Now, consider a data set in which there is a single data state, say 000. All predictor-target pairs are consistent relative to the data, and only one function is required for each gene (Table II). Each function requires three of its four values to be determined by generalization (arbitrarily relative to the data). The result is a Boolean network in which 000 is a singleton attractor.

We say that a non-data state $\mathbf{x} = x_1x_2 \cdots x_n$ is *partially mapped* by the data if there exists at least one sub-vector, $x_1x_2 \cdots x_{k-1}x_{k+1} \cdots x_n$, which has been observed in the data, so that there exists a function ψ_k for x_k for which $\psi_k(x_1x_2 \cdots x_{k-1}x_{k+1} \cdots x_n)$ has been determined by the data, not by generalization. For the single observation 000 and the Boolean network of Table II, the states 001, 010, and 100 are partially mapped. A non-data state $\mathbf{x} = x_1x_2 \cdots x_n$ is *fully unmapped* by the data if no sub-vector $x_1x_2 \cdots x_{k-1}x_{k+1} \cdots x_n$ has been observed in the data. For the single observation 000, the states 011, 101, 110, and 111, are fully unmapped. A non-data state is *fully mapped* if all sub-vectors have been observed in the data, which was the case for 110 in the data of Table I(a).

Continuing with the single observation 000 and the network of Table II, for which the non-data states 001, 010, and 100 are partially mapped by the data, the single network function yields 001 $\rightarrow$ xx0, 010 $\rightarrow$ x0x and 100 $\rightarrow$ 0xx. The actual transitions depend on the generalization; nevertheless, these partially determined non-data states are not singleton attractors. The remaining data states, 011, 101, 110, and 111, are fully unmapped by the data, so that their transitions depend totally on generalization, which can yield singleton non-data attractors. In this example, 011 becomes a singleton attractor if and only if we define $\psi^x(11) = 0$, $\psi^y(01) = 1$, and $\psi^z(01) = 1$; 101 becomes a singleton attractor if and only if we define $\psi^x(01) = 1$, $\psi^y(11) = 0$, and $\psi^z(10) = 1$; 110 becomes a singleton attractor if and only if we define $\psi^x(10) = 1$, $\psi^y(10) = 1$, and $\psi^z(11) = 0$; and 111 becomes a singleton attractor if and only if we define $\psi^x(11) = 1$, $\psi^y(11) = 1$, and $\psi^z(11) = 1$. Note that 110 and 111 cannot simultaneously be singleton attractors, nor can 011 and 111 simultaneously be singleton attractors.

We now provide some formal propositions regarding attractors.

Proposition 1 *A data state is a singleton attractor in at least one context.*

Proof. If $\mathbf{x} = x_1x_2\cdots x_n$ is a data state, then for each gene x_k, there is at least one function, ψ^k, inferred from the data for which $\psi^k(x_1x_2\cdots x_{k-1}x_{k+1}\cdots x_n) = x_k$. $\mathbf{x}$ is a singleton attractor for the context $\{\psi^1, \psi^2, \cdots, \psi^n\}$. ∎

Proposition 2 *A fully or partially mapped non-data state is not a singleton attractor in any context.*

Proof. If $\mathbf{x} = x_1x_2\cdots x_n$ is a fully or partially mapped non-data state, then there exists a gene x_k determined from the data relative to $x_1x_2\cdots x_n$. Suppose $x_1x_2\cdots x_n \rightarrow$

$x_1x_2\cdots x_n$ in some context $\{\psi^1, \psi^2, \cdots, \psi^n\}$. Then $x_k = \psi^k(x_1x_2\cdots x_{k-1}x_{k+1}\cdots x_n)$. Since this relationship has been determined from the data, $x_1x_2\cdots x_n$ must be a data state, which is a contradiction. ∎

Proposition 3 *If a non-data state and its complement are both fully mapped, then they form a two-state attractor cycle in every context.*

Proof. If $\mathbf{x} = x_1x_2\cdots x_n$ is fully mapped, then in any context $\{\psi^1, \psi^2, \cdots, \psi^n\}$,

$$\mathbf{x} \to \psi^1(x_1x_2\cdots x_n)\psi^2(x_1x_3\cdots x_n)\cdots\psi^n(x_1x_2\cdots x_{n-1})$$

It must be that $\psi^k(x_1x_2\cdots x_{k-1}x_{k+1}\cdots x_n) = x_k^c$, since otherwise the fact that $x_1x_2\cdots x_{k-1}x_{k+1}\cdots x_n$ has been observed in the data would mean that $x_1x_2\cdots x_n$ has been observed in the data, which it has not. Hence $\mathbf{x} \to \mathbf{x}^c$. The same argument applied to $\mathbf{x}^c$ shows that $\{\mathbf{x}, \mathbf{x}^c\}$ is a two-state attractor. ∎

Proposition 4 *Generalization can always make a given fully unmapped non-data state be or not be a singleton attractor.*

Proof. If $\mathbf{x} = x_1x_2\cdots x_n$ is a fully unmapped non-data state, then there are no data-determined functions $\psi^k(x_1x_2\cdots x_{k-1}x_{k+1}\cdots x_n)$. To make $\mathbf{x}$ an attractor, define $\psi^k(x_1x_2\cdots x_{k-1}x_{k+1}\cdots x_n) = x_k$ for all k; to make $\mathbf{x}$ not a singleton attractor, define ψ^k in any other manner. ∎

Any attractor composed solely of non-data states will be called an artificial attractor. As we have noted previously, it may not be possible to make two fully unmapped non-data states into singleton attractors. According to Proposition 2, artificial singleton attractors are fully unmapped. Every singleton attractor is either a data state or an artificial attractor. According to Proposition 3, if a non-data state and its complement are both fully mapped, then they form an artificial two-state attractor cycle in every context.

The state transitions for a PBN produce an ergodic Markov chain possessing a steady-state distribution. When a PBN is designed from data, the implicit assumption is that the data have been obtained in the steady state. This means that the state transitions of the designed PBN do not correspond to transitions in biological time but to synthetic (mathematical) time. Hence, there is no direct correspondence between transient states of the PBN and data states. There should be, however, correspondence between steady-state behavior and the data states. Since we expect network switching to be infrequent in a real system, most of the steady-state mass should belong to the attractors, and since the data has been drawn from the steady state, we would expect it to be highly likely that the data states are attractors. In this sense, Proposition 1 provides strong support for the context-switching model. Proposition 2 is also encouraging relative to steady-state and data distribution correspondence. Propositions 3, while not encouraging, posits the strong requirement that a non-data state to be fully mapped. Finally, Proposition 4 only asserts existence and says nothing about the consequences of a reasonable generalization.

C. Discussion

1. Reflections on Standard and Contextual Inference

Whereas a Boolean network is assured for a single observed data state, two data states may require a PBN. At the other extreme, only a Boolean network is required for consistency if the data states are 001, 010, 100, and 111, and all four states would be singleton attractors. The issue of the number of contexts is related to a deeper issue of how we have chosen to use the data for inference, not only here but in previous papers. At its root, the matter concerns learning predictors for a dynamical system from steady-state data.

To explain, we consider a three-gene Boolean network with vectors xyz and data set $\{000, 001\}$. If we observe 000 more often than 001, why define the prediction $\psi^z(00) = 0$? After all, in the real system, 000 might transition to another state, and therefore $xy = 00$ may predict z being 1. For instance, if in the actual system $000 \rightarrow 001$, then would it not be better to predict z by $\psi^z(00) = 1$? Perhaps it would be had we dynamical data to indicate so, but we only have steady-state data. The original use of prediction for gene expression was to measure multivariate gene interaction [16]: based on the data, if $xy = 00$ is observed in the steady-state, then what is the best prediction for z. The prediction methodology is purely statistical and makes no inference regarding causality. Clearly, if we observe 000 in the data more often than 001, then the best prediction on observing $xy = 00$ in a future observation would be to predict $z = 0$. This approach has been adopted for network inference, and represents a kind of generalization because a network involves dynamical behavior. Nonetheless, under the assumption that the data come from the steady state, and assuming that when in the steady state the network spends the great majority of its time in its attractors, when choosing between the singleton attractor 000 ($\psi^z(00) = 0$) and the singleton attractor 001 ($\psi^z(00) = 1$), a majority decision based on the data indicates the singleton attractor 000. Indeed, if 001 were only observed rarely in the data, one might conjecture it to be a noisy version of 000 or a transient state of the form $001 \rightarrow 000$.

The situation becomes more flexible with the use of PBNs. We re-consider the three-gene situation with data set $\{000, 001\}$. At first glance it may appear that we have three possibilities: (1) 000 and 001 compose an attractor cycle in the same Boolean network; (2) they are singleton attractors in a single Boolean network; or (3) they are singleton attractors in different contexts. But the first situation is not possible because $000 \rightarrow 001$ requires $\psi^z(00) = 1$, and $001 \rightarrow 000$ requires $\psi^z(00) = 0$.

As for the second possibility, it involves the choice just discussed. If we choose $\psi^z(00) = 0$, then to have the network remain in an attractor, we must have $\psi^x(00) = 0$ and $\psi^y(00) = 0$, in which case 000 is an attractor and 001 is a transient state; if we choose $\psi^z(00) = 1$, then to have the network remain in an attractor, we must have $\psi^x(01) = 0$ and $\psi^y(01) = 0$, in which case 001 is an attractor and 000 is a transient state. Thus, we choose $\psi^z(00)$ based on the majority decision. The third possibility occurs by using context: 000 and 001 are singleton attractors in different contexts, in which case we have $\psi^z(00) = 0$ in one context and $\psi^z(00) = 1$ in the other, with all conflicts being resolved. Note that this same analysis applies whenever there are two data points and they differ only for a single gene.

For another situation, consider the data set $\{000, 111\}$. The same three apparent possibilities appear, but now they are all truly possible. We could have the cycle $000 \rightarrow 111$. This would not create a conflict in any predictor definitions: $\psi^x(00) = \psi^y(00) = \psi^z(00) = 1$ and $\psi^x(11) = \psi^y(11) = \psi^z(11) = 0$. They could also form two singleton attractors in the same Boolean network, with $\psi^x(00) = \psi^y(00) = \psi^z(00) = 0$ and $\psi^x(11) = \psi^y(11) = \psi^z(11) = 1$. Lastly, they could be singleton attractors in different contexts of a PBN. Using either non-contextual or contextual design, they appear as singleton attractors in a single Boolean network. Were the data actually reflective of a cycle in a real regulatory system, then the inference would be erroneous. Because the steady-state data is insufficient to infer dynamics, a learning assumption has been made (here and in the past) that favors short cycles over long, in this case favoring singleton attractors. Moreover, the number of contexts is minimized by assuming them to be singleton attractors in a Boolean network. Note that the same analysis applies whenever there are two data points and they differ by more than a single gene.

To help clarify the issue, we define two states to be neighbors if they differ by a single gene. A data state is said to be isolated if it has no neighbors in the data and non-isolated otherwise. If two data states are neighbors, as are 000 and 001, then they require two contexts to avoid data inconsistency. Since context selection depends on the data frequencies, the frequencies of 000 and 001 affect the resulting PBN probabilities. On the other hand, if a data state is isolated, as is the case of 000 for the data set $\{000, 110, 111\}$, then it does not generate contexts. This is what happened in the preceding illustration using the data states 000 and 111. Both are isolated in the data and therefore there is a single context. When a data state is isolated, its frequency in the data does not affect the PBN probabilities.

Finally, note that there are many ways that data states can interact when taken as a group. For instance, a PBN with data set $\{000, 001, 010, 011\}$ has 4, 4, and 1 functions for x, y, and z, respectively, with a total of 16 contexts. A PBN with data set $\{010, 100, 101, 110\}$ has 2, 2, and 2 functions for x, y, and z, respectively, with a total of 8 contexts.

2. Filtering

We have addressed data inconsistency from the perspective of biological context. The context problem is inherent to an open system, one that receives inputs from external variables that affect the system output. We have focused on system design, and as with all inference procedures, the design precision is affected by noise. Data-consistent design begins with binary state vectors (profiles), under the assumption of previous filtering, normalization, and quantization. Generally speaking, it is hard to model the impact of various noise sources on high-level data analysis algorithms, the central problem being the large number of sources of variance inherent in the process of making these measurements — for instance, using cDNA microarrays. In many statistical papers, the measured gene expression data are assumed to have multiple noise sources: sample preparation, labeling, hybridization, background fluorescence, different arrays, fluorescent dyes, and different printing locations. As with any high-level processing, network design is influenced by lower-level processing. In our case, noisy observation vectors can negatively affect design because our aim is to have the steady-state distribution of the designed network agree with the empirical distribution. In particular, noisy observations can result in spurious contexts.

Relative to data-consistent design, there is a more fundamental issue than observation noise pertaining to the number of contexts generated by the data, namely, sample heterogeneity. In many cases microarray data are obtained from heterogeneous cell populations, in particular, when tumor samples are analyzed. In fact, the entire issue of contextual modeling relates to data heterogeneity: the data relating to a specific set of genes composing a network derive from heterogeneous sources because each source is conditioned by factors external to the network. This heterogeneity affects model design. If in the case of a Bayesian network the conditional probability of a gene given its parents is estimated across sample data arising from heterogeneous subpopulations, then the conditioning is in effect averaged across different data sources and the resulting conditional probability does not specifically apply to any of the subpopulations. The same can be said of PBN (or PGRN) design using coefficients of determination computed relative to the full sample. It is precisely our desire to make PGRN design specific to the subpopulations (contexts) arising from external latent variables that has motivated data-consistent design. Consequently, when there is excessive sample heterogeneity there can be an extraordinarily large number of contexts.

To reduce the large number of contexts arising from excessive data heterogeneity (or from observation noise) we can filter the data by reducing the binary profiles. Specifically, if two profiles are very close, we can join them, thereby identifying their individual contexts. Since we lack a heterogeneity model it is impossible to optimally derive this identification filter and we therefore take an intuitive approach, which has generally been how data filtering has proceeded in the context of microarrays. The filter is applied in the following manner: (1) if a profile is observed more than once in the data, then it remains invariant; (2) if a profile appears only once in the data and it is within Hamming distance 1 of a repeated profile, then it is identified with the repeated profile; (3) if an unrepeated profile is not within Hamming distance 1 of a repeated profile, then it is left invariant. The details of a profile reduction algorithm can be found in Appendix B. The idea is straightforward: Singleton profiles that are almost identical to repeated profiles are assumed to result from either noise or statistically less important contexts very close to more important contexts. In practice, one can choose to use or not use the Hamming filter.

D. Application: Melanoma Network

We apply the contextual-design method to a genetic network that has served as a model to study the external control of genetic regulatory networks, in particular, for the regulatory avoidance of metastatic melanoma — for instance, in [46], where the context-sensitive PBN was constructed by the Bayesian connectivity approach.

The ten genes/proteins considered here were first identified in a study concerned with the feasibility of producing Markovian networks whose stationary distributions closely reflect the data [29]. The chosen genes/proteins arose from data in a study of metastatic melanoma [47]. In that study, the abundance of messenger RNA for the gene WNT5A was found to be highly discriminating between cells with properties typically associated with high metastatic competence versus those with low metastatic competence. These findings were validated and expanded in a second study [48].

In the second study, 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. A further finding of interest was that an intervention that blocked the Wnt5a protein from activating its receptor, the use of an antibody that binds Wnt5a protein, could substantially reduce Wnt5a's ability to induce a metastatic phenotype. This suggests a study of control based on interventions that alter the contribution of the WNT5A gene's action to biological regulation, since the available data suggest that disruption of this influence could reduce the chance of a melanoma metastasizing. The control objective is to externally down-regulate the WNT5A gene, because WNT5A ceasing to be down-regulated is strongly predictive of the onset of metastasis. Owing to computational issues relating to dynamic programming, in the control studies only 7 of the original 10 genes/proteins were used; here we use the full set of 10 to demonstrate network design: RET-1, HADHB, MMP-3, S100P, pirin, MART-1, synuclein, STC2, PHO-C, and WNT5A.

In the original expression study, 31 expression profiles were found for the 10 genes, with some profiles repeated. Table IV lists the 20 distinct profiles, along with their counts. As discussed previously, when we design a PBN, we must generalize the unspecified entries in the truth table. Here we do so by majority vote: if half or more of the entries have value 1, then set all the unspecified entries to 1; otherwise set them to 0. If we design a PBN based on the 20 profiles without any filtering, the resulting PBN has 128 contexts. The Hamming-distance filter yields 18 distinct profiles. They and their counts are shown in Table V. Under the Hamming-distance filter and majority-vote generalization, the designed PBN has 4 contexts. Table VI lists the attractors in each context and the data profiles (in decimal form for convenience). As must be the case, the PBN captures all the data profiles as attractors. There is only

Table III. Gene/Protein Annotations of Melanoma Data

Gene/protein	Pseudoname
RET-1	g_1
HADHB	g_2
MMP-3	g_3
S100P	g_4
pirin	g_5
MART-1	g_6
synuclein	g_7
STC2	g_8
PHO-C	g_9
WNT5A	g_{10}

one spurious attractor point, 702.

