GENOMIC REGULATORY NETWORKS, REDUCTION MAPPINGS AND CONTROL

A Dissertation

by

NOUSHIN GHAFFARI

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

May 2012

Major Subject: Computer Engineering

GENOMIC REGULATORY NETWORKS, REDUCTION MAPPINGS AND CONTROL

A Dissertation

by

NOUSHIN GHAFFARI

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

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Edward R. Dougherty
Committee Members,	Ivan Ivanov
	Aniruddha Datta
	Byung-Jun Yoon
Head of Department,	Costas N. Georghiades

May 2012

Major Subject: Computer Engineering

ABSTRACT

Genomic Regulatory Networks,
Reduction Mappings and Control. (May 2012)
Noushin Ghaffari, B.S., Tehran Central Azad University;
M.S., University of Houston - Clear Lake
Chair of Advisory Committee: Dr. Edward R. Dougherty

All high-level living organisms are made of small cell units, containing DNA, RNA, genes, proteins etc. Genes are important components of the cells and it is necessary to understand the inter-gene relations, in order to comprehend, predict and ultimately intervene in the cells' dynamics. Genetic regulatory networks (GRN) represent the gene interactions that dictate the cell behavior. Translational genomics aims to mathematically model GRNs and one of the main goals is to alter the networks' behavior away from undesirable phenotypes such as cancer.

The mathematical framework that has been often used for modeling GRNs is the probabilistic Boolean network (PBN), which is a collection of constituent Boolean networks with perturbation, BN_p. This dissertation uses $BN_p s$, to model gene regulatory networks with an intent of designing stationary control policies (CP) for the networks to shift their dynamics toward more desirable states. Markov Chains (MC) are used to represent the PBNs and stochastic control has been employed to find stationary control policies to affect steady-state distribution of the MC. However, as the number of genes increases, it becomes computationally burdensome, or even infeasible, to derive optimal or greedy intervention policies.

This dissertation considers the problem of modeling and intervening in large GRNs. To overcome the computational challenges associated with large networks, two approaches are proposed: first, a reduction mapping that deletes genes from the

network; and second, a greedy control policy that can be directly designed on large networks. Simulation results show that these methods achieve the goal of controlling large networks by shifting the steady-state distribution of the networks toward more desirable states.

Furthermore, a new inference method is used to derive a large 17-gene Boolean network from microarray experiments on gastrointestinal cancer samples. The new algorithm has similarities to a previously developed well-known inference method, which uses seed genes to grow subnetworks, out of a large network; however, it has major differences with that algorithm. Most importantly, the objective of the new algorithm is to infer a network from a seed gene with an intention to derive the Gene Activity Profile toward more desirable phenotypes. The newly introduced reduction mappings approach is used to delete genes from the 17-gene GRN and when the network is small enough, an intervention policy is designed for the reduced network and induced back to the original network. In another experiment, the greedy control policy approach is used to directly design an intervention policy on the large 17-gene network to beneficially change the long-run behavior of the network.

Finally, a novel algorithm is developed for selecting only non-isomorphic BNs, while generating synthetic networks, using a method that generates synthetic BNs, with a prescribed set of attractors. The goal of the new method described in this dissertation is to discard isomorphic networks.

To Love of My Life, Omid

ACKNOWLEDGMENTS

My graduate studies at Texas A&M University have been a wonderful personal and professional experience. I could not expect a mentor and advisor better than Dr. Dougherty. His vast knowledge and experience, invaluable vision, careful observation of details, managing skills, support, personal attention to each individual, care for goal-oriented thinking, intellectual guidance and generous time taught me more than I can put in words. I deeply appreciate all I have learned from him about my research interests and also about life. I would like to thank Dr. Ivanov for all his guidance, support, advice, valuable suggestions and encouragement. He inspired me tremendously through the preparation of my articles and dissertation. The fruitful discussions with him greatly assisted me in developing my ideas.

I am very grateful to my committee members Dr. Datta and Dr. Yoon. Their supports, encouragements, courses and advices improved my knowledge, and particularly enhanced my dissertation.