E. Conclusion

This chapter provides an inference procedure for probabilistic genetic regulatory networks in which the network contains contexts to model the data in such a way that it is consistent for each context. The intent is to view genomic regulation as deterministic (up to gene perturbation), with data inconsistencies due to variables outside the modeled network. A key aspect of the inference procedure is that every data state must be an attractor in at least one context, which is consistent with the assumption that the data states are attractor states for the real biological system. The dynamics, and therefore the steady-state distribution of the model, depend on generalization. This is to be expected since the inference problem is an ill-posed inverse problem ow-

Table IV. Expression Profiles for Melanoma

Profile	Genes/proteins										Count
#	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}	
1	1	0	0	1	1	1	1	1	0	1	2
2	1	1	0	1	1	1	0	0	0	0	1
3	1	0	1	0	1	1	1	1	0	1	1
4	1	0	0	1	1	1	1	1	0	0	2
5	0	1	0	1	1	1	0	0	1	1	1
6	1	0	1	1	1	1	1	1	1	1	1
7	0	1	0	1	1	1	1	1	0	1	1
8	0	1	0	0	1	1	0	0	0	1	2
9	0	0	0	1	1	1	0	0	0	1	1
10	0	1	1	0	1	0	0	0	1	1	1
11	0	1	0	0	1	0	0	0	1	1	1
12	1	0	1	0	1	0	1	0	1	0	1
13	1	0	1	1	1	0	1	1	1	0	2
14	1	0	1	0	0	0	1	1	1	0	8
15	0	0	1	0	1	1	0	0	0	0	1
16	0	0	1	0	0	0	1	1	1	0	1
17	0	1	0	1	0	1	0	0	1	0	1
18	0	1	0	0	1	1	0	0	1	1	1
19	0	0	1	1	0	1	1	0	1	0	1
20	0	0	1	0	0	0	1	0	1	0	1

Table V. Filtered Expression Profiles

g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}	Count
1	0	0	1	1	1	1	1	0	1	2
1	1	0	1	1	1	0	0	0	0	1
1	0	1	0	1	1	1	1	0	1	1
1	0	0	1	1	1	1	1	0	0	2
0	1	0	1	1	1	0	0	1	1	1
1	0	1	1	1	1	1	1	1	1	1
0	1	0	1	1	1	1	1	0	1	1
0	1	0	0	1	1	0	0	0	1	3
0	0	0	1	1	1	0	0	0	1	1
0	1	1	0	1	0	0	0	1	1	1
0	1	0	0	1	0	0	0	1	1	1
1	0	1	0	1	0	1	0	1	0	1
1	0	1	1	1	0	1	1	1	0	2
1	0	1	0	0	0	1	1	1	0	9
0	0	1	0	1	1	0	0	0	0	1
0	1	0	1	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0	1	0	1
0	0	1	0	0	0	1	0	1	0	1

ing to a lack of dynamical data. Hence, generalization concerns the following issue: given the attractors, what kind of inference can be obtained relative to the dynamics of the network? The attractors constrain the dynamical behavior, but do not determine it. In particular, they alone do not determine their basin structure. Future work will concentrate on the critical issue of generalization. Given a set of prior network properties postulated in accord with biological considerations, the aim will be to construct generalizations that yield networks possessing the desired properties.

Of particular importance is the manner in which generalization affects network connectivity. Whereas it is often assumed in PBN design that connectivity is limited and this limitation is imposed on design, the theory in this chapter depends on the possibility of full connectivity. We refer to this possibility because once the realizations are determined they can be reduced so that they only involve essential variables, thereby reducing the connectivity. The degree to which the connectivity is reduced

Table VI. Attractors of the Melanoma Network

Context 1	113 138 176 218 305 338 371 381 419 637 654 682 701 702 750 767 880
Context 2	113 138 176 218 291 305 338 371 381 637 654 682 701 702 750 767 880
Context 3	113 138 176 218 305 338 371 381 419 636 654 682 701 702 750 767 880
Context 4	113 138 176 218 291 305 338 371 381 636 654 682 701 702 750 767 880
All Attractors	113 138 176 218 291 305 338 371 381 419 636 637 654 682 701 702 750 767 880
Data Profiles	113 138 176 218 291 305 338 371 381 419 636 637 654 682 701 750 767 880

by logic reduction depends on the generalization. Going further, one might at the outset choose to limit the connectivity. Prior limitation might make data-consistent design impossible; however, one might try to achieve close-to-data-consistent design, where the closeness is based on some objective criterion. These considerations lead to two areas of ongoing research: (1) posing a suitable definition of connectivity minimization and developing efficient algorithms to select a generalization minimizing connectivity, and (2) defining an appropriate probabilistic criterion for approximate data consistency and developing efficient algorithms to optimize design relative to the criterion.

CHAPTER IV

OPTIMIZATION OF THE CONTEXT-SENSITIVE MODEL FOR GENETIC REGULATORY NETWORKS*

A. Introduction

When building function-based (rule-based) genetic networks from gene-expression data, the functions are often derived via some optimization-based criterion, with perhaps the imposition of biological constraints [43]. This requires determining, for each gene g, the genes that will serve as input to the function giving the value of g and the structure of the function. Except in rare circumstances, the optimal function for a gene will not be a perfect predictor owing to the inconsistencies in the data. This means that a specific combination of values of the regulatory genes will not necessarily correspond to a single value of the target gene. Thus, network design is inherently probabilistic. These inconsistencies can be modeled in a manner reflecting context changes in regulation, as discussed in the last chapter (also see [10]). The network can be in any of a number of contexts. Within a context, the network behaves deterministically and the generated data are consistent.

Chapter III proposed a method for inferring context-sensitive probabilistic Boolean models for genetic regulatory networks. In this chapter we address two issues arising

*© 2006 IEEE. Reprinted from *IEEE Transactions on Circuits and Systems I*, see [49] for complete publication information.

with consistency-based model design.

- First, an issue of *generalization* results from the inference method of the last chapter: Based strictly on the data, typically there is more than one PBN complying with the same data under the consistency requirement. Thus, we must find a criterion that enables us to perform a model selection on the candidate PBNs, thereby giving rise to an optimization problem.

- Second, for both computational and biological reasons, we may wish to constrain the connectivity optimization; that is, we may wish to restrict the maximum number of variables allowed in a regulatory function. This may require loosening the consistency requirement. Thus, there is the question of how to adapt the network with minimal loss of fidelity.

For the generalization problem, we recognize the connectivity (number of predictors for a target gene) and realization complexity in terms of an optimality measure for the PBN and two algorithms are proposed. The second algorithm is proven to be able to find a minimally connected PBN. We treat the constrained connectivity issue as a rephrased lossy coding problem and design an algorithm that attains the required connectivity by removing some of the predictors (regulators) of each target gene in a way that minimizes the probability of error in every regulating function. As in the last chapter, we remain in a binary setting.

B. Generalization via Optimization Criteria

1. Optimization Criteria: A Motivating Example

When applying the PBN design procedure in Chapter III, the truth tables are usually incompletely specified by available data, and these ambiguous entries must be assigned values according to some protocol. This is called *generalization* issue. Addressing this issue is essential because it affects the structure and final expression of Boolean functions of the designed PBN.

Example 1 *Suppose gene Z is potentially regulated by genes W, X and Y, and the relationships in Table VII are inferred from gene expression data. This table is*

Table VII. Truth Table for Example 1

Row number	W	X	Y	Z
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	$\times$
4	1	0	0	0
5	1	0	1	0
6	1	1	0	$\times$
7	1	1	1	1

incompletely specified, since entries marked with $\times$ (meaning "don't-care") cannot be inferred from available information, and generalization is needed. If we assign 0 to $\times$ in row 3 and 1 to $\times$ in row 6, then the Boolean function will be

$$Z = f_1(W, X) = \bar{W}\bar{X} + WX,$$

which reads "((not W) and (not X)) or (W and X)". If both $\times$'s are assigned 1, then we obtain the Boolean function

$$Z = f_2(W, X, Y) = \bar{W}\bar{X} + \bar{W}XY + WX.$$

It can be seen that the former generalization involves less variables, and requires less logic gates. □

This example suggests that we can look for a generalization that achieves a minimally interconnected PBN, with the simplest logic rules. In other words, for each gene, we would like its predictors to be as few as possible with a simple prediction rule; hence the PBN is minimally connected, and the realization of Boolean functions is minimized. The significance of minimizing connectivity is not limited to its own sake, but also reflects the biological propensity for low connectivity.

2. Preliminary Knowledge

There are many ways to describe a Boolean function via logic gates. Here, we adopt the standard sum of products (SOP) form, which can be implemented with two-level Boolean logic circuits consisting solely of AND and OR gates (besides NOT gates whenever necessary), where AND gates are used only in the first level, and OR gates are used only in the second level. Therefore, we have the following objective in mind when generalizing a PBN:

Objective 1 ***Generalization for Gene Networks***

1. *Achieve a minimally connected network;*
2. *Seek the simplest sum-of-products Boolean realizations.*

Notice that a simplest SOP Boolean realization may not be one that has minimal variables, and vice versa. For instance, $WX+Y$ versus $WX+\bar{W}\bar{X}$. Thus in achieving the objective, it is a matter of balancing goal (1) and goal (2). One can either give goal (2) a higher priority by seeking a simplest SOP realization with as few variables as possible or give goal (1) a higher priority by achieving minimal variables first and finding a simplest SOP realization based on the minimal variables.

Since we are often faced with a small sample of data, there can be a considerable number of "don't cares" in the truth tables. Taking account of only the number of genes and contexts, the sheer multitude of possible ways to generalize will be intimidating. To address the generalization problem, we employ a Boolean function minimization technique tailored to the case of incompletely specified truth tables. Given any completely fixed truth table, an initial Boolean function in SOP form can be obtained by the following steps:

1. Pick out every combination of input variables that yields output 1.
2. Represent each combination (called a *minterm*) as a product (logic AND) of input variables or their complements. If an input variable X value is 1, use the variable itself; otherwise, use its inverse $\bar{X}$.
3. Write the Boolean function as a sum (logic OR) of all minterms.

A Boolean function obtained directly from a truth table (without any simplification) in SOP form is often written as a sum using binary string representations of the minterms. For instance, $f(W, X, Y) = WX\bar{Y} + WXY$ can be written as $f(W, X, Y) = 110 + 111$. It can also be expressed as a sum of row numbers of the minterms, i.e., each minterm is represented by a decimal number converted from its binary string, so that $f(W, X, Y) = \sum(6, 7)$.

After finding an initial Boolean function, one can often simplify its expression by applying the following logic rules repeatedly: (1) $X \cdot 1 = 1 \cdot X = X$; (2) $X + \bar{X} = 1$; and (3) $WX + WY = W(X + Y)$. During simplification, if a variable vanishes, it will be replaced by $-$ in binary string representation. The following definitions provide the nomenclature for Boolean function minimization.

Definition 1 *[50] A Boolean expression φ is said to imply another Boolean expression ψ, designated as $\varphi \Rightarrow \psi$, if, when φ and ψ are considered as functions, ψ has the value 1 at least at every combination at which φ has the value 1.*

Definition 2 *[51, 50] Let ξ be product of literals (a literal is a variable or its complement) and ψ be a Boolean function. If $\xi \Rightarrow \psi$, then ξ is said to be an implicant of ψ.*

Definition 3 *[50] A product of literals ξ is a prime implicant of ψ if $\xi \Rightarrow \psi$ and if deleting any literal from ξ results in a new product term that does not imply ψ.*

The following theorem suggests that, to find the minimal expression of a Boolean function, it suffices to look for its smallest set of prime implicants.

Theorem 2 *[50] Any minimal sum-of-products expression of a Boolean function ψ is equivalent to a sum of prime implicants of ψ.*

3. Optimization Algorithms

Simplification of a Boolean function can be achieved via the Quine-McCluskey (Q-M) method [52, 53], which is a tabular algorithm for Boolean reduction that can be adapted to deal with incompletely specified truth tables. The Q-M method lists all the minterms and applies the adjacency rule repeatedly to combine qualified minterm pairs. After finding every prime implicant (PI) of the Boolean function, it then tabulates the results to search for a smallest set of PIs that covers the Boolean function. Since the objective is to find a generalization resulting in the simplest Boolean function (in SOP form), we are not only concerned with minimization given a (fixed) truth table, but also optimization upon an incompletely specified truth table, with uncertain outputs ("don't-cares"). This can be achieved by a slight variation of the original Q-M method, allowing don't-cares along with minterms, but excluding them when searching for the minimum set of prime implicants.

a. Algorithm 1

Algorithm 1 ***PBN Generalization via Modified Quine-McCluskey Minimization***

1. *Construct truth tables of the PBN from sample data, leaving undetermined entries as $\times$'s. In each table, pick out all minterms and don't-cares and group them according to the number of* 1*'s (the* index*) in each term, e.g. the index of* 01101 *is* 3*. Put them in ascending order of index in table $\mathcal{T}_j$, initially setting $j := 1$;*

2. *In the group of index i, for each term, find all its partners in the adjacent group of index $i + 1$ such that each partner differs from it by one digit only (i.e.,* 0 *and* 1 *respectively). Combine them to form a new term in which the formerly different digit is replaced by $-$, e.g,* 01001 *(index =* 2*) combines with* 01011 *(index =* 3*) to form* 010$-$1 *(index =* 2*); while* 01$-$10 *(index =* 2*) combines with* 01$-$11 *(index =* 3*) to form* 01$-$1$-$ *(index =* 2*). Put the newly formed terms into table $\mathcal{T}_{j+1}$. If a term has no partner at all, put itself to $\mathcal{T}_{j+1}$;*

3. *Let $i := i + 1$ and repeat step 2 until end of table $\mathcal{T}_j$;*

4. *Let $j := j + 1$ and repeat the above process until no further combination is possible. The final table $\mathcal{T}_f$ now contains all* generalized *prime implicants (GPIs). They are GPIs but not real PIs, because some of them actually imply don't-cares in the truth table, not the original* 1*s.*

5. *Construct a two-dimensional* prime implicant table *$\mathcal{P}$, where each column corresponds to a minterm (but not don't-care), and each row corresponds to an*

initial generalized prime implicant. Mark the (i, j) position of $\mathcal{P}$ by $*$ if the ith GPI implies the jth minterm. Delete rows (GPIs) without $*$. The remaining rows are PIs.

6. *If a column in $\mathcal{P}$ has only one $*$, then the corresponding row is identified as an* essential prime implicant *(EPI). Find all EPIs and include them in the set of minimum prime implicants Ξ. Put the variables contained in the EPIs to set $\mathcal{V}$. Then remove all EPIs and corresponding minterms (i.e., columns with single $*$) from $\mathcal{P}$.*

7. *Find a row with the maximum number of $*$'s, denoting this number s.* ***Search all rows that contain s $*$'s and pick the PI that adds the least number of new variables to $\mathcal{V}$.*** *Include this PI in Ξ and update the variable set $\mathcal{V}$. Remove this row and its corresponding columns from $\mathcal{P}$.*

8. *Repeat the above step until there is no $*$ in $\mathcal{P}$. Now, Ξ is the minimum set of PIs and the variables contained in $\mathcal{V}$ form the minimum predictor set.*

Remark 1 Standard Boolean minimization focuses on a minimal realization rather than connectivity. Our modification (the highlighted part) favors a realization with lower connectivity when two or more realizations are equally simple.

Example 2 *This example illustrates the above algorithm step by step. Consider the partially specified truth table of Table VII. Construct the following initial reduction table (the numbers in the parentheses denote the decimal numbers corresponding to the minterms or don't-cares):*

Indices	*Minterms*
0	000 (0)
1	001 (1)
2	011 (3)
	110 (6)
3	111 (7)

Minterms (0) *and* (1) *can merge to form an implicant* 00−, (1) *and* (3) *can be merged to implicant* 0−1, (3) *and* (7) *can merge to* −11, *and* (6) *and* (7) *can merge to* 11−. *Thus we have the second reduction table (below, left). Since no further reduction can be done to this table, we construct a table (below, right) for finding the minimal set of prime implicants, where the rows correspond to prime implicants, and the columns correspond to minterms (but not don't-cares).*

Indices	*Minterms*
0	00− (0, 1)
1	0−1 (1, 3)
2	−11 (3, 7)
	11− (6, 7)

	(0)	(1)	(7)
(0, 1)	∗	∗	
(1, 3)		∗	
(3, 7)			∗
(6, 7)			∗

Note that only (0), (1) *and* (7) *are real minterms, so any prime implicant involving only the don't-cares [*(3) *and* (6)*] will not appear. First we identify prime implicant* (0, 1) *to be the only EPI, and it contains variables* W, X. *That leaves minterm* (7) *open. Both* (3, 7) *and* (6, 7) *cover* (7), *but the latter does not add extra variables while the former does (variable* Y*). So we select* (6, 7). *Now all the minterms are covered and we have successfully found the minimal prime cover. Therefore, the generalized function should be* $f(W, X, Y) = \sum((0, 1), (6, 7)) = \sum(00-, 11-) = \bar{W}\bar{X} + WX$.