I extend many thanks to my colleagues at GSP lab. I also would like to thank Dr. Charles D. Johnson at AgriLife Genomics and Bioinformatics Services for his assistance, flexibility and support in the last year of my study in the PhD program.

I am in a life-long debt to my family, especially my parents, and my sister and brother. I cannot thank them enough for their unconditional love, encouragement, inspiration and support throughout my entire life.

Most importantly, I would like to express my deepest gratitude to my love, Omid. He loved me and supported me in all the good/bad days. He believed in me and gave me the strength to dream big. Without him, I could not complete my graduate studies and achieve many other goals in my life. I dedicate this dissertation to him.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

TABLE Page

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

A. Translational genomics, toward personalized medicine

The building blocks of all living organisms are small units called cells. Cells are very complex dynamical systems, much more complex than man-made systems, with many components, inputs, outputs, feedback signals, stress management mechanisms etc and cells' components interact to carry out a variety of functionalities. One goal of *systems biology* is to understand such a phenomenon, in particular, this field aims to study the cells' behavior with models that represent the cells' dynamics closely, while having tractable mathematical complexity. Genes play an important role in regulating cells and it is important to understand their functions and interrelations. Genomic Regulatory Networks (GRNs) model the interactions among genes that dynamically determine the cell behavior. One of the main objectives of modeling GRNs is to alter the behavior of the system toward more desirable phenotypes, i.e. disease-free stages.

The cells' behavior is mainly determined by the time evolution of the their Gene Activity Profile (GAP), i.e. the gene expression level of all the genes within the network. In recent years, advancements in technologies such as microarrays and Next Generation Sequencing (NGS), have made it possible to measure the expression levels of thousands of genes simultaneously, with relatively low cost. Having a snapshot of the genomic signals from the patient samples, provides the possibility of delivering *personalized medicine*. The field of *translational genomics* involves modeling genomic systems with the ultimate goal of deriving therapeutic techniques. Figure 1 shows the key steps in deriving personalized treatment, based on an individual's

---

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

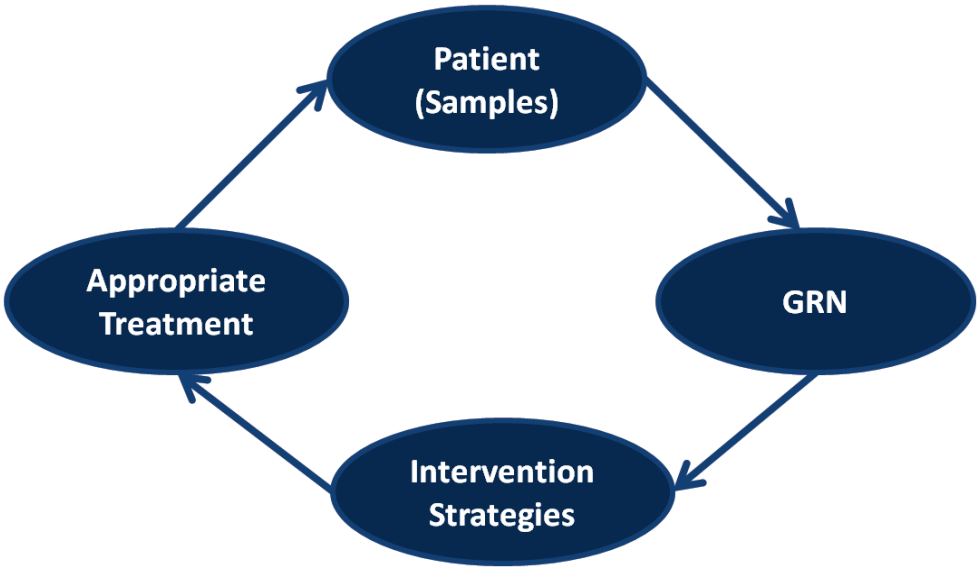

Fig. 1. Personalized Medicine

measurements. The process starts by taking the samples from a patient, constructing the model that represents his/her GRN relatively closely, deriving the personalized intervention strategy and finally prescribing the appropriate treatment in terms of medicine, chemo-therapy, radiation-therapy etc.