Notice the highlighted part of Algorithm 1 is a necessary modification to the standard Boolean minimization technique, for without it, prime implicant $(3, 7)$ *could have been selected in place of* $(6, 7)$ *and the function would have been* $f(W, X, Y) = \sum((0, 1), (3, 7)) = \sum(00-, -11) = \bar{W}\bar{X} + XY$*, which is a minimal SOP realization, but not an optimal solution in terms of minimal predictors.* □

b. Algorithm 2

In spite of the improvement made, Algorithm 1 favors minimal SOP realization more than minimal connectivity. If a minimal connectivity is more preferable, we may use the following alternative "prune-and-minimize" algorithm. The idea is to find through exhaustive search a smallest predictor set and the corresponding pruned truth table, then perform Algorithm 1 to obtain a minimal realization with regard to the smallest predictor set.

Algorithm 2 ***Prune-and-Minimize***

1. *Construct the original truth tables of the PBN.*
2. *For each truth table, suppose the set of variables are* $\mathcal{V}$*, with cardinality* $|\mathcal{V}| = k$*.*
3. *Let* $m = 1$*. If there exists a subset* $\mathcal{V}_m = \{v_1, \cdots, v_m\} \subset \mathcal{V}$ *such that for all rows in the truth table which have the same values of* $v_1, \cdots, v_m$*, the function values do not contain both* 0 *and* 1 *(it is all right to have don't-cares), then* $\mathcal{V}_m$ *is a minimal predictor set. Prune the truth table accordingly. In determining the function value of the pruned truth table, if in the original table, under the same* $v_1, \cdots, v_m$*, the function values are all don't-cares, then the corresponding function value in the pruned table will also be don't-care. Otherwise, if the original function values under the same* $v_1, \cdots, v_m$ *contain at least one* 0 *(or* 1*,*

but not both), then the pruned function value will be 0 *(or* 1*, accordingly). If there does not exist such a subset* $\mathcal{V}_m$*, then let* $m := m+1$ *and repeat the search.*

4. *After pruning all the truth tables, perform Algorithm 1 (with the following modification) to find minimal SOP realizations of Boolean functions, thus completing the generalization of the PBN.*

5. *Modification: change highlighted part in Algorithm 1 to:* ***Search all rows that contain*** s $*$***'s and pick the PI with least variables.***

Proposition 5 *A PBN designed by the procedure in Algorithm 2 has* (a) *the minimum connectivity, and* (b) *the minimal SOP realizations on the found predictor sets.*

Proof. The exhaustive search on the predictor set with lowest cardinality guarantees (a). As to (b), once a smallest predictor set is found and the truth table pruned, Algorithm 2 uses Algorithm 1, which always selects with priority the prime implicant which covers as many minterms as possible. Thus, the realization found will have the least number of products (incurring the smallest number of "OR" gates). The modification in Algorithm 2 chooses the prime implicants with least variables (thus shortest product), so that the number of "AND" gates is also minimal. Therefore, the SOP realization on the found predictor set must be minimal. ∎

From a general computational perspective, Algorithm 1 has the same level of computational complexity as the Q-M algorithm. However, for a fixed number of variables, the more sparsely specified the truth table, the more time it takes to perform Q-M algorithm. For a large-scale gene network with n genes, since the number of samples $N \ll 2^n$, the resulting functions will be extremely sparsely specified. Therefore, we recommend its application on PBNs of no more than 15 genes. Algorithm 2 may handle a larger gene network because it reduces predictors of any gene from $n-1$ down to n^* before Boolean minimization, and in practical problems the genetic regulatory network usually has low connectivity, typically $n^* \leq 5$. The time used for searching n^* variables is well compensated by performing Boolean minimization on less variables, which is confirmed in the melanoma example, Section 1. This is the main advantage of Algorithm 2.

C. Constrained Connectivity Optimization

1. Algorithm

In this section we consider the constrained optimization problem in which an upper limit is set on the maximum number of predictors allowed in a PBN, i.e., we restrict the connectivity to be no more than a prescribed integer κ^* in the entire network. Under the constraint, the resultant PBN may no longer reflect the original data with complete information; instead, there is a tradeoff between information capacity and network connectivity. From previous discussion, we are able to find the smallest predictor sets (called *full predictor sets*) for genes within the network such that consistency is satisfied. We make the following assumption.

Assumption *Under the constraint on maximum allowed connectivity, the suboptimal predictor set is a subset of the full predictor set.*

Under this assumption, the constrained-connectivity optimization problem can be interpreted as a lossy coding problem: we are to transmit a random row of a truth table by a fixed code length. The truth table contains $k+1$ binary variables, where $\mathbf{x} = (x_1, \cdots, x_k)$ is the input vector, and y is the output variable. For the lth row, the values of the input and output variables are denoted by $\mathbf{x}^l = (x_1^l, \cdots, x_k^l)$ and y^l, respectively. Each row of truth table can be encoded into a string of $k+1$ bits and this coding will be lossless. When code length is limited to $\kappa^* + 1$ with $\kappa^* < k$, however, we must select a subset of κ^* input variables from $x_1, \cdots, x_k$ so that the code will convey as much correct information as possible so that the original truth table can be recovered from received shortened codes with the least error.

Suppose the Boolean function of the original truth table is $f(x_1, \cdots, x_k)$, which in the case of reduced input variables has become $f'(x_{n_1}, \cdots, x_{n_{\kappa^*}})$, with $\{x_{n_1}, \cdots, x_{n_{\kappa^*}}\}$ being a subset of $\{x_1, \cdots, x_k\}$. As a result, for the lth row of original truth table, $f'(x^l_{n_1}, \cdots, x^l_{n_{\kappa^*}})$ may or may not equal y^l. More generally, if a weight w_l is assigned to the lth row, then we can define the cost function concerning chosen reduced inputs $\mathbf{x}_n = (x_{n_1}, \cdots, x_{n_{\kappa^*}})$ to be the expected (i.e. the weighted average of) probability of error, P_e, written as

$$J = E[P_e] = \sum_{l=0}^{2^k-1} I_{f'(\mathbf{x}^l_n)\neq y^l}(\mathbf{x}^l_n)w_l,$$

where w_l satisfies $\sum_{l=0}^{2^k-1} w_l = 1$.

This idea can naturally be extended to the constrained-connectivity-optimization problem for PBNs. Consider the ith context in a PBN, supposing gene g_j has κ_{ij} predictors. By forcing connectivity to be no more than κ^* ($\kappa^* < \kappa_{ij}$), $\kappa_{ij} - \kappa^*$ predictors have to be dropped, and doing so will create discrepancies when predicting g_i. We desire the best κ^* predictors out of the original κ_{ij} such that the target gene can be predicted with the least probability of error. If we assume that all combinations of values for the predictor genes are equally likely, then the following algorithm suffices.

Algorithm 3 ***Constrained Connectivity Optimization***

1. *Apply Algorithm 2 to obtain a minimally connected n-gene PBN, which has r contexts. For the ith context and jth gene, the optimal predictor set is $\mathcal{V}^{ij}$, and its cardinality $|\mathcal{V}_{ij}| = \kappa_{ij}$. Set $i := 1, j := 1$.*

2. *If $\kappa_{ij} \leq \kappa^*$, there is nothing to be done. Otherwise, if $\kappa_{ij} > \kappa^*$, choose a subset $\mathcal{V}^{(k)}_{ij} \subset \mathcal{V}_{ij}$ with $|\mathcal{V}^{(k)}_{ij}| = \kappa^*$, $k = 1, \cdots, \binom{\kappa_{ij}}{\kappa^*}$. Variables in $\mathcal{V}^{(k)}_{ij}$ can take on*

2^{κ^*} *different combinations of values, ranging from* $00\cdots0$ *to* $11\cdots1$*. For the* l*th combination they take, those remaining variables (not included in* $\mathcal{V}_{ij}^{(k)}$*) can have* $2^{\kappa_{ij}-\kappa^*}$ *different combinations, from which one can count the times that the function value (i.e., target gene* g_j*) is* 1 *or* 0 *in the truth table, and denote them by* $t_{1,ijl}^{(k)}$ *and* $t_{0,ijl}^{(k)}$ *, respectively (the sum of the two equaling* $2^{\kappa_{ij}-\kappa^*}$*). Let* $t_{ijl}^{(k)} := \min(t_{1,ijl}^{(k)}, t_{0,ijl}^{(k)})$*. Compute* $t_{ij}^{(k)} = \sum_{l=1}^{2^{\kappa^*}} t_{ijl}^{(k)}$*, and the probability of error for* $\mathcal{V}_{ij}^{(k)}$*,*

$$\epsilon_{ij}^{(k)} = \frac{t_{ij}^{(k)}}{2^{\kappa_{ij}}}.$$

3. *Compute* $\epsilon_{ij}^{(k)}$ *for all subsets* $\mathcal{V}_{ij}^{(k)}$*,* $k : 1 \le k \le \binom{\kappa_{ij}}{\kappa^*}$*, and let*

$$\epsilon_{ij}^* := \min_k \epsilon_{ij}^{(k)}, \quad k_{ij}^* := \operatorname*{argmin}_k \ \epsilon_{ij}^{(k)}.$$

Then $\mathcal{V}_{ij}^* := \mathcal{V}_{ij}^{(k_{ij}^*)}$ *is the suboptimal predictor set for gene* g_j *in context* i*.*

4. *Repeat for all genes and all contexts. Redefine the Boolean functions according to suboptimal predictor sets.*

2. Example

This algorithm is illustrated in the following example.

Example 3 *In one context of some PBN,* g_1 *is determined by* g_2, g_3, g_4 *via the Boolean function*

$$g_1 = \psi_1(g_2, g_3, g_4) = g_2g_3 + g_2g_4. \tag{4.1}$$

Find the best two genes which can predict g_1 *with the least probability of error, supposing all the combinations of predictor values are equally probable. Table VIII is the truth table for Boolean function (4.1). In rows 4 and 5,* $g_2, g_3 = 1, 0$*, while* g_1 *outputs* 0 *and* 1 *in rows 4 and 5, respectively. When* $g_2, g_3 = 0, 0$ *(or* $0, 1$*, or* $1, 1$*),* g_1 *outputs*

Table VIII. Truth Table for Function (4.1)

Row number	g_2	g_3	g_4	g_1
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

the same value for $g_4 = 0$ and $g_4 = 1$. Thus, the probability of error with predictor set $\{g_2, g_3\}$ is $1/8 = 0.125$. In rows 1 and 5, when $g_3, g_4 = 0, 1$, while g_1 outputs 0 and 1 in rows 1 and 5, respectively. Similar observations apply for rows 2 and 6, as well as for rows 3 and 7. Therefore, the error probability with predictor set $\{g_3, g_4\}$ is $3/8 = 0.375$.

When using two variables to predict g_1, the minimum probability of error (0.125) is achieved by either $\{g_2, g_3\}$ or $\{g_2, g_4\}$. Consequently, by using a subset of predictors, we lose fidelity of the original Boolean function. Here, the new Boolean function disagrees with the old one by 12.5%. □

D. Applications

1. Melanoma Network

We demonstrate the application of Algorithms 1 and 2 on the melanoma network considered in Chapter III.

The original data set for the study of metastatic melanoma [47] consists of 31 expression profiles for 10 genes and proteins: RET-1, HADHB, MMP-3, S100P, pirin, MART-1, synuclein, STC2, PHO-C and WNT5A (see Tables III and IV). Consider designing a 10-gene PBN from the data set given in the following table, which has resulted from applying the Hamming filter with distance $H = 2$, and in which, for convenience, the gene/protein names are replaced by the labels g_1 through g_{10}:

$g_1 g_2 g_3 \cdots g_9 g_{10}$	Count
1 0 0 1 1 1 1 1 0 1	5
1 1 0 1 1 1 0 0 0 0	1
1 0 0 1 1 1 1 1 0 0	2
0 1 0 0 1 1 0 0 0 1	6
0 1 1 0 1 0 0 0 1 1	1
1 0 1 1 1 0 1 1 1 0	3
1 0 1 0 0 0 1 1 1 0	10
0 0 1 0 1 1 0 0 0 0	1
0 1 0 1 0 1 0 0 1 0	1
0 0 1 1 0 1 1 0 1 0	1

For each gene/protein, the possible predictors are the remaining 9 genes/proteins, which means the connectivity is up to 9. After obtaining an initial 2-context PBN (with partially specified Boolean functions) from the consistency based design method in Chapter III, we must generalize the network. If no optimization is done and we generalize it by majority vote (if at least half of the specified entries in a truth table have value 1, then set all the don't-cares to 1; otherwise set them to 0), then the connectivity is 9 for each gene. If we apply Algorithm 1, then the connectivity is much lower (see Table IX). Note that for g_1 through g_9, each has only one function (i.e., same function in both contexts), while g_{10} has two functions (different function for each context). Note that, for g_2, Algorithm 1 with the highlighted improvement (bold-faced text in Step 7) gives a smaller predictor set than without (marked by $*$), even though both require the same number of AND and OR gates. Table X gives

Table IX. Ten-Gene PBN Generalized through Algorithm 1

Gene/Protein	Connectivity	Predictor set	Boolean function
g_1	3	g_4, g_8, g_9	$g_8 + g_4\bar{g}_9$
g_2	3	g_3, g_6, g_7	$\bar{g}_3\bar{g}_7 + \bar{g}_6\bar{g}_7$
$g_2(*)$	$4(*)$	g_1, g_3, g_6, g_7 $(*)$	$\bar{g}_1\bar{g}_6 + \bar{g}_3\bar{g}_7$ $(*)$
g_3	3	g_1, g_2, g_6	$\bar{g}_1\bar{g}_2 + \bar{g}_6$
g_4	2	g_1, g_5	$g_1g_5 + \bar{g}_1\bar{g}_5$
g_5	4	g_4, g_6, g_9, g_{10}	$\bar{g}_9 + g_{10} + g_4\bar{g}_6$
g_6	3	g_1, g_2, g_3	$\bar{g}_1\bar{g}_2 + \bar{g}_3$
g_7	3	g_2, g_5, g_8	$\bar{g}_2\bar{g}_5 + g_8$
g_8	2	g_1, g_2	$g_1\bar{g}_2$
g_9	2	g_5, g_6	$\bar{g}_5 + \bar{g}_6$
g_{10} (function 1)	3	g_2, g_3, g_4	$\bar{g}_2\bar{g}_3 + g_2\bar{g}_4$
g_{10} (function 2)	2	g_2, g_4	$g_2\bar{g}_4$

the results obtained through Algorithm 2. The highlighted part shows the difference with Table IX.

Let us compare g_1 in both tables, where the latter has lower connectivity, while the former has a simpler realization. The reason lies in the different emphases of the two algorithms. Since we have more interest in achieving a lowest connectivity, the result of the latter algorithm is still preferred. Comparing g_5 in both tables, both Boolean functions need an equal total number of AND and OR gates, while

Table X. Ten-Gene PBN Generalized through Algorithm 2

Gene/Protein	Connectivity	Predictor set	Boolean function
g_1	**2**	$g_4, \mathbf{g_5}$	$\mathbf{g_4 g_5 + \bar{g}_4 \bar{g}_5}$
g_2	3	g_3, g_6, g_7	$\bar{g}_3\bar{g}_7 + \bar{g}_6\bar{g}_7$
g_3	3	g_1, g_2, g_6	$\bar{g}_1\bar{g}_2 + \bar{g}_6$
g_4	2	g_1, g_5	$g_1 g_5 + \bar{g}_1\bar{g}_5$
g_5	**2**	$\mathbf{g_1}, g_4$	$\mathbf{g_1 g_4 + \bar{g}_1 \bar{g}_4}$
g_6	3	g_1, g_2, g_3	$\bar{g}_1\bar{g}_2 + \bar{g}_3$
g_7	3	$\mathbf{g_1}, g_2, \mathbf{g_4}$	$\mathbf{g_1 \bar{g}_2 + \bar{g}_2 g_4}$
g_8	2	g_1, g_2	$g_1\bar{g}_2$
g_9	2	g_5, g_6	$\bar{g}_5 + \bar{g}_6$
g_{10} (function 1)	3	g_2, g_3, g_4	$\bar{g}_2\bar{g}_3 + g_2\bar{g}_4$
g_{10} (function 2)	2	g_2, g_4	$g_2\bar{g}_4$

the latter has lower connectivity. Comparing the results on g_7, the latter algorithm seems to lose its lead because it gives a worse realization than the former algorithm. However, as a minimal predictor set is not unique, the predictor sets obtained by the two algorithms are both the smallest. Thus the latter algorithm still finds the best possible Boolean realization on the found predictor set, although it does not compete with the realization on another minimal predictor set. If one is keen on finding the very best realization on *all* minimal predictor sets, this can be done by comparing minimal realizations on all minimal predictor sets and choosing the best.