Constructing gene regulatory networks that represent the gene interactions inside the cells is very important, because it is believed that the time evolution of GAPs and phenotypes are related. As an example, a previous study discovered a two-gene classifier that accurately identifies two different forms of sarcoma: gastrointestinal stromal tumor (GIST) and leiomyosarcomas (LMSs) [1], based on the expression level of genes *OBSCN* and *C9orf65*. The two different phenotypes require different treatments, and thus, it is important to classify the patients' samples correctly. Additionally, in a recent work, the gene *WNT5A* that is known to be associated with increased metastatic melanoma [2] was used in the respective model of the GRN

to shift the network toward a desired phenotype, which is down-regulated *WNT5A*, using another gene: *pirin* that is a strong predictor of *WNT5A* expression.

B. Modeling genomic regulatory networks

One of the widely used mathematical frameworks for modeling GRNs is the Probabilistic Boolean Networks [3]. Each PBN is a collection of Boolean Networks with a perturbation, BN_p, where the perturbation term p is a small probability of each gene randomly flipping its value [3]. Gene expression levels inside the cells can be coarsely quantified to be under/over expressed, represented as 0/1 in the PBN model. The dynamics of a PBN are represented by its associated Markov Chain. The flipping probability p makes the MC ergodic, and therefore its steady-state distribution (SSD) exists. The ultimate goal of modelings GRNs via PBNs is to beneficially alter their long-run behavior toward more desirable phenotypes such as cancer-free stages.

From a theoretical standpoint, to change the steady-state behavior of a PBN, one can always derive the optimal control policy [4, 5], using dynamic programming techniques. However, that task requires extensive computations [6, 7] and thus the application of the optimal control policy is limited to networks with a small number of genes. As an alternative to the optimal intervention, greedy control approaches such as *Mean-First-Passage-Time (MFPT-CP)* and *Steady-State Distribution (SSD-CP)* were proposed [8, 9]. They are often close approximations to the optimal policy. However, as the number of genes increases in the network, even these greedy control policies cannot be designed for the networks with many genes.

C. Reduction mappings and control policy for large networks

This dissertation focuses on the problem of modeling and intervening large $BN_p s$. Appendix B discusses reduction in other contexts and compares them with the proposed methods. Herein, two major solutions are described for controlling large GRNs:

1. Reduction mappings by sequentially deleting genes from large networks and then inducing the control policy designed on the reduced network back to the original network

2. A procedure for designing control policy directly on large networks

In the first approach, a reduction mapping framework is designed for Boolean network with perturbation. This algorithm employs the coefficient of determination (CoD) [10] to choose genes for deletion, utilizes the collapsing heuristic to construct the wiring of the reduced network, designs a control policy on the reduced network and finishes with a procedure to induce that control policy on the original network. The overall procedure is referred to as *CoD-Reduce*. The CoD measures how a set of random variables improves the prediction of a target variable, relative to the best prediction in the absence of any conditioning observation. The performance of the algorithm is evaluated by its effects on the SSD of the network and on how well it approximates the stationary control designed on the full network. The algorithm is formulated for Boolean networks [11] with perturbation; however, since a binary context-senstive probabilistic Boolean network (PBN) [12][13] is a of collection Boolean networks with perturbation endowed with a selection probability structure (and a general PBN is a collection of more finely quantized versions of BNs), the algorithm can be applied to a PBN by applying it to each constituent Boolean network for the same gene, thereby reducing the PBN. The efficacy of *CoD-Reduce* is demonstrated on networks of 10

genes or less, where it is possible to compare the steady-state shifts of the induced and original policies (because the latter can be derived), and by applying it to a 17-gene gastrointestinal network where it is shown that there is substantial beneficial steady-state shift. The *CoD-Reduce* algorithm and the corresponding simulation studies are provided in Chapter III.