The latter algorithm takes a much shorter time (less than 6 seconds with MATLAB 6.5) than the former (more than 1 hour with MATLAB 6.5), perhaps contrary to intuition. The reason is that there are far more don't-cares than specified values in this case, as a result of which the reduction process in Boolean minimization is

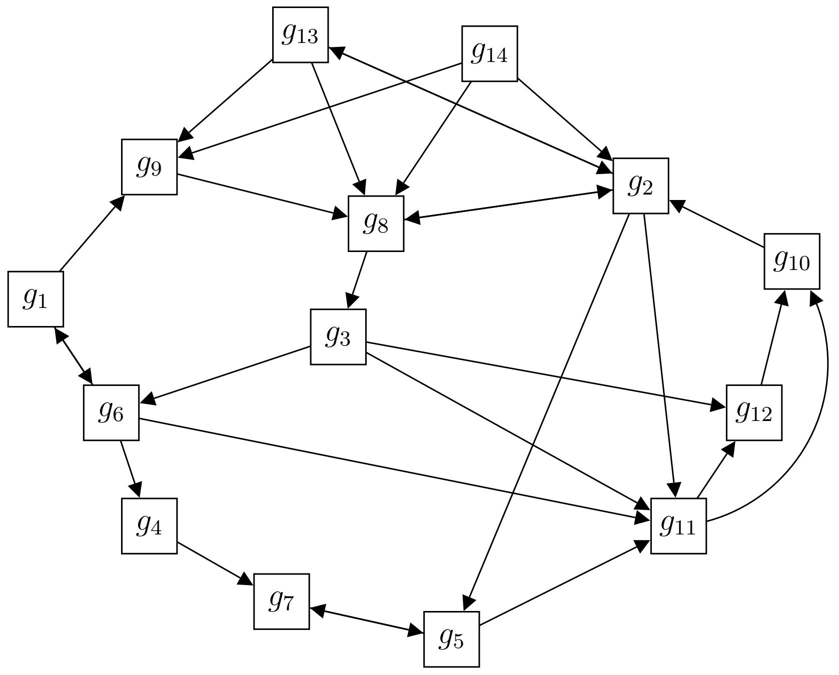


Fig. 2. Glioma network in reference [1].

time-consuming. Thus one is better off pruning the variables first. Moreover, the exhaustive search method will discard a predictor set when the first sign of failure emerges (both 0 and 1 show up in function values where $v_1, \cdots, v_m$ are the same), and will complete the mission immediately when the first satisfying set $\mathcal{V}_m$ is encountered. These are time-saving measures.

2. Glioma Network

Consider the following data set containing 14 genes and proteins (listed in Table XI) chosen from the glioma data used in [1]. The sample size is 26.

From the glioma data in Table XII, a context-sensitive PBN is designed with g_1 having 2 functions, g_{14} having 4 functions, and the others having 1 function each. The Prune-and-Minimize algorithm is applied to generalize the PBN and the results are listed in Table XIII.

Table XI. Gene/Protein Annotations of Glioma Data

Gene/protein	Pseudoname
c-rel proto-oncogene protein	g_1
(CCK4); transmembrane receptor PTK7	g_2
GNB1; transducin beta 2 subunit 2	g_3
GNB1; transducin beta 1	g_4
NKEFB; TSA; TDPX1	g_5
(MAP kinase 1; MAPK1; PRKM1); (ERK2)	g_6
NDKB; NME2; PUF; NM23B	g_7
GRB2; ASH	g_8
FSHR	g_9
DSG2; HDGC	g_{10}
(GDF1)+UOG-1	g_{11}
(RAI;RNH);	g_{12}
VEGF	g_{13}
FGF7; KGF	g_{14}

It can be seen that for the 14-node network, the connectivity of each gene ranges from 1 to 5. Now let us compare the results to the network of [1] (see Fig. 3 on p. 1245 of the reference), shown in Fig. 2. Note that the construction of network Fig. 2 used influence to define the strength of connections between nodes (not shown here, but shown in Fig. 3 of [1]). It was not constructed by considering coefficient of determination or context-sensitive design, and not intended to discover the prediction relationship. Therefore, the network in Fig. 2 was designed in an entirely different framework from the PBN shown in Table XIII.

Table XII. Glioma Data Preprocessed with Hamming Distance Filter

Profile	Genes/proteins														Count
#	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}	g_{11}	g_{12}	g_{13}	g_{14}	
1	1	1	1	1	1	1	1	1	1	0	1	1	0	0	1
2	1	0	1	1	0	1	0	1	0	0	1	1	0	1	1
3	0	0	1	1	0	1	0	1	0	0	1	1	0	1	1
4	1	0	1	1	1	1	1	1	0	0	0	0	0	1	9
5	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1
6	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1
7	0	1	1	1	0	1	0	0	0	0	0	0	1	0	1
8	1	0	1	1	0	1	0	0	1	1	1	1	0	0	1
9	1	1	1	1	0	1	0	0	0	1	0	0	1	0	1
10	1	1	1	1	1	1	1	0	0	0	1	1	1	0	1
11	0	0	1	1	1	1	1	1	0	0	1	1	0	0	1
12	1	0	1	1	0	1	0	1	0	0	1	1	0	0	1
13	1	0	0	0	1	1	0	0	1	1	0	0	0	0	5
14	1	0	1	1	1	1	1	0	1	0	1	1	0	0	1

In spite of the differences, the similarities between Fig. 2 and our network (Table XIII) are apparent. For instance, in Table XIII, g_2, g_8 and g_9 form a tight relationship, such that each one is predicted by the other two. Such relationship is also present in Fig. 2, in which there are bi-directional links between g_2 and g_8, a direct link from g_9 to g_8, and an indirect link from g_9 to g_2 through g_{14}. Also look at the relationship between g_3 and g_4, or the relationship between g_{11} and g_{12}, both in Table XIII, and they are well reflected in Fig. 2. Table XIII shows g_{13} can be predicted by g_2, g_8 and g_{10}, which partly coincides with Fig. 2. However, the result in Table XIII regarding the predictors of g_1 is very different from Fig. 2. Moreover, among the predictors for g_{14} in Table XIII, only g_8 (present in 3 functions out of the total 4 in the table) is related to g_{14} in Fig. 2 by being downstream of g_{14}, while other predictors show no strong relations in Fig. 2.

E. Conclusion

We have addressed generalization in consistency-based PBN design by considering the connectivity and realization complexity in terms of an optimality measure, and have developed two algorithms in this framework, the second guaranteed to produce a PBN with minimum connectivity and the minimal SOP realizations on the predictor sets. Next, we have treated constrained connectivity as a rephrased lossy coding problem and designed an algorithm that attains the required connectivity by removing some of the predictors of each target gene in a way that minimizes the probability of error in every regulating Boolean function. Future work will include the development of generalization and constrained connectivity optimization methods that take into consideration the effect on the network steady-state distribution.

Table XIII. Glioma PBN Generalized through the Prune-and-Minimize Algorithm

Gene/protein	Connectivity	Predictor set	Boolean function
g_1 (function 1)	5	$g_2, g_5, g_8, g_{10}, g_{11}$	$g_{10} + \bar{g}_8 g_{11} + \bar{g}_2\bar{g}_5 + \bar{g}_2\bar{g}_{11} + g_2 g_{11}$
g_1 (function 2)	5	$g_2, g_5, g_8, g_{10}, g_{14}$	$g_{10} + g_5 g_{14} + g_5\bar{g}_8 + g_2 g_8 + \bar{g}_2\bar{g}_5\bar{g}_{14}$
g_2	2	g_8, g_9	$\bar{g}_8\bar{g}_9 + g_8 g_9$
g_3	1	g_4	g_4
g_4	1	g_3	g_3
g_5	2	g_3, g_7	$\bar{g}_3 + g_7$
g_6	1	g_2	g_2
g_7	2	g_3, g_5	$g_3 g_5$
g_8	2	g_2, g_9	$\bar{g}_2\bar{g}_9 + g_2 g_9$
g_9	2	g_2, g_8	$\bar{g}_2\bar{g}_8 + g_2 g_8$
g_{10}	4	g_1, g_7, g_8, g_{13}	$g_8 g_{13} + g_1\bar{g}_7\bar{g}_8$
g_{11}	1	g_{12}	g_{12}
g_{12}	1	g_{11}	g_{11}
g_{13}	3	g_2, g_8, g_{10}	$g_2\bar{g}_8 + g_8 g_{10}$
g_{14} (function 1)	3	g_5, g_{10}, g_{11}	$\bar{g}_5\bar{g}_{10} + g_5\bar{g}_{10}\bar{g}_{11} + g_5 g_{10} g_{11}$
g_{14} (function 2)	3	g_1, g_5, g_8	$\bar{g}_5 g_8 + g_1 g_8$
g_{14} (function 3)	4	g_1, g_2, g_5, g_8	$g_1\bar{g}_2 g_5 g_8 + \bar{g}_1\bar{g}_2\bar{g}_5$
g_{14} (function 4)	3	g_1, g_5, g_8	$\bar{g}_1\bar{g}_5 g_8$

CHAPTER V

THE IMPACT OF PERTURBATIONS IN THE BOOLEAN MODELS OF GENETIC REGULATORY NETWORKS*

A. Introduction

A network is said to be *robust* relative to a certain network characteristic if a small change in network structure does not significantly affect the characteristic. In the case of a Boolean network, which is a rule-based binary network, a key form of robustness is with respect to how a small change in a regulatory rule, say the flip of one value in its truth table, affects the steady state of the network. Robustness is a double-edged sword. For instance, BNs are used to model gene regulation, with gene expressions being quantized as 0 and 1 to represent *not expressed* and *expressed states*, respectively, and gene regulation is described by Boolean logic. Because network inference is inherently ill-posed on account of measurement error and the impact of latent variables, which are either immeasurable or simply not included in the model (whose influence nevertheless still exists), model robustness is desirable for inference so that slightly differently inferred networks will exhibit similar fundamental characteristics. On the other hand, if the goal is to intervene in the network, for instance, to modify its long-term behavior so as to drive it away from undesirable states, say, metastasis in cancer, then robustness is undesirable because it impedes intervention. This chapter addresses the robustness of Boolean networks relative to small perturbations of the regulatory rules.

The objective of this study is comprised of two aspects:

*© The Author 2007. Reprinted, with permission of the Oxford University Press, from *Bioinformatics*, see [54] for complete publication information.

- Provide a theory to analytically predict the consequences of function perturbations in a Boolean network, including the impacts on state transitions and steady-state properties (mainly attractors).

- Apply the analytical theory to intervene or control Boolean networks to obtain desirable properties.

Given that Boolean networks are often used to model genetic regulation, the preceding two aspects will help gain insight into gene regulatory network modeling in the following ways:

- By helping to analyze the influences of modeling uncertainty and latent variables, since these two common phenomena can often be formulated as changed Boolean functions, thereby falling into the function perturbation category.

- By aiding in the design of intervention methods to control gene regulation for the purpose of altering steady-state cell behavior.

- By providing the means to identify the regulatory perturbations underlying observed changes in gene behavior.

This study is motivated by the fact that these issues have not been treated thoroughly in the literature: (1) previous works usually concern state perturbation, which is temporary in nature, instead of function perturbation; (2) many works study ensembles of BNs by exploring their overall (statistical) behavior, but do no consider the effect on a specific BN; (3) many works do not mention attractors, and these characterize the long-term properties of BNs, which in the context of gene regulatory networks represent phenotypic properties. A review of the existing literature on BN robustness is provided below.

There exist two kinds of variations with respect to a BN: perturbation of the states and perturbation of the functions (the regulatory rules). The first kind of perturbation is temporary and entails a reset of the network state to a new state. Such a perturbation does not alter the structure of the BN and has no influence on its steady-state properties; however, it does affect the network dynamics by replacing the original time trajectory with a new trajectory. Since a BN possesses one or more attractors, and each attractor has its basin of attraction (consisting of the states that will eventually transit to this attractor), after a perturbation of the state, the new trajectory may converge to the original trajectory and reach the same attractor, or it may go to another basin of attraction and reach a different attractor. If it is preferable to reach the original attractor after the perturbation, then it is desirable that the BN be *stable* or *dynamically robust*, meaning that it has a tendency to resist disturbance of the state and converge to its original trajectory.

The second kind of perturbation, namely, perturbation of the functions, has a more fundamental impact on the BN and has been less studied. The network steady-state distribution may undergo a permanent transformation: the basins of attraction for some attractors will enlarge, shrink or shift; some attractors will disappear; and new attractors may be created. Therefore, starting from the same initial state, the new trajectory may or may not reach the original attractor. Understanding the impact of function perturbation on steady-state properties is important for application. As noted, depending on the context, robustness may or may not be desirable: the robustness of an attractor and its basin of attraction is desirable if we would like to preserve them.

State perturbations affect the network dynamics, not the steady-state properties. This issue has been well-studied, often via the ensemble behavior of a large number of random synchronous BNs (all the functions in the BN are updated simultaneously at each time step). One study shows that a natural class of robust networks is composed of scale-free networks, in which a small (but significant) fraction of the elements are highly connected and the majority of the elements are poorly connected [55]. Another demonstrates the robustness of BNs whose functions belong to certain Post classes [56]. The conclusions in [55] and [56] are based on the ensemble behavior of random Boolean networks. A different approach explores the robustness of annealed (the connections and functions will change at each time step) Boolean networks through the bias-map, which is a mapping $b_t \rightarrow b_{t+1}$, b_t being the probability of a gene being 1 at time step t [57]. It introduces the concept of a stabilizing Boolean function and relates it to network stability (dynamic robustness). It shows that many Post and canalizing functions are stabilizing functions, which agrees with [55] and [56]. Another perspective is to define robustness as the expected probability of a single flipped input altering the output of a Boolean function over a distribution for K-input functions and averaged over all the nodes of the network [58]. Flipping the function input is a form of state perturbation and the robustness measure concerns the ensemble behavior of random Boolean networks. The conclusion is that Boolean networks with canalizing functions are stable (the robustness measure is less than 1). Robustness to noise is treated in [59], where the function output has a probability of flipping its value, and the first crossing time is defined as the time needed for two time trajectories with different initial states to cross. The paper concludes that, for two categories of random Boolean networks (in the chaotic and ordered phases), whether the initial states belong to the same basin of attraction or not, the average first crossing time over a large number of networks behaves robustly.

In the above cited studies, only state perturbations are considered. As a state perturbation affects the network in a temporary manner, meaning that only the network dynamics are affected, we naturally would like to pursue a further issue, namely, how a perturbation of the functions in the network will influence the attractors and the long-term behaviors of the network. The effect of function perturbation is less studied in the literature. One of such studies is based on the ensemble performance of a large number of random Boolean networks: In [60], network stability is measured by the overlap of state space transitions of the original BN and the one-bit mutant BN resulting from a perturbation of a Boolean function through a one-bit change to its truth table. The authors discover that adding a redundant node can boost the robustness of one-bit mutant Boolean networks. Here, although function perturbation is studied, emphasis on the network attractors is lacking. Another work, contributed by [61], studies a different kind of perturbation: updating the functions in an asynchronous setting. The effect of asynchronous updates of the functions on the dynamics of Boolean-type models for the *Drosophila melanogaster* segment polarity genes is considered and different asynchronous update schemes are tested. One model is found to be robust to changes in the initial state and robust to the update variability; certain restrictions (a minimal prepattern) on the update scheme must be satisfied to ensure convergence to the desired wild-type steady state. Owing to the nature of asynchronous networks, the attractor robustness is not treated in the sense that attractors exactly constitute the steady state in a synchronous Boolean network.

Here we study the effect of function perturbation on the attractors in a homogenous synchronous BN, meaning that the Boolean functions are updated simultaneously and the functions do not change over time. Our analysis is applicable to any individual BN instead of being targeted at the ensemble performance of a type of BNs such as in [55, 58], or a particular BN such as in [61]. Therefore, the methods and results in this chapter are more general. We do not define a robust measure; rather, we explore the exact consequences of function perturbations and show how they can be utilized for analysis and synthesis. We focus on function perturbation in the form of a one-bit change of the truth table and explore its impact on the attractors. We address both the robustness and flexibility issues, and show that the latter can be useful for intervention in Boolean networks. Since state transitions completely characterize the network dynamics and define a BN's attractors and basins of attraction, we propose to pursue our objective through the following issues: (1) Impact on state transitions; (2) Impact on attractors, namely, which attractors will be invariant to the perturbations and which non-attractor states will become new attractors; and (3) Applications, including intervention (given a BN, design an intervention strategy to achieve a certain objective through function perturbation) and perturbation identification (given the observed state transitions of a BN and the state transitions after function perturbation, identify the perturbation).

B. Robustness Analysis of Function Perturbation

1. Problem Formulation

Recall from Chapter II that a (homogeneous and synchronous) Boolean network can be represented as $G(U, \mathbf{f})$, where $U = \{x_1, x_2, \cdots, x_n\}$, $\mathbf{f}) = (f_1, f_2, \cdots, f_n)$. The value of node x_i at time $t+1$ can be predicted by its k_i input nodes at time t by Equation 2.1, or in the following simplified form

$$x_i = f_i(x_{i1}, x_{i2}, ..., x_{ik_i}), \quad 1 \leq i \leq n.$$

An input variable to some function is said to be a *fictitious* if its value being one or another has no effect on the function output; otherwise, it is called an essential variable. We will use the notations $f_i(x_1, x_2, \cdots, x_n)$ and $f_i(x_{i1}, x_{i2}, ..., x_{ik_i})$ interchangeably, with the former using all the variables (some of which are fictitious) and the latter specifying only the essential variables.

The state of the Boolean network at time t is denoted as $\mathbf{x}(t)$ and a specific state can be written as an n-dimensional binary vector $(a_1, a_2, \cdots, a_n)$, where $a_i \in \{0,1\}, 1 \leq i \leq n$, or written as $a_1a_2\cdots a_n$ in a compact form. Since the state space of an n-node Boolean network is $S = \{0,1\}^n = \{00\cdots0, 00\cdots1, \cdots, 11\cdots1\}$, a list of the one-step successor states of every state in S can thus be constructed, which is generally called the *state transition rules* or *state transitions*. For instance, if a Boolean network has 3 nodes, its one-step state transition rules will consist of 8 states, which are the successors of states $000, 001, \cdots, 110, 111$, respectively.

The long-term behavior of a Boolean network is characterized by its attractors. An attractor can either be a singleton attractor or an attractor cycle, depending on the number of states it contains. The attractors in BNs modeling biological networks are typically associated with phenotypes and tend to be short [13], with biological stability contributing to singleton attractors. For instance, singleton attractors have been associated with phenotypes such as cell proliferation and apoptosis [14]. Our basic results will be stated for singleton attractors, for which the analysis leads to tractable propositions, and we will then show how they can be extended to multiple-state attractor cycles, albeit, with increased complexity.

A Boolean function can be represented by a truth table shown below.

Row label	$x_{i1}x_{i2}\cdots x_{ik_i}$	$f_i(\cdot)$
1	$00\cdots0$	$f_i(00\cdots0)$
2	$00\cdots1$	$f_i(00\cdots1)$
$\vdots$	$\vdots$	$\vdots$
2^{k_i}	$11\cdots1$	$f_i(11\cdots1)$

If the function f_i depends on k_i input variables $x_{i1}, x_{i2}, \cdots, x_{ik_i}$, then the evaluated input vector on row j $(1 \leq j \leq 2^{k_i})$ of the truth table is denoted by $\mathbf{a}^i_j$, with $\mathbf{a}^i_j \in \{0,1\}^{k_i}$. For instance, in the truth table above, $\mathbf{a}^i_2 = 00\cdots1$.

The state transition $\mathbf{s} \to \mathbf{w}$ depends on the vector function $\mathbf{f} = (f_1, f_2, \cdots, f_n)$. Restricting our attention to f_i means considering the ith mapping $\mathbf{s} \to w_i$. Since f_i depends on $(x_{i1}, x_{i2}, \cdots, x_{ik_i})$, the mapping depends on $(s_{i1}, s_{i2}, \cdots, s_{ik_i})$ only. If $\mathbf{u} \neq \mathbf{s}$ but $(u_{i1}, u_{i2}, \cdots, u_{ik_i}) = (s_{i1}, s_{i2}, \cdots, s_{ik_i})$, then $\mathbf{u}$ and $\mathbf{s}$ both map to w_i under f_i. If we let $\mathrm{In}_i(\mathbf{s}) = (s_{i1}, s_{i2}, \cdots, s_{ik_i})$ denote the input vector for function f_i as it operates on state $\mathbf{s}$, then $\mathrm{In}_i(\mathbf{u}) = \mathrm{In}_i(\mathbf{s})$ implies $f_i(\mathbf{u}) = f_i(\mathbf{s})$. For instance, if $n = 5$, $\mathbf{s} = 01001$, $\mathbf{u} = 00011$, and $x_{i1}x_{i2}\cdots x_{ik_i} = x_1x_3x_5$, then $\mathrm{In}_i(\mathbf{u}) = \mathrm{In}_i(\mathbf{s}) = 001$ and $f_i(\mathbf{u}) = f_i(\mathbf{s}) = f_i(001)$.

In a Boolean network, a one-bit perturbation occurs when one chooses a function f_i and makes a one-bit change of its truth table by flipping the value on the jth entry $(1 \leq j \leq 2^{k_i})$, that is, change 0 to 1 or change 1 to 0. We denote the new function by $f_i^{(j)}$, so that the one-bit perturbation on row j takes the form $f_i \to f_i^{(j)}$, where $f_i^{(j)}(\mathbf{a}_j^i) = 1 - f_i(\mathbf{a}_j^i)$. Since single-node flips play a key role in our analysis, we introduce the following notation: if $\mathbf{s} = (s_1, s_2, \cdots, s_n)$, then $\mathbf{s}^{(i)} = (s_1, \cdots, s_{i-1}, 1 - s_i, s_{i+1}, \cdots, s_n)$.

2. Theoretical Results on One-Bit Function Perturbation

We begin with a proposition and corollaries describing the basic effects of a single one-bit perturbation on the state transitions of a Boolean network.

Proposition 6 *The state transition* $\mathbf{s} \to \mathbf{w}$ *is affected by the one-bit perturbation* $f_i \to f_i^{(j)}$ *if and only if* $\mathrm{In}_i(\mathbf{s}) = \mathbf{a}_j^i$. *If the state transition is affected, then the new state transition will be* $\mathbf{s} \to \mathbf{w}^{(i)}$.

Proof. The first statement follows at once from the fact the ith mapping transition $\mathbf{s} \to w_i$ depends only on $\mathrm{In}_i(\mathbf{s})$ and this is affected by the perturbation if and only if $\mathrm{In}_i(\mathbf{s}) = \mathbf{a}_j^i$. Next, suppose the transition is affected by the perturbation. Then, absent perturbation, the ith mapping is given by $\mathbf{s} \rightarrow f_i(\mathrm{In}_i(\mathbf{s})) = f_i(\mathbf{a}_j^i) = w_i$; with perturbation, the ith mapping is $\mathbf{s} \rightarrow f_i^{(j)}(\mathrm{In}_i(\mathbf{s})) = f_i^{(j)}(\mathbf{a}_j^i) = 1 - f_i(\mathbf{a}_j^i) = 1 - w_i$. Since the other $n-1$ mappings remain unchanged, following the perturbation, state $\mathbf{s}$ will transit to $\mathbf{w}^{(i)}$. ∎

Corollary 1 *If $|\mathbf{a}_j^i| = k_i$, then the one-bit perturbation $f_i \rightarrow f_i^{(j)}$ will result in 2^{n-k_i} changed state transitions in the state transition diagram. This is equivalent to 2^{n-k_i} altered edges in the state transition diagram.*

Proof. According to the preceding proposition, the transition of a state $\mathbf{s}$ is affected by the perturbation $f_i \rightarrow f_i^{(j)}$ if and only if $\mathrm{In}_i(\mathbf{s}) = \mathbf{a}_j^i$. Among the 2^n states $\mathbf{s}$, $\mathrm{In}_i(\mathbf{s}) = \mathbf{a}_j^i$ for exactly 2^{n-k_i} of them. ∎

Corollary 2 *(**Invariant singleton attractor**) Suppose state $\mathbf{s}$ is a singleton attractor. It will no longer be a singleton attractor following the one-bit perturbation $f_i \rightarrow f_i^{(j)}$ if and only if $\mathrm{In}_i(\mathbf{s}) = \mathbf{a}_j^i$.*

Proof. According to the proposition, subsequent to the perturbation, $\mathbf{s} \rightarrow \mathbf{s}^{(i)}$ if $\mathrm{In}_i(\mathbf{s}) = \mathbf{a}_j^i$, in which case it is no longer a singleton attractor, and $\mathbf{s} \rightarrow \mathbf{s}$ if $\mathrm{In}_i(\mathbf{s}) \neq \mathbf{a}_j^i$, in which case it remains a singleton attractor. ∎

Corollary 3 *(**Emerging singleton attractor**) A non-singleton-attractor state $\mathbf{s}$ becomes a singleton attractor as a result of the one-bit perturbation $f_i \rightarrow f_i^{(j)}$ if and only if the following are true: (1) $\mathrm{In}_i(\mathbf{s}) = \mathbf{a}_j^i$, and (2) absent the perturbation, $\mathbf{s} \rightarrow \mathbf{s}^{(i)}$.*

A natural question arises when a desirable singleton attractor $\mathbf{s}$ is lost on account of a one-bit perturbation: Can a second perturbation restore it? If the perturbation $f_i \rightarrow f_i^{(j)}$ causes $\mathbf{s}$ to no longer be a singleton attractor, then the new transition of $\mathbf{s}$

must be $\mathbf{s} \to \mathbf{s}^{(i)}$. According to the previous corollary, $\mathbf{s}$ will again become a singleton attractor as a result of the one-bit perturbation $f_k \to f_k^{(l)}$ if and only if $\text{In}_k(\mathbf{s}) = \mathbf{a}_l^k$, and, absent the perturbation, $\mathbf{s} \to \mathbf{s}^{(k)}$. From the second condition, since we know that $\mathbf{s} \to \mathbf{s}^{(i)}$, we must have $k = i$. Hence, from the first condition we must have $\text{In}_i(\mathbf{s}) = \mathbf{a}_l^i$, but from the fact that $\mathbf{s}$ has been affected by the perturbation $f_i \to f_i^{(j)}$, we know that $\text{In}_i(\mathbf{s}) = \mathbf{a}_j^i$. Hence, $\mathbf{s}$ is restored to being a singleton attractor by the same one-bit perturbation $f_i \to f_i^{(j)}$ that caused it to cease being a singleton attractor, and no other. This means that the original Boolean network is restored.

According to Corollary 2, a singleton attractor $\mathbf{s}$ is no longer a singleton attractor following a one-bit perturbation $f_i \to f_i^{(j)}$ if $\text{In}_i(\mathbf{s}) = \mathbf{a}_j^i$, but could it remain an attractor state as part of an attractor cycle following perturbation? Indeed it could. Consider the following situation for 3 nodes: 000 is a singleton attractor, 001 $\to$ 010 and 010 $\to$ 000, so that 001 and 010 are in the basin of 000. The three functions are defined by $f_1(x_2, x_3) = 0$ for all x_2, x_3; $f_2(x_1, x_3) = 0$ for all x_1, x_3 except for $f_2(0, 1) = 1$; and $f_3(x_2, x_3) = 0$ for all x_2, x_3. Consider the one-bit perturbation $f_3 \to f_3^{(00)}$. Following the perturbation, the third function becomes $f_3^{(00)}(x_2, x_3) = 0$ for all x_2, x_3, except for $f_3^{(00)}(0, 0) = 1$. This leads to the following transitions, 000 $\to$ 001, 001 $\to$ 010, and 010 $\to$ 000, and hence the attractor cycle 000 $\to$ 001 $\to$ 010 $\to$ 000. Thus, our care in stating the results is not unwarranted.

From the preceding example, we see that a one-bit perturbation can result in a singleton attractor becoming a member in a multiple-state attractor cycle. On the other hand, it should be clear from Proposition 6 that a one-bit perturbation can affect a multiple-state attractor cycle. Suppose $\mathbf{s}_1 \to \mathbf{s}_2 \to \cdots \to \mathbf{s}_m \to \mathbf{s}_1$ is an m-state attractor cycle. It follows from Proposition 6 that this cycle will be affected by the one-bit perturbation $f_i \to f_i^{(j)}$ if and only if $\mathbf{a}_j^i \in \{\text{In}_i(\mathbf{s}_1), \text{In}_i(\mathbf{s}_2), ..., \text{In}_i(\mathbf{s}_m)\}$. By being affected, we mean that exact cycle is not an attractor cycle following perturbation.

For instance, suppose $\mathbf{a}_j^i = \mathrm{In}_i(\mathbf{s}_1)$. Then, following perturbation, $\mathbf{s}_1 \rightarrow \mathbf{s}_2^{(i)} \neq \mathbf{s}_2$. Of course, $\mathbf{s}_1$ might still be an attractor state as part of some other attractor cycle.

Corollary 1 puts an upper bound on the number of singleton attractors that can be lost owing to a one-bit perturbation $f_i \rightarrow f_i^{(j)}$, namely, $\min\{2^{n-k_i}, N\}$, where $|\mathbf{a}_j^i| = k_i$ and N is the number of singleton attractors. Taking a network view, the total number of singleton attractors lost is bounded by

$$\min\{2^n \sum_{i=1}^{n} 2^{-k_i}, N\} \leq \min\{n 2^{n-k_{\min}}, N\}$$

where $k_{\min}$ is the minimum connectivity among the nodes. Increased connectivity provides greater robustness relative to the loss of singleton attractors via one-bit perturbations.

Thus far, except for considering a second perturbation to restore a singleton attractor lost on account of a one-bit perturbation, we have focused on single one-bit perturbations. Increasing the number of one-bit perturbations increases the complexity of the problem; indeed, any Boolean network on the same variables can be obtained from any other via a sufficiently long sequence of one-bit perturbations. If we consider two one-bit perturbations, then there are two cases: (1) the same function is changed and a flip occurs on two rows; and (2) two functions are changed. The two cases can be expressed as: (1) $f_i \rightarrow f_i^{(j)}$ and $f_i^{(j)} \rightarrow f_i^{(j,l)}$, $j \neq l$; (2) $f_i \rightarrow f_i^{(j)}$ and $f_k \rightarrow f_k^{(l)}$, $i \neq k$. To extend Proposition 6 to two one-bit perturbations, we let $\mathbf{w}^{(i,k)}$ denote the state obtained from $\mathbf{w}$ by flipping the ith and kth nodes. We state separate extensions for the two cases: (1) The state transition $\mathbf{s} \rightarrow \mathbf{w}$ is affected by the two-bit perturbation $f_i \rightarrow f_i^{(j,l)}$, $j \neq l$, if and only if $\mathrm{In}_i(\mathbf{s}) \in \{\mathbf{a}_j^i, \mathbf{a}_l^i\}$, and if it is affected, then $\mathbf{s} \rightarrow \mathbf{w}^{(i)}$. (2) The state transition $\mathbf{s} \rightarrow \mathbf{w}$ is affected by the two one-bit perturbations $f_i \rightarrow f_i^{(j)}$ and $f_k \rightarrow f_k^{(l)}$ ($i \neq k$), if and only if $\mathrm{In}_i(\mathbf{s}) = \mathbf{a}_j^i$ or

$\mathrm{In}_k(\mathbf{s}) = \mathbf{a}_l^k$, and if it is affected, then $\mathbf{s} \to \mathbf{w}^{(i)}$ when $\mathrm{In}_i(\mathbf{s}) = \mathbf{a}_j^i$ and $\mathrm{In}_k(\mathbf{s}) \neq \mathbf{a}_l^k$, $\mathbf{s} \to \mathbf{w}^{(k)}$ when $\mathrm{In}_k(\mathbf{s}) = \mathbf{a}_l^k$ and $\mathrm{In}_i(\mathbf{s}) \neq \mathbf{a}_j^i$, and $\mathbf{s} \to \mathbf{w}^{(i,k)}$ when $\mathrm{In}_i(\mathbf{s}) = \mathbf{a}_j^i$ and $\mathrm{In}_k(\mathbf{s}) = \mathbf{a}_l^k$.

3. Extension to Two-Bit Function Perturbation

The corollaries of Proposition 6 are extended to the two one-bit perturbations as follows.