As the second approach, a novel methodology is developed for deriving large $BN_p s$ toward more desirable states. The method relies on the predictive power of a small group of genes, which includes the *control* gene, for predicting the *target* gene and designs a stationary control policy that alters the SSD of the model. The algorithm is designed for the specific class of networks where there is a *path* from the *control* to the *target* gene - an assumption which has a natural interpretation in terms of the biochemical regulatory pathways present in cells. This method simplifies the procedure of designing the stationary control policy and eliminates the need to have a complete knowledge about the state transition matrix of the Markov chain. Most importantly, the new algorithm can be used to design a stationary control policy directly on large networks without deleting any genes/states. This novel algorithm is called *CoD-CP* because the *CoD* is the main tool and *CP* stands for control policy. The *CoD-CP* uses the marginal probabilities of the individual genes obtained from the SSD of the network to calculate the *CoD*s. Simulation experiments show that in small networks, where it is possible to derive the currently available greedy MFPT-CP [8] and SSD-CP [9] policies, *CoD-CP* achieves a similar performance. Most importantly, when the size of the network is large and MFPT-CP or SSD-CP cannot be designed directly on the original model, *CoD-CP* is easily constructed and applied to the network without any reduction mappings and induction of the CP from the reduced network back to the original model. Chapter IV discusses the *CoD-CP* algorithm and its corresponding simulation results.

D. Avoiding isomorphic synthetic GRNs

In order to study different intervention approaches for GRNs, it is essential to generate synthetic networks to have large number of samples to study the properties of the model. The synthetic networks are usually generated with a pre-specified set of criteria, thus, their characteristics are known, which enables scientists to make meaningful conclusions about the networks. It is important to select the major characteristics of the real networks and generate the synthetic networks according to those specifications. As an example, the attractors are important components of the networks and play a crucial role in the long-run behavior of the system. A previous study, developed a method for generating synthetic Boolean networks with a prescribed set of attractors [14], in order to impose the attractor structure of a real network in the generation of the synthetic networks. This algorithm is widely used for generating synthetic Boolean networks, mainly for studying the intervention approaches for PBNs. One important issue while using this algorithm is the possibility of generating *isomorphic* BNs. The input of the algorithm is a set of prescribed attractors and it is possible that the output BNs can be mapped to one another by relabeling their genes. These networks are called *isomorphic*. This dissertation proposes an algorithm for eliminating isomorphic networks to ensure that all the synthetic networks used in a study are *non-isomorphic*. The new algorithm highly relies on enforcing restrictions on the attractor structure of the network and the states within a basin of each attractor. Definition of the isomorphic BNs and related material are discussed in Chapter V.

E. Inference of GRNs, with an intent for intervention

The main objective of the current dissertation is to control large GRNs. In order to test the methods introduced for large networks, it was necessary to have a mechanism to infer a large real-world derived Boolean network. This method needs to infer the GRNs with an intention of intervening the networks' behavior to avoid undesirable phenotypes. The problem of inferring networks had been previously studied by Hashimoto et. al. with a method referred to as seed-growing algorithm [15]. The objective of the seed-growing algorithm is to generate subnetworks with tight interconnection and outputs directed graphs. The idea behind this algorithm is to start with a set of genes that are important for the phenotype of interest, called seed genes, and adjoin genes sequentially to generate subnetworks that are strongly connected within the subnetwork.

This dissertation introduces a new algorithm that infers networks form the gene expression measurements with the ultimate goal of controlling the model's dynamics. The algorithm, similarly to the well-known seed-growing method, starts with a small set of important genes, usually only the *target* gene, which is highly related to the phenotype of interest. However, the main objective of the new algorithm is to infer networks that can be controlled via available intervention policies, and thus, it differs from the seed-growing algorithm in its steps and the output. The two components of the method that are similar to the seed-growing algorithm are: first, starting from an important seed gene(s), and second, incorporation of the strength of gene connectivity into the inference method. This new algorithm is called *CoD-Control-Embedded-inference (CoD-CE-Inference)*, to reflect its similarity to the seed-growing method in using CoD, and also emphasizing its intention of controlling the inferred network. The *CoD-CE-Inference* starts by a target gene, then in each step, adds one

gene to the network that is strongly connected to at least one of the genes inside the network. The CoD is used to measure the strength of gene connections. After adding a new gene, the wiring of the network is updated using all the genes within the network, including the recently added gene. And finally, the algorithm generates the truth table of the network and updates it after addition of a new gene. In summary, the seed-growing algorithm and the *CoD-CE-Inference* have four major differences: first, *CoD-CE-Inference* considers the ultimate goal of controlling the GRNs via a stationary control policy, therefore, each step of the process is designed to assist in achieving this ultimate goal, while the seed-growing algorithm is more concerned about the topology of the network in the context of graph-theory; second, *CoD-CE-Inference* generates the truth table of the GRN; third, *CoD-CE-Inference* re-wires the network after adding any new gene, meaning that adding each gene can affect the entire network, by changing predictors of any given gene; and finally, *CoD-CE-Inference* generates a unique BN, originated from the target gene, but the seed-growing algorithm generates many networks that can be very similar or very different.

The *CoD-CE-Inference* algorithm is used to infer two large 17-gene BN from *gastrointestinal cancer microarray* dataset introduced in [1]. These network have different seed genes and are used to demonstrate the methods described within this dissertation for controlling large GRNs: *CoD-Reduce* and *CoD-CP*. The *CoD-CE-Inference* algorithm and its related experiments are represented in Chapter VI.

CHAPTER II

BACKGROUND

This chapter provides the background material essential for understanding the methods developed by this dissertation. *Boolean networks* are chosen as the model for representing genomic regulatory networks. The *coefficient of determination* is widely used by the new methods, e.g. the *reduction mappings* and the *greedy control policy for large networks*. A previously proposed method for generating Boolean networks with *prescribed attractor sets*, ia used for generating *synthetic* networks; and one chapter of this dissertation is dedicated to selecting *non-isomorphic* synthetic networks. Additionally, inferring GRNs from *seed genes* is previously addressed; and this dissertation proposes a new method that is comparable with the *seed-growing algorithm*. Finally, the two *greedy control polices* described in this chapter are used for comparing the performance of the newly introduced algorithms.

1. Boolean Networks

A *Boolean network (BN) with perturbation p*, $BN_p = (V, \mathbf{f})$, on n genes is defined by a set of nodes $V = \{x_1, ..., x_n\}$ and a vector of Boolean functions $\mathbf{f} = [f^1, ..., f^n]$. The variable $x_i \in \{0, 1\}$ represents the expression level of gene i, with 1 representing high and 0 representing low expression [3]. The regulatory rules between genes are represented by $\mathbf{f}$. At every time step, the value of x_i is predicted by the values of a set, W_i, of genes at the previous time step, based on the regulatory function f^i. $W_i = \{x_{i_1}, ..., x_{i_{k_i}}\}$ is called the *predictor set* and the function f^i is called the *predictor function* of x_i. A state of the BN_p at time t is a vector $\mathbf{s} = (x_1(t), ..., x_n(t)) \in \{0, 1\}^n$, also called *Gene Activity Profile (GAP)*, and the *state space* of the BN_p is the collection S of all states of the network. The perturbation parameter $p \in (0, 1]$

models random gene mutations, i.e. at each time point there is a probability p of any gene changing its value uniformly randomly. The underlying model of a BN_p is a finite Markov chain and its dynamics are completely described by its $2^n \times 2^n$ state transition matrix (STM), $P = (p(\mathbf{s}_i, \mathbf{s}_j))_{i,j=1}^{2^n}$, where $p(\mathbf{s}_i, \mathbf{s}_j)$ is the probability of the chain undergoing the transition from the state $\mathbf{s}_i$ to the state $\mathbf{s}_j$. For n genes, the Markov chain has $N = 2^n$ states and the collection of all the states is called *State Space: S*. The perturbation probability p makes the chain ergodic and therefore it possesses a steady-state probability distribution π which satisfies [16]:

$$\pi = \pi P \tag{2.1}$$

The *Truth table (TT)* of the network is a $N \times n$ matrix, where all the *states* $\in S$ are the rows and columns represent the predictor functions. The TT can be used to derive the STM and SSD of the network. The Markov chain starts the transitions from an *initial state* and continues to transition from a state to another state until it eventually enters a set of states where it cycles forever. These set of states are called *attractors* of the network. If there is only one state in the attractor set, then it is called *singleton attractor*. If there are more than one state within the attractor set, the network posses the *cyclic attractor*. Each network can have more than one attractor set. *Non-attractor* states of the network are called *transient* states. Each transient state belongs to one attractor set, because all the transitions eventually end within an attractor. The number of transitions needed for a state to reach its attractor set determines its *level*. In general the network can be partitioned using these levels: all the states in a level need exactly the same number of transitions to reach their corresponding attractor.