Corollary 4 *Supposing two one-bit perturbations take place in the Boolean network, consider the following two cases.*

Case (a): $f_i \to f_i^{(j,l)}$ (i.e., f_i is perturbed on rows j and l, $j \neq l$) will result in 2^{n-k_i+1} changed state transitions.

Case (b): $f_i \to f_i^{(l_1)}$ and $f_j \to f_j^{(l_2)}$ ($i \neq j$), assuming $|\mathbf{a}_{l_1}^i| = k_i$, $|\mathbf{a}_{l_2}^j| = k_j$, and f_i and f_j have k_{ij} input variables in common. If each of the k_{ij} variables takes the same value in $\mathbf{a}_{l_1}^i$ and in $\mathbf{a}_{l_2}^j$, then there will be $2^{n-k_i} + 2^{n-k_j} - 2^{n-k_i-k_j+k_{ij}}$ changed state transitions; otherwise, there will be $2^{n-k_i} + 2^{n-k_j}$ changed state transitions.

Corollary 5 *(**Invariant singleton attractor**) Suppose state $\mathbf{s}$ is a singleton attractor. It will no longer be a singleton attractor following the two-bit perturbation $f_i \to f_i^{(j,l)}$ if and only if $\mathrm{In}_i(\mathbf{s}) = \mathbf{a}_j^i$ or $\mathbf{a}_l^i$. $\mathbf{s}$ will cease to be a singleton attractor following the two one-bit perturbations $f_i \to f_i^{(l_1)}$ and $f_j \to f_j^{(l_2)}$ if and only if $\mathrm{In}_i(\mathbf{s}) = \mathbf{a}_{l_1}^i$ or $\mathrm{In}_j(\mathbf{s}) = \mathbf{a}_{l_2}^j$.*

Corollary 6 *(**Emerging singleton attractor**) A non-singleton-attractor state $\mathbf{s}$ becomes a singleton attractor as a result of the two-bit perturbation $f_i \to f_i^{(j,l)}$ if and only if the following is true: $\mathrm{In}_i(\mathbf{s}) = \mathbf{a}_j^i$ or $\mathbf{a}_l^i$, and absent the perturbation, $\mathbf{s} \to \mathbf{s}^{(i)}$.*

$\mathbf{s}$ will become a singleton attractor following the two one-bit perturbations $f_i \to f_i^{(l_1)}$ and $f_j \to f_j^{(l_2)}$ if and only if one of the following is true:

(1) $\mathrm{In}_i(\mathbf{s}) = \mathbf{a}^i_{l_1}$ *and* $\mathrm{In}_j(\mathbf{s}) = \mathbf{a}^j_{l_2}$, *and absent the perturbation,* $\mathbf{s} \to \mathbf{s}^{(i,j)}$;

(2) $\mathrm{In}_i(\mathbf{s}) = \mathbf{a}^i_{l_1}$ *and* $\mathrm{In}_j(\mathbf{s}) \neq \mathbf{a}^j_{l_2}$, *and absent the perturbation,* $\mathbf{s} \to \mathbf{s}^{(i)}$;

(3) $\mathrm{In}_i(\mathbf{s}) \neq \mathbf{a}^i_{l_1}$ *and* $\mathrm{In}_j(\mathbf{s}) = \mathbf{a}^j_{l_2}$, *and absent the perturbation,* $\mathbf{s} \to \mathbf{s}^{(j)}$.

From the extension of Proposition 6 and its corollaries to two one-bit perturbations, it is clear how to extend them to more than two one-bit perturbations, albeit, with an increased number of cases.

C. Algorithms

1. Network Intervention

One objective of network modeling is to use the model to design intervention strategies for affecting the dynamic evolution of the network. The methods apply whether or not a network possesses a steady-state distribution. We will point out whether the steady-state distribution is affected if it exists. Such intervention studies have focused on three general approaches: state perturbation, optimal control, and function perturbation.

With state perturbation, the state of the network is reset to an initial state and the network is allowed to evolve from there, the point being that the new trajectory will visit more desirable states [32]. The network structure is not changed. If the network possesses a steady-state distribution, then that distribution is not changed.

For optimal control, there exist one or more controllable variables that affect the transition probabilities of the network and these can be used to desirably affect its dynamic evolution [62]. The network structure is not changed. If the network possesses a steady-state distribution and the control is applied over a finite time horizon and then stopped, then the steady-state distribution is not changed [62, 46]; however, if the control is applied over infinite time (forever), then the steady-state distribution is changed to one whose mass is more concentrated in favorable states [63]. For instance, based upon a study finding that blocking the Wnt5a protein from activating its receptor could substantially reduce Wnt5a's ability to induce a metastatic phenotype [48], optimal control theory has been been applied to an expression-based network including the WNT5A gene in such a manner as to down-regulate WNT5A [62, 46, 63], the objective being to decrease the likelihood of metastasis.

In the case of function perturbation, one or more Boolean functions are changed to desirably alter the network. The network structure is changed [33]. If the network possesses a steady-state distribution, then that distribution is changed.

These applications have been in the context of probabilistic Boolean networks; however, since Boolean networks are a special case of probabilistic Boolean networks, the results apply at once. For instance, complexity issues relating to optimal control have been studied in the context of Boolean networks [64]. If a BN possesses random node flips, so that at any time point there is a positive probability of any node flipping from 0 to 1 or from 1 to 0, then it possesses a steady-state distribution. The classical deterministic BNs are free of such random node flips [13]. Thus, unless there is a single attractor cycle, there does not exist a steady-state distribution.

Before formally providing the procedure to control the stationary probabilities in a PBN via function perturbation, we revisit a problem considered in [33] to motivate and illustrate the methodology. In the Discussion section we will apply the procedure to alter a WNT5A network in order to decrease the likelihood of metastasis.

Given two sets of states, perhaps representing two different cellular functional states or phenotypes, the general problem is to specify some optimization criterion regarding the stationary probabilities of the states and to discover some multiple perturbation of a single function that best achieves the optimization. The analysis is for a probabilistic Boolean network. These have been discussed in many places, including review papers, so we leave their precise mathematical formulation to the literature [22]. Let us simply state that the analysis in [33] corresponds to an instantaneously random PBN, which means that at each time point each node function is randomly chosen from a set of possible functions and at each time point there is a probability p that a node value can be flipped. Each set of selected functions corresponds to a single Boolean network, so that probabilistically, each of these corresponds to a realization of the network.

In the example considered in [33], the PBN consists of three nodes, x_1, x_2 and x_3. Two functions correspond to x_1, with probabilities 0.6 and 0.4 respectively; node x_2 has a single function, and node x_3 also has two functions, with equal likelihood. Table XIV (see [33], pp. 436, Table 1) lists the functions and their selection probabilities (c_{ij}), and Fig. 3 (see [33], pp. 437, Fig. 1) shows the state transition diagram of the PBN when free of random node flips. Thus the PBN has $2 \times 1 \times 2 = 4$ constituent Boolean networks (Table XV). The probability of a node flip is $p = 0.01$, in which case the stationary probabilities of the two singleton attractors are $\pi(000) = 0.0752$ and $\pi(111) = 0.7310$. The intervention objective is to use function perturbation to make both new stationary probabilities, $\mu(000)$ and $\mu(111)$, close to 0.4, which means minimizing the error criterion $|\mu(000) - 0.4| + |\mu(111) - 0.4|$, while maintaining their total stationary mass. This corresponds to the objective of reducing the stationary probability of the undesirable state 111 while increasing the stationary probability of the desirable state 000 (see [33] for a detailed discussion).

The state transition diagrams absent node perturbation ($p = 0$) for the four individual BNs are shown in Fig. 4. Let BOA$\{x_1x_2x_3\}$ denote the basin of attraction of $x_1x_2x_3$ and $|$BOA$\{x_1x_2x_3\}|$ denote its size. It can be seen that in both BN1 and BN3, $|$BOA$\{000\}| = 1$ and $|$BOA$\{111\}| = 7$. In BN2, $|$BOA$\{000\}| = |$BOA$\{111\}| = 1$ and in BN4, $|$BOA$\{000\}| = 2$ and $|$BOA$\{111\}| = 1$. Since the sum of the stationary

Table XIV. Definition of Functions in the PBN

$x_1x_2x_3$	f_{11}	f_{12}	f_{21}	f_{31}	f_{32}
000	0	0	0	0	0
001	1	1	1	0	0
010	1	1	1	0	0
011	1	0	0	1	0
100	0	0	1	0	0
101	1	1	1	1	0
110	1	1	0	1	0
111	1	1	1	1	1
c_{ij}	0.6	0.4	1	0.5	0.5

Table XV. The Four Components of PBN

Network i	Network function f_i	Probability P_i
BN1	$(f_{11},\ f_{21},\ f_{31})$	0.3
BN2	$(f_{11},\ f_{21},\ f_{32})$	0.3
BN3	$(f_{12},\ f_{21},\ f_{31})$	0.2
BN4	$(f_{12},\ f_{21},\ f_{32})$	0.2

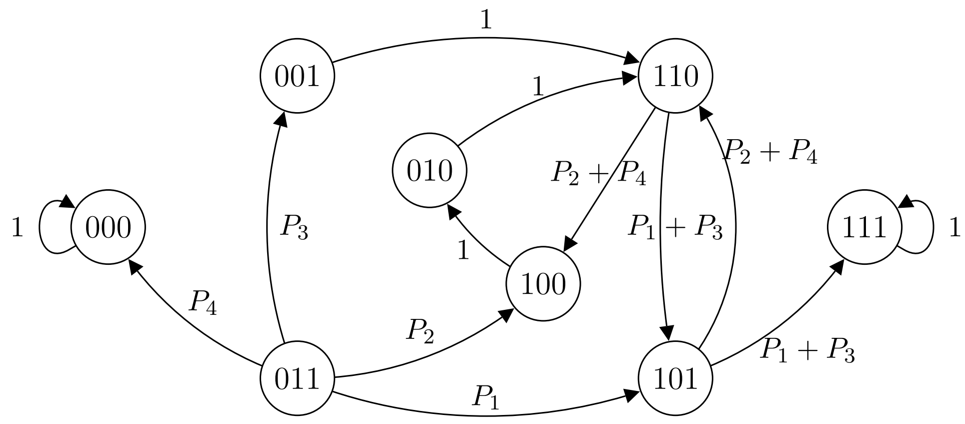
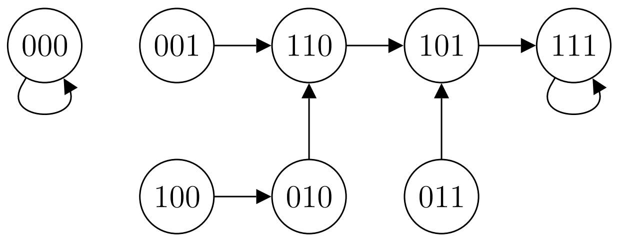


Fig. 3. State transition diagram of the PBN (probability of state perturbation $p = 0$).

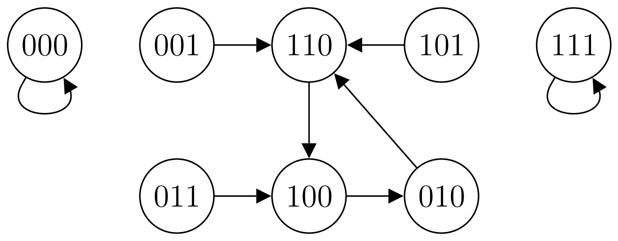
probabilities of 000 and 111 must not change, we need to increase the BOA of 000 and decrease the BOA of 111. Since function f_{21} is used in all four BNs, its perturbation will result in changes in all BNs. On the other hand, if we perturb any of the other four functions, only two BNs will be affected. So we prefer not to perturb f_{21} unless necessary. Recalling Corollary 1, in this example, $n = k_i = 3$ and $2^{n-k_i} = 1$, so perturbing one row of a function truth table will affect only one state transition.

First, consider BN1 and BN3. Can we increase the BOA of 000 by finding any state whose successor state differs from 000 by only 1-bit (preferably on the 1st and 3rd bit)? The answer is 011 of BN3. However, even if we make a one-bit perturbation of function f_{31} to let 011 $\rightarrow$ 000 in BN3, |BOA{000}| will increase by only 1. This perturbation also affects BN1 (011 $\rightarrow$ 100), but has no effect on BOA{000} or BOA{111}. Thus we give up this attempt.

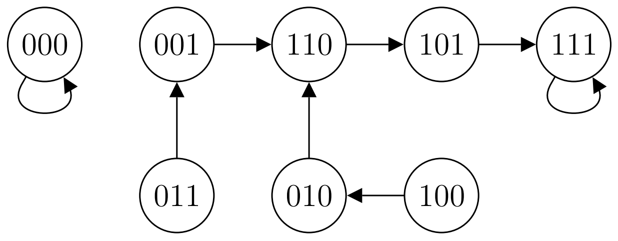
Now consider BN2 and BN4. Can we increase |BOA{000}| by a one-bit function perturbation (preferably not f_{21})? One possibility is to change the state transition 110 $\rightarrow$ 100 to 110 $\rightarrow$ 000 in BN2 and BN4, i.e., perturb f_{11} or f_{12}. Another possibility is to change the state transition 100 $\rightarrow$ 010 to 100 $\rightarrow$ 011 in BN4, namely perturb f_{32} to $f_{32}^{(5)}$. Consider the following choices:



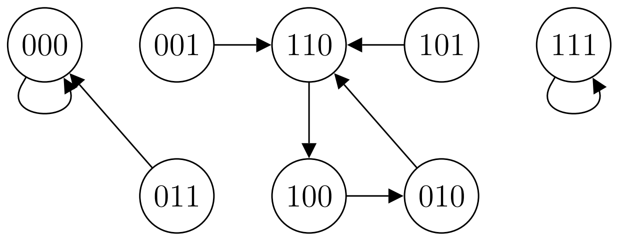
(a) BN1



(b) BN2



(c) BN3



(d) BN4

Fig. 4. State transition diagram of the 4 BNs (probability of state perturbation $p = 0$).

Choice 1: Perturb f_{11} to $f_{11}^{(7)}$ where $\mathbf{a}_7^{11} = 110$. As a consequence, in BN1, $110 \rightarrow 001$ and $|\text{BOA}\{111\}|$ decreases to 3; in BN2, $110 \rightarrow 000$, and $|\text{BOA}\{000\}|$ increases to 7. This is a possible candidate.

Choice 2: Perturb f_{12} to $f_{12}^{(7)}$. As a consequence, in BN3, $110 \rightarrow 001$ and $|\text{BOA}\{111\}|$ decreases to 2; in BN4, $110 \rightarrow 000$ and $|\text{BOA}\{000\}|$ increases to 7. This is a possible candidate.

Choice 3: Perturb f_{32} to $f_{32}^{(5)}$, where $\mathbf{a}_5^{32} = 100$. As a consequence, in BN4, $100 \rightarrow 011$ and $|\text{BOA}\{000\}|$ increases to 7; in BN2, $100 \rightarrow 011$, which does not affect BOA$\{111\}$ or BOA$\{000\}$. Since BN1 and BN3 adopt f_{31} rather than f_{32}, they are unaffected. Overall, this perturbation only increases $|\text{BOA}\{000\}|$ in BN4, without affecting BOA$\{111\}$. Thus, it cannot achieve the desired goal, which requires increasing $|\text{BOA}\{000\}|$ and decreasing $|\text{BOA}\{111\}|$. This choice is ruled out.

Simulations show that choice 1 yields stationary probabilities $\mu(000) = 0.6$ and $\mu(111) = 0.25$. Choice 2 yields $\mu(000) = 0.43$ and $\mu(111) = 0.41$, which are close to the goal. Hence, we adopt choice 2. Compare our solution to that of [33], where the function f_{12} is perturbed from $0, 1, 1, 0, 0, 1, 1, 1$ to $0, 0, 0, 1, 0, 1, 0, 1$ (4-bit perturbation, on rows 2, 3, 4, and 7), and the resulting stationary probabilities are $\mu(000) = 0.4068$ and $\mu(111) = 0.4128$. The solution of [33] is obtained by exhaustive search of all possible 1280 function perturbations (allowing one function to be perturbed by any number of flips in its truth table). Our solution is close to optimal with only a single-bit perturbation and it does not require an exhaustive search.

Using the preceding example as a guide, we have the following general procedure for optimizing the stationary probabilities in BNs or PBNs. Here we assume one-bit perturbation only.

1. Recognize the goal of optimization and formulate the error criterion. In the

preceding example, the goal is to have equal stationary probability mass for the target states 000 and 111, while the sum of the probability masses remains unchanged. The error criterion is $|\mu(000) - 0.4| + |\mu(111) - 0.4|$. Therefore, we must increase the probability mass of 000 and decrease that of 111 at the same time.