2. Coefficient of Determination (CoD)

The *coefficient of determination* (CoD) measures how a set of random variables improves the prediction of a target variable, relative to the best prediction in the absence of any conditioning observation [10]. Let $\mathbf{X} = (X_1, X_2, ..., X_n)$ be a vector of predictor binary random variables, Y a binary target variable, and f a Boolean function such that $f(\mathbf{X})$ predicts Y. The Boolean case is represented and used in this dissertation, however, the basic definition for $CoD_X(Y)$ is not so restricted [10]. The mean-square error (MSE) of $f(\mathbf{X})$ as a predictor of Y is the expected squared difference, $E[|f(\mathbf{X}) - Y|^2]$. Let $\varepsilon_{opt}(Y, \mathbf{X})$ be the minimum MSE among all predictor functions $f(\mathbf{X})$ for Y and $\varepsilon_0(Y)$ be the error of the best estimate of Y without any predictors. The CoD is defined as:

$$CoD_{\mathbf{X}}(Y) = \frac{\varepsilon_0(Y) - \varepsilon_{opt}(Y, \mathbf{X})}{\varepsilon_0(Y)}. \tag{2.2}$$

Letting $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{2^n}$ denote the 2^n possible values for $\mathbf{X}$, running from $(0, 0, ..., 0)$ to $(1, 1, ..., 1)$, the relevant quantities are given by

$$\varepsilon_{opt}(Y, \mathbf{X}) = \sum_{j=1}^{2^n} P(\mathbf{X} = \mathbf{x}_j) \min[P(Y = 0|\mathbf{x}_j), P(Y = 1|\mathbf{x}_j)] \tag{2.3}$$

and

$$\varepsilon_0(Y) = \min[P(Y = 0), P(Y = 1)] \tag{2.4}$$

[10]. The CoD can be used to measure the strength of the connection between a target gene and its predictors and has been used since the early days of DNA microarray analysis to characterize the nonlinear multivariate interaction between genes [17]. More recently, CoD was used to characterize canalizing genes [18] and contextual genomic regulation [19].

3. Mean-first-passage-time control policy (MFPT-CP)

For beneficially changing the long-run behavior of a GRN, optimal intervention is usually formulated as an optimal stochastic control problem [6]. The intervention is achieved by toggling the value of a single control gene g, and stationary control policies $\mu_g : S \to \{0, 1\}$ are based on g. The values 0/1 are interpreted as off/on for the application of the control: 1 meaning that the current value of g is toggled, and 0 meaning that no control is applied.

The *mean-first-passage-time* ($MFPT$) policy is based on the comparison between the MFPTs of a state $\mathbf{s}$ and its *flipped* state $\tilde{\mathbf{s}}^g$ [8]. The flipped state $\tilde{\mathbf{s}}^g$ has the same binary values for all its genes, expect the value of gene g: if $g = 0$ in $\mathbf{s}$, then $g = 1$ in $\tilde{\mathbf{s}}^g$ or vice versa. When considering intervention, the state space S can be partitioned into desirable (D) and undesirable (U) states according to the expression values of a given *target* gene x, that is the leftmost gene in the state's binary representations, i.e. $x_1 = x$, $\mathbf{s} = (x, x_2, ..., x_n)$, and the desirable states correspond to the value $x = 0$. With these assumptions, the state transition matrix P of the network can be written as

$$P = \begin{pmatrix} P_{DD} & P_{DU} \\ P_{UD} & P_{UU} \end{pmatrix}. \tag{2.5}$$