2. Determine the priority of perturbation for the functions. For instance, in a PBN, if some of the functions are common in two or more constituent BNs, it is preferred to perturb a function that affects as few BNs as possible. For another example, in a BN, it is more favorable to perturb a function that results in fewer changes in the state transitions.

3. Plot the state transition diagrams. Analyze the BOAs of the target states by taking into consideration the BOA sizes and the probabilities of BNs (in the case of a PBN). To increase the BOA of a target state $\mathbf{s}$ by a one-bit perturbation, find a candidate state outside BOA$\{\mathbf{s}\}$ whose next state differs from a state within BOA$\{\mathbf{s}\}$ by only one-bit. Find all such candidates. To decrease the BOA of a target state $\mathbf{s}$ by a one-bit perturbation, find a candidate state in BOA$\{\mathbf{s}\}$ whose next state differs from a state outside BOA$\{\mathbf{s}\}$ by only one-bit. Notice that in a PBN, perturbation of one function can result in changes in two or more constituent BNs.

4. List all the options of perturbation, from the highest priority to the lowest. For each option, draw new state transition diagrams and analyze the BOAs of the target states again. Throw away the options that are far from the goal. Notice that the steady-state probability mass of an attractor state is mainly affected by the size of its BOA, but also has to do with the BOA structure and the node flipping probability p. Therefore the BOA sizes can be used to estimate (but

are not deterministic of) the probability masses of the target states.

5. For the remaining options, make computations either by simulation or by direct computation through Markov chain analysis (see [32] for details), and pick the option that is closest to the goal. If two or more options are equally good, pick the one with the highest priority (e.g., perturbed function is present in the least number of BNs, or carries the least weight in a pre-defined importance rank, etc).

2. Identifying Function Perturbations

Suppose we have knowledge of the state transitions of a Boolean network. Imagine a perturbation occurs in the Boolean model unbeknownst to us except that we observe the new state transitions. By comparing the state transitions before and after perturbation, we may ask two questions: (1) Which Boolean function is perturbed? (2) On which row of the truth table is the perturbation? These are identification problems, useful in diagnosing changes in gene regulatory networks, such as changes caused by a disease, radiation therapy, drug treatment, etc.

We now present a general procedure for identifying function perturbations in a Boolean network:

1. Find out perturbed function(s).

   Let the list of state transition of the original Boolean network be given by $\mathbf{s}_0, \mathbf{s}_1, \cdots, \mathbf{s}_{2^n}$, these being the successor states of $00\cdots0, 00\cdots1, \cdots, 11\cdots1$, respectively. The list of state transitions of the perturbed Boolean network is denoted by $\mathbf{s}'_0, \mathbf{s}'_1, \cdots, \mathbf{s}'_{2^n}$. According to Proposition 6, if a state transition is affected by a one-bit perturbation on Boolean function f_i, then the new successor state differs from the old by the value of node x_i. Thus, by comparing the two lists, one can tell which function(s) is(are) perturbed.

2. Locate the flipped entries of the perturbed function(s).

   For the simple case of a one-bit perturbation in one function f_i, recall from Corollary 1 that 2^{n-k_i} states will have changed successors. Moreover, those states share the same value on nodes $x_{i1}, \cdots, x_{ik_i}$. Assume the differences between the state transition rules before and after perturbation are given by the states $\mathbf{s}_{d1}, \mathbf{s}_{d2}, \cdots, \mathbf{s}_{dm_i}$ versus the states $\mathbf{s}'_{d1}, \mathbf{s}'_{d2}, \cdots, \mathbf{s}'_{dm_i}$ and k_i is unknown. We can find k_i by computing $k_i = n - \log_2 m_i$. Knowing that those states are the successor states of $(d1)_2, (d2)_2, \cdots, (dm_i)_2$, which are the length-$n$ binary representations of decimal numbers $d1, d2, \cdots, dm_i$ (e.g, for a 3-node Boolean network, $(6)_2 = 110$), we may compare the states $(d1)_2, (d2)_2, \cdots, (dm_i)_2$ to find the common bits in order to identify the parent nodes of x_i, which are $x_{i1}, \cdots, x_{ik_i}$. Moreover, if

   $$\mathrm{In}_i((d1)_2) = \mathrm{In}_i((d2)_2) \cdots = \mathrm{In}_i((dm_i)_2) = (u_1, \cdots, u_{k_i}) = \mathbf{a}^i_j,$$

   then we can conclude that the perturbation on f_i takes place on the jth row of its truth table.

3. For the case of two-bit perturbations, we can refer to Corollaries 4, 5 and 6 for a similar analysis. Likewise, we can treat more complex perturbations, albeit with increased difficulty.

D. Applications

In this section we will apply the intervetion pocedure to beneficially control the gene WNT5A in a network releted to melanoma and to indentify a function perturbation in a *Drosophila melanogaster* segmentation polarity gene network.

1. Intervention in a WNT5A Network

We consider a Boolean network taken from a context-sensitive PBN model for a WNT5A network constructed for an intervention study [46]. The original PBN model was inferred from the data collected in a metastatic melanoma study [47] which found that that the abundance of the messenger RNA of the WNT5A gene was highly discriminating between cells with properties typically associated with high and low metastatic competence. A subsequent study [48] validated and expanded this finding by experimentally increasing the levels of the Wnt5a protein secreted by a melanoma cell line, as a result of which the metastatic competence of the cell line was directly altered. It was also found that through an intervention blocking the Wnt5a protein from activating its receptor, Wnt5a's ability to induce metastasis can be substantially reduced. These results suggest that using an intervention to down-regulate the WNT5A gene can lower the chance of metastasis in a cell line.

In [46] seven genes, WNT5A, pirin, S100P, RET1, HADHB and STC2, were selected for inference and intervention in a melanoma network. The objective was to exert a control variable for a finite time to steer the network dynamics towards desirable states, those for which WNT5A $= 0$, not highly expressed. Their method uses an external control to change the state transitions temporarily instead of changing the network structure. In our study, we will use a different approach and permanently alter the network structure by a minimal amount of function perturbation.

To study the chosen Boolean network, we relabel the seven genes, WNT5A, pirin, etc., as $x_1, x_2, \cdots, x_7$. The functions are defined in Table XVI, where under the heading "function values" each item is a binary string whose ith bit represents the function value on the ith row of the truth table. For instance,in the last row the

Table XVI. Definition of a WNT5A Boolean Network

Function	Input variables	Function values
f_1	x_6	10
f_2	x_2, x_4, x_6	00010111
f_3	x_3, x_4, x_7	10101010
f_4	x_4, x_6, x_7	00001111
f_5	x_2, x_5, x_7	10101111
f_6	x_2, x_3, x_4	01110111
f_7	x_2, x_7	1101

entry 1101 means that for the inputs $00, 01, 10, 11$, the function outputs are $1, 1, 0, 1$, respectively.

This BN has four attractors 0101111, 0110110, 0111110 and 1000001. Their BOA sizes are 48, 4, 16 and 60, respectively. The last attractor $\mathbf{s} = 1000001$ is undesirable, because WNT5A gene is up-regulated. Moreover, $\mathbf{s}$ has a large BOA (consisting of nearly 50% of the total number of states), so the possibility of reaching it is high. Our objective is to eliminate this attractor or minimize its BOA if elimination is impossible, that is, $\min|\text{BOA}\{\mathbf{s}\}|$. We will achieve this goal through function perturbations, with two constraints: (1) to perturb as few bits in the functions as possible; (2) to affect as few state transitions as possible.

For constraint (2), recall that the number of affected state transitions by one-bit perturbation equals 2^{n-k_i}, and it is preferable to choose from the following functions for perturbation: f_2, f_3, f_4, f_5, and f_6. As a result, 16 state transitions (1/8 of the total) will be affected.

To eliminate attractor $\mathbf{s}$, we will choose from the above five functions for one-bit perturbation and follow the steps below.

(i) In an effort to eliminate attractor $\mathbf{s}$ by a one-bit perturbation of a function, we wish to change the state transition from $\mathbf{s} \rightarrow \mathbf{s}$ to $\mathbf{s} \rightarrow \mathbf{u}$, such that $\mathbf{u}$ differs from $\mathbf{s}$ by exactly one bit. Since the 5 preferred functions to perturb are given, the candidate states will be $\mathbf{u}_1 = 1100001$, $\mathbf{u}_2 = 1010001$, $\mathbf{u}_3 = 1001001$, $\mathbf{u}_4 = 1000101$ and $\mathbf{u}_5 = 1000011$.

(ii) Consider the following 5 options.

Option 1 : Change the state transition to $\mathbf{s} \rightarrow \mathbf{u}_1$. Noticing $\mathrm{In}_2(\mathbf{s}) = 000$, we can achieve it through perturbation $f_2 \rightarrow f_2^{(1)}$. By applying the theoretical results, we can easily find that after the perturbation, the other 3 attractors are unaffected, and that now $\mathbf{s} \rightarrow 1100001 \leftrightarrow 1000101$. It can be seen that a new attractor cycle is formed, and its constituent states have x_1=1, which is undesirable.

Option 2 : Change the state transition to $\mathbf{s} \rightarrow \mathbf{u}_2$. This can be done through $f_3 \rightarrow f_3^{(2)}$. As a result, the other three attractors remain the same, and a new attractor cycle, $\{1000011, 0010001\}$, will be formed, and the first constituent state is undesirable (x_1=1).

Option 3 : Change the state transition to $\mathbf{s} \rightarrow \mathbf{u}_3$. This can be done through $f_4 \rightarrow f_4^{(2)}$. As a result, the other 3 attractors are still the same, while $\mathbf{s}$ disappears. This is a viable option.

Option 4 : Change the state transition to $\mathbf{s} \rightarrow \mathbf{u}_4$ by $f_5 \rightarrow f_5^{(2)}$. The result is a new undesirable attractor cycle, $\{\mathbf{s}, 1000101\}$, while the other 3 attractors do not change.

Option 5 : Change the state transition to $\mathbf{s} \rightarrow \mathbf{u}_5$ by $f_6 \rightarrow f_6^{(2)}$. The result is a new attractor 0000011, while the other 3 attractors do not change. This is also a viable option.

(iii) Both Option 3 and Option 5 can eliminate the undesirable attractor without creating a new undesirable one; however, Option 3 does not create any new attractor, while Option 5 creates a new attractor 0000011. By comparison, Option 3 has less impact on the original network, so it is the better solution.

By following the outlined procedure, we are able to eliminate the undesirable attractor associated with high competence of cellular metastasis with a one-bit function perturbation, with no other attractors affected or any new attractor created. Moreover, the perturbation is chosen so that a minimum number of state transitions will be affected. All of these are achieved without exhaustive search, and without any complex computation.

2. Perturbation Identification in a *Drosophila melanogaster* Segmentation Polarity Gene Network

We will now apply the perturbation identification strategy to a *Drosophila melanogaster* segmentation polarity gene network [65]. Consider a Boolean network described by equation (4) of [65]. There are 8 nodes (genes), wg_1, wg_2, wg_3, wg_4, PTC_1, PTC_2, PTC_3 and PTC_3. The functions are defined as follows,

$$\begin{aligned}
wg_1 &= wg_1 \cdot \overline{wg}_2 \cdot \overline{wg}_4, \\
wg_2 &= wg_2 \cdot \overline{wg}_1 \cdot \overline{wg}_3, \\
wg_3 &= wg_1 + wg_3, \\
wg_4 &= wg_2 + wg_4, \\
PTC_1 &= \overline{wg}_2 \cdot \overline{wg}_4 + PTC_1 \cdot \overline{wg}_1 \cdot \overline{wg}_3,
\end{aligned}$$

$$PTC_2 = \overline{wg}_1 \cdot \overline{wg}_3 + PTC_2 \cdot \overline{wg}_2 \cdot \overline{wg}_4,$$

$$PTC_3 = 1,$$

$$PTC_4 = 1.$$

This network has 10 singleton attractors, 00001111 (15), 00010111 (23), 00011111 (31), 00101011 (43), 00101111 (47), 00110011 (51), 01010111 (87), 01011111 (95), 10101011 (171), 10101111 (175) (see [65], Fig. 6). In the parentheses following each attractor are the corresponding decimal numbers.

Among the attractors, 23 and 31 lead to a wild-type pattern and a variant of a wild-type pattern, respectively. 43 and 47 lead to patterns without parasegment. Those patterns are well-known experimentally. 87 and 95 lead to patterns similar to wild-types, and 171 and 175 lead to patterns similar to non-parasegment patterns, but these patterns are not observed experimentally. Assume the network is modified so that the attractors 87, 95, 171 and 175 disappear, leaving 6 attractors. Under the modification, suppose state 87 $\in$ BOA{23}, 95 $\in$ BOA{31}, 171 $\in$ BOA{43}, and 175 $\in$ BOA{47}, each reaching its attractor in one step. Suppose we have no knowledge about other changes in the network. Based on the partial knowledge, can we find out how the network is modified?

States 87 (01010111) and 23 (00010111) differ by the 2nd bit. States 95 (01011111) and 31 (00011111) also differ by the 2nd bit. States 171 and 43 differ by the 1st bit. States 175 and 47 differ by the 1st bit too. According to Proposition 6, perturbations occur on the function for gene wg_2 and on the function for gene wg_1. The former function has genes $wg1$, wg_2 and wg_3 as inputs, and in both states 87 and 95, $(wg_1, wg_2, wg_3) = 010 = \mathbf{a}_3^2$, so the 3rd row of the truth table is flipped. The latter function has genes wg_1, wg_2 and wg_4 as inputs, and in both states 171 and 175, $(wg_1, wg_2, wg_4) = 100 = \mathbf{a}_5^1$, so the 5th row of the truth table is flipped. The new definitions for the two functions are:

$$wg_1 = \begin{cases} wg_1 \cdot \overline{wg}_2 \cdot \overline{wg}_4, & \text{if } (wg_1, wg_2, wg_4) \neq 100, \\ 1 - wg_1 \cdot \overline{wg}_2 \cdot \overline{wg}_4, & \text{if } (wg_1, wg_2, wg_4) = 100, \end{cases}$$

$$wg_2 = \begin{cases} wg_2 \cdot \overline{wg}_1 \cdot \overline{wg}_3, & \text{if } (wg_1, wg_2, wg_3) \neq 010, \\ 1 - wg_2 \cdot \overline{wg}_1 \cdot \overline{wg}_3, & \text{if } (wg_1, wg_2, wg_3) = 010. \end{cases}$$

Simulation results of the new Boolean network with the above modifications agree with our conclusion.

To make the above identification, we do not require complete knowledge of the state transitions. This is because even a 1-bit difference in a single function can result in 2^{n-k_i} changes in state transitions, and when $n > k_i$, there is a lot of redundant information.

E. Conclusion

This chapter provides several analytical results concerning the perturbation of functions in a Boolean network, and in doing so extends previous work on network stability in a new direction: the effect of structural perturbation on network stability and long-term behavior. It shows how to apply the analytical results to control the stationary probabilities of states in a PBN and has applied the method to intervene in a WNT5A network to avoid high competence metastatic cellular states. It shows to to use the analytical results to identify function perturbations when changes in network behavior are observed and has applied this method to identify structural changes made in a drosophila polarity gene network. The application procedures do not require exhaustive searches or complex computation.

CHAPTER VI

SUMMARY

In this dissertation, the following three issues in genetic regulatory networks are studied:

- Inferring probabilistic Boolean models for genetic regulatory networks from the gene expression profiles;
- Optimizing the structure of genetic regulatory network models based on some constraints;
- Analyzing the impact of function perturbations in the Boolean models of genetic regulatory networks, in terms of both the system dynamics and the long-term behavior.

In the first issue, when inferring the rule-based models (Boolean networks) from gene expression profiles, the optimal predictor function for a target gene is usually not perfect owing to the inconsistencies in the data set, such that even for the same combination of input values, the output can be different. Therefore, a consistency-based inference method is developed to design a context-sensitive probabilistic Boolean model, which consists of a family of Boolean networks (known as contexts), each governed by a set of deterministic regulatory functions. The existence of alternative function outputs can be interpreted as the result of random switches between the contexts, while within each context, a function output is uniquely defined.

For the second issue, when the consistency-based inference method is applied, it is often the case that the sample size is not sufficiently large to infer the network fully, such that there are multiple models that all agree with the data under the data consistency requirement. Therefore it is necessary to generalize, that is, to select models, preferably through an optimization criterion. To this end, the connectivity and the physical realization of the regulatory rules are taken into consideration, and algorithms are developed to achieve unique models that are featured by low connectivity and simplified physical (logic) realization.

When it comes to the robustness of Boolean models subject to function perturbations, it should be noted that this issue serves as a middle layer between the inference of genetic regulatory network model and the application of intervention/control in the model. A robust genetic regulatory network model means it is resistant to small perturbations in the network, which is important in the maintenance of key biological functions; however, the opposite of robustness, namely adaptability, can be desirable when the network adapts itself to deal with environmental changes or drug interventions, so as to achieve an advantage in evolution or survival. The impact of function perturbations in genetic regulatory networks is studied in the context of Boolean models, and theoretical results are developed to facilitate a formal analysis on the network state transitions and attractors. Moreover, the theoretical results can be applied to design effective intervention strategies to change the network long-term behavior, or to identify perturbations in the network function, which are potentially useful in genetic network control and diagnosis of genetic malfunction.

REFERENCES

[1] R. F. Hashimoto, S. Kim, 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.

[2] E. R. Dougherty, A. Datta, and C. Sima, "Research issues in genomic signal processing," *IEEE Signal Processing Magazine*, vol. 22, pp. 46–68, 2005.

[3] Wikipedia, "http://en.wikipedia.org/wiki/gene_regulatory_network," Internet resource, accessed April 2007.

[4] H. de Jong, "Modeling and simulation of genetic regulatory systems: a literature review," *Computational Biology*, vol. 9, pp. 67–103, 2002.

[5] E. R. Dougherty, I. Shmulevich, and M. L. Bittner, "Genomic signal processing: the salient issues," *Applied Signal Processing*, vol. 4, pp. 146–153, January 2004.

[6] E. Segal, M. Shapira, A. Regev, D. Pe'er, D. Botstein, D. Koller, and N. Friedman, "Module networks: identifying regulatory modules and their condition-specific regulators from gene," *Nature Genetics*, vol. 34, pp. 166–176, 2003.

[7] H. Li and M. Zhan, "Systematic intervention of transcription for identifying network response to disease and cellular phenotypes," *Bioinformatics*, vol. 22, pp. 96–102, 2006.

[8] P. Sebastiani, M. F. Ramoni, V. Nolan, C. T. Baldwin, and M. H. Steinberg, "Genetic dissection and prognostic modeling of overt stroke in sickle cell anemia," *Nature Genetics*, vol. 37, pp. 435 – 440, April 2005.

[9] K. Basso, A. A. Margolin, G. Stolovitzky, U. Klein, R. Dalla-Favera, and A. Califano, "Reverse engineering of regulatory networks in human B cells," *Nature Genetics*, vol. 37, pp. 382 – 390, April 2005.

[10] E. R. Dougherty and Y. Xiao, "Design of probabilistic Boolean networks under the requirement of contextual data consistency," *IEEE Trans. Signal Processing*, vol. 54, pp. 3603–3613, September 2006.

[11] 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.

[12] 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. Pan, M. J. Schilstra, P. J. 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.

[13] S. A. Kauffman, *The origins of order: self-organization and selection in evolution*, New York: Oxford University Press, 1993.

[14] 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.

[15] Y. Chen S. Kim K. Sivakumar J. Barrera P. Meltzer E. R. Dougherty, M. Bittner and J. Trent, "Nonlinear filters in genomic control," in *Proc. IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing*, 1999, pp. 10–15.

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

[17] A. J. Fornace M. L. Bittner R. Pal, A. Datta and E. R. Dougherty, “Boolean relationships among genes responsive to ionizing radiation in the NCI 60 ACDS,” *Bioinformatics*, vol. 21, pp. 1542–1549, 2005.

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

[19] 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.

[20] S. A. Kauffman, “Metabolic stability and epigenesis in randomly constructed genetic nets,” *Theoretical Biology*, vol. 22, pp. 437–467, 1969.

[21] S. A. Kauffman, “Homeostasis and differentiation in random genetic control networks,” *Nature*, vol. 224, pp. 177–178, 1969.

[22] I. Shmulevich, E. R. Dougherty, and W. Zhang, “From Boolean to probabilistic Boolean networks as models of genetic regulatory networks,” *Proceedings of the IEEE*, vol. 90, pp. 1778–1792, 2002.

[23] S. A. Kauffman, “The large scale structure and dynamics of genetic control circuits: an ensemble approach,” *Theoretical Biology*, vol. 44, pp. 167–190, 1974.

[24] R. Somogyi and C. Sniegoski, “Modeling the complexity of gene networks: understanding multigenic and pleiotropic regulation,” *Complexity*, vol. 1, pp. 45–63, 1996.

[25] N. Friedman, M. Linial, I. Nachman, and D. Pe'er, "Using Bayesian network to analyze expression data," *Journal of Computational Biology*, vol. 7, pp. 601–620, 2000.

[26] N. Friedman and D. Koller, "Being Bayesian about network structure: a Bayesian approach to structure discovery in Bayesian networks," *Machine Learning*, vol. 50, pp. 95–125, 2003.

[27] D. Husmeier, "Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks," *Bioinformatics*, vol. 19, pp. 2271–2282, 2003.

[28] 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.

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

[30] 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.

[31] I. Shmulevich, I. Gluhovsky, R. Hashimoto, E. R. Dougherty, and W. Zhang, "Steady-state analysis of genetic regulatory networks modeled by probabilistic Boolean networks," *Comparative Functional Genomics*, vol. 4, pp. 601–608, 2003.

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

[33] I. Shmulevich, E. R. Dougherty, and W. Zhang, “Control of stationary behavior in probabilistic Boolean networks by means of structural intervention,” *Biological Systems*, vol. 10, pp. 431–445, 2002.

[34] 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.

[35] 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.

[36] T. Akutsu, S. Miyano, and S. Kuhara, “Identification of genetic networks from a small number of gene expression patterns under the Boolean network model,” *Pacific Symp. on Biocomputing*, vol. 4, pp. 17–28, 1999.

[37] T. Akutsu, S. Miyano, and S. Kuhara, “Inferring qualitative relations in genetic networks and metabolic pathways,” *Bioinformatics*, vol. 16, pp. 727–734, 2000.

[38] T. E. Ideker, V. Thorsson, and R. M. Karp, “Discovery of regulatory interactions through perturbation: inference and experimental design,” *Pacific Symp. on Biocomputing*, vol. 5, pp. 302–313, 2000.

[39] S. Liang, S. Fuhrman, and R. Somogyi, “Reveal, a general reverse engineering algorithm for inference of genetic network architectures,” *Pacific Symp. on Biocomputing*, vol. 3, pp. 18–29, 1998.

[40] H. Lahdesmaki, I. Shmulevich, and O. Yli-Haria, "On learning gene regulatory networks under the Boolean network model," *Machine Learning*, vol. 52, pp. 147–167, 2003.

[41] K. Noda, A. Shinohara, M. Takeda, S. Matsumoto, S. Miyano, and S. Kuhara, "Finding genetic network from experiments by weighted network model," *Genome Informatics*, vol. 9, pp. 141–150, 1998.

[42] I. Shmulevich, A. Saarinen, O. Yli-Harja, and J. Astola, "Inference of genetic regulatory networks under the best-fit extension paradigm," in *Computational and Statistical Approaches to Genomics*, W. Zhang and I. Shmulevich, Eds. Boston, MA: Kluwer Academic Publishers, 2002.

[43] E. P. van Someren, L. F. A. Wessels, E. Backer, and M. J. T. Reinders, "Multi-criterion optimization for genetic network modeling," *Signal Processing*, vol. 83, pp. 763–775, 2003.

[44] E. R. Dougherty, S. Kim, and Y. Chen, "Coeffcient of determination in nonlinear signal processing," *Signal Processing*, vol. 80, pp. 2219–2235, 2000.

[45] X. Zhou, X. Wang, and E. R. Dougherty, "Gene prediction using multinomial probit regression with Bayesian gene selection," *Applied Signal Processing*, vol. 4, pp. 115–124, 2004.

[46] 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.

[47] 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. Marin-

cola, C. Gooden, J. Lueders, A. Glatfelter, P. Pollock, J. Carpten, E. Gillanders, D. Leja, K. Dietrich, C. Beaudry, M. Berens, D. Alberts, V. Sondak, N. Hayward, and J. Trent, "Molecular classification of cutaneous malignant melanoma by gene expression profiling," *Nature*, vol. 406, pp. 536–540, 2000.

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

[49] Y. Xiao and E. R. Dougherty, "Optimizing consistency-based design of context-sensitive gene regulatory networks," *IEEE Trans. Circuits and Systems I*, vol. 53, pp. 2431–2437, November 2006.

[50] G. Langholz, A. Kandel, and J. L. Mott, *Foundations of Digital Logic Design*, Singapore: World Scientific, 1998.

[51] W. G. Schneeweiss, *Boolean Functions with Engineering Applications and Computer Programs*, Berlin: Springer-Verlag, 1989.

[52] W. V. Quine, "The problem of simplifying truth functions," *American Mathematical Monthly*, vol. 59, pp. 521–531, October 1952.

[53] E. J. McCluskey, "Minimization of Boolean functions," *Bell System Technical Journal*, vol. 35, pp. 1417–1444, November 1956.

[54] Y. Xiao and E. R. Dougherty, "The impact of function perturbations in Boolean networks," *Bioinformatics*, vol. 23, pp. 1265–1273, May 2007.

[55] M. Aldana and P. Cluzel, "A natural class of robust networks," *PNAS*, vol. 100, pp. 8710–8714, 2003.

[56] I. Shmulevich, H. Lahdesmaki, E. R. Dougherty, J. Astola, and W. Zhang, "The role of certain Post classes in Boolean network models of genetic networks," *PNAS*, vol. 100, pp. 10734–10739, 2003.

[57] P. Ramo, J. Kesseli, and O. Yli-Harja, "Stability of functions in Boolean models of gene regulatory networks," *Chaos*, vol. 15, pp. 034101, 2005.

[58] S. A. Kauffman, C. Peterson, B. Samuelsson, and C. Troein, "Genetic networks with canalyzing Boolean rules are always stable," *PNAS*, vol. 101, pp. 17102–17107, 2004.

[59] X. Qu, M. Aldana, and L. P. Kadanoff, "Numerical and theoretical studies of noise effects in the Kauffman model," *Journal of Statistical Physics*, vol. 109, pp. 967–986, December 2002.

[60] C. Gershenson, S. A. Kauffman, and I. Shmulevich, "The role of redundancy in the robustness of random Boolean networks," in *Artificial Life X, Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems*, available online at http://arxiv.org/abs/nlin/0511018, 2006.

[61] M. Chaves, R. Albert, and E. D. Sontag, "Robustness and fragility of Boolean models for genetic regulatory network," *Journal of Theoretical Biology*, vol. 235, pp. 431–449, 2005.

[62] A. Datta, A. Choudhary, M. L. Bittner, and E. R. Dougherty, "Control in Markovian genetic regulatory networks," *Machine Learning*, vol. 52, pp. 169–191, 2003.

[63] R. Pal, A. Datta, and E. R. Dougherty, "Optimal infinite horizon control for probabilistic Boolean networks," *IEEE Trans. Signal Processing*, vol. 54, pp.

2375–2387, 2006.

[64] T. Akutsu, M. Hayashida, W-K. Ching, and M. K. Ng, "Control of Boolean networks: hardness results and algorithms for tree structured networks," *Journal of Theoretical Biology*, vol. 244, pp. 670–679, 2002.

[65] R. Albert and H. G. Othmer, "The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in Drosophila melanogaster," *Journal of Theoretical Biology*, vol. 223, pp. 1–18, 2003.

APPENDIX A

SUMMARY OF CONSISTENCY-BASED GENETIC REGULATORY NETWORK DESIGN ALGORITHM*

Assume there are n genes in the genetic regulatory network, each taking a value from the set $\{0, 1\}$. Denote the n dimensional binary vector space as $S = \{0, 1\}^n$.

Algorithm 4 ***Consistency-based PBN Design***

Step 1 *Let the data set be $\mathcal{G} = \{\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^{n'}\} \subseteq S$ ($n' \leq 2^n$) which contains n' distinctive gene expression profiles (a profile may occur multiple times but only one copy is included in $\mathcal{G}$) of genes g_1, $\cdots$, g_n. The frequency (or number of copies) of profile $\mathbf{x}^k$ is denoted by $\nu(\mathbf{x}^k)$.*

Step 2 *Without loss of generality, let g_n be the target gene. Suppose there exist l pairs of data, $\mathbf{x}^{k_{10}}, \mathbf{x}^{k_{11}}, \cdots, \mathbf{x}^{k_{l0}}, \mathbf{x}^{k_{l1}}$, such that the data in each pair differ only on the value of target gene, namely,*

$$\mathbf{x}^{k_{h0}} = [x_{k_h 1}, \cdots, x_{k_h(n-1)}, 0],$$

and

$$\mathbf{x}^{k_{h1}} = [x_{k_h 1}, \cdots, x_{k_h(n-1)}, 1],$$

$(h = 1, \cdots, l)$. Then $m = 2^l$ functions can be defined for g_n, namely,

$$g_n = \phi_j(g_1, \cdots, g_{n-1}), \quad j = 1, \cdots, m.$$

*© 2006 IEEE. Reprinted, with permission, from *IEEE Transactions on Signal Processing*, see [10] for complete publication information.

Each function is assigned a probability

$$\mathrm{P}\{\phi_j\} = \prod_{h=1}^{l} p_{hb},$$

where $p_{hb} = \nu(\mathbf{x}^{k_{hb}})/[\nu(\mathbf{x}^{k_{h0}}) + \nu(\mathbf{x}^{k_{h1}})]$, $b = 0$ *if* ϕ_j *is consistent with* $\mathbf{x}^{k_{h0}}$, $b = 1$ *if it is consistent with* $\mathbf{x}^{k_{h1}}$, *and* $\mathrm{P}\{\phi_1\} + \cdots + \mathrm{P}\{\phi_m\} = 1$. *Apart from the* l *pairs,* ϕ_j *is consistent with the remaining data in* $\mathcal{G}$. *If* $l = 0$, *then only one function is defined and it is consistent with all data in* $\mathcal{G}$.

Step 3 *Apply Step 2 to each gene in turn. If there are* $m_1, m_2, \cdots, m_n$ *functions for genes* $g_1, g_2, \cdots, g_n$ *respectively, then by choosing a function for each gene and making all possible combinations, we obtain* $m_1 m_2 \cdots m_n$ *network functions and associated selection probabilities. Therefore, there are* $r = m_1 m_2 \cdots m_n$ *contexts.*

As seen from the algorithm, the size of (the truth table of) a Boolean functions is determined by 2^n, and there are altogether $\sum_{i=1}^{n} m_i$ Boolean functions. Therefore the complexity of the contextual design is $O(2^n \sum_{i=1}^{n} m_i)$; it is noteworthy that $\sum_{i=1}^{n} m_i$ depends primarily on the relations among data, but cannot be solely accounted for by either the number of genes n or data set size n' alone.

APPENDIX B

A PROFILE REDUCTION ALGORITHM*

To reduce the number of contexts arising from excessive data heterogeneity or from observation noise, we propose a filtering method based on Hamming-distance between the profiles in the data set. Recall that the *Hamming distance* is the number of different bits between two equal-length binary strings.

Algorithm 5 ***Profile Reduction***

1. *Given a batch of n-gene (binary) expression profiles, split them into a single-copied data group $\mathcal{G}_1$ and a multiple-copied data group $\mathcal{G}_2$. Set $i := 1, j := 1, k := 1$.*
2. *Select the i-th profile ρ_{1i} from group $\mathcal{G}_1$ and compare it with the j-th profile ρ_{2j} in $\mathcal{G}_2$. If their Hamming distance does not exceed k, merge ρ_{1i} into ρ_{2j}, i.e., delete ρ_{1i} and increase the number of copies of profile ρ_{2j} by 1. If ρ_{1i} does not merge with ρ_{2j}, let $j := j + 1$, and repeat until either ρ_{1i} is deleted or the end of $\mathcal{G}_2$ is reached.*
3. *Let $i := i + 1$, and repeat step 2 until the end of $\mathcal{G}_1$.*
4. *Let $k := k+1$, if k is less than or equal to a prescribed value K ($K \geq 1$), repeat steps 2 and 3.*

Selection of K in the algorithm is a heuristic decision and contingent on n. The reason for introducing the loop $k = 1, \cdots, K$ is that we always merge a profile to its nearer neighbor with higher priority.

*© 2006 IEEE. Reprinted, with permission, from *IEEE Transactions on Circuits and Systems I*, see [49] for complete publication information.

VITA

Name:	Yufei Xiao
Address:	Department of Electrical and Computer Engineering
	Mail Stop 3128
	Texas A&M University
	College Station, TX 77843-3128
Email Address:	fei@tamu.edu
Education:	B.S., Zhejiang University, Hangzhou, Zhejiang, P. R. China
	M.S., University of Virginia, Charlottesville, VA
	Ph.D., Texas A&M University, College Station, TX