Using this representation, one can compute the mean-first-passage-time vectors required for a state $\mathbf{s}$ to reach the boundary between desirable and undesirable states. Computation of these average times is performed in the time scale used for the state transitions of the network. If one uses the states of the network to index the components of the vectors in the 2^n dimensional Euclidean space $\mathbb{R}^{2^n}$, then one can form the vectors K_U and K_D that contain the mean-first-passage-times needed for the states

in D and U to reach the undesirable and the desirable states, respectively. The two vectors K_U and K_D are of dimension 2^{n-1}, and, according to a well-known result form the theory of Markov chains [16], are given as solutions to the following system of linear equations:

$$K_U = e + P_{DD}K_U \tag{2.6}$$

$$K_D = e + P_{UU}K_D \tag{2.7}$$

where e denotes the vector of dimension 2^{n-1} with all of its co-ordinates equal to 1.

To understand the intuition behind the MFPT-CP algorithm it is important to notice that, because the control gene g is different from the target gene, every state $\mathbf{s}$ belongs to the same class of states, D or U, as its flipped state $\tilde{\mathbf{s}}^g$. With this in mind, if a desirable state $\mathbf{s}$ reaches U on average faster than $\tilde{\mathbf{s}}^g$, it is reasonable to apply control and start the next network transition from its flipped state $\tilde{\mathbf{s}}^g$. Thus, the design of the stationary MFPT-CP is based on the differences $K_D(\mathbf{s}) - K_D(\tilde{\mathbf{s}}^g)$ and $K_U(\tilde{\mathbf{s}}^g) - K_U(\mathbf{s})$. To avoid too frequent application of control, the MFPT-CP algorithm uses a tuning parameter $\gamma > 0$, and these differences are compared to the value of γ, which is related to the cost of applying control.

The MFPT concept could be used in two different ways to design the intervention strategy. The first approach is called "model-dependent" and needs the state transition matrix of the Markov Chain. The time-course measurements can be used to estimate the transition probabilities for all states. Then the STM is used to find the K_U and K_D vectors to design the control policy. In the second approach, called "model-free," the MFPTs are directly estimated from the time-course data and the inference of the STM is skipped. The model-dependent MFPT-CP is used in the simulation studies of this dissertation. The synthetic networks are generated using

the method proposed in [14]. Using these networks, it is possible to generate the STM of the networks and derive the K_U and K_D vectors.

4. Steady-state distribution control policy (SSD-CP)

The *steady-state-distribution* control policy (*SSD-CP*) [9] uses the steady-state distribution of a perturbed Markov chain given in [20] to quantify the shift in the steady-state mass after applying possible controls. A perturbation in the logic defining the Boolean network changes the original transition probability matrix P and steady-state distribution π to $\tilde{P}$ and $\tilde{\pi}$, respectively. In [20], the fundamental matrix, Z, is used to represent $\tilde{\pi}$ in terms of π. $Z = [I - P + e\pi^T]^{-1}$, where T denotes transpose and e is a column vector whose components are all unity [21]. For a *rank-one perturbation*, the perturbed Markov chain has the transition matrix $\tilde{P} = P + ab^T$, where a, b are two arbitrary vectors satisfying $b^T e = 0$, and ab^T represents a rank-one perturbation to the original Markov chain P. In the special case where the transition mechanisms before and after perturbation differ only in one state, say state $\mathbf{k}$,

$$\tilde{\pi}^T = \pi^T + \frac{\pi^T e(\mathbf{k})}{1 - b^T Z e(\mathbf{k})} b^T Z = \pi^T + \frac{\pi(\mathbf{k})}{1 - \beta(\mathbf{k})} \beta^T \tag{2.8}$$

where $\beta^T = b^T Z$ and $e(\mathbf{k})$ is the elementary vector with a 1 in the $\mathbf{k}$th position and 0s elsewhere [20, 21, 22]. For the ith state,

$$\tilde{\pi}_i = \pi_i + \frac{\pi_k \beta_i}{1 - \beta_k}. \tag{2.9}$$

The results for these special cases can be extended to arbitrary types of perturbations so that it is possible to compute the steady-state distributions of arbitrarily perturbed Markov chains in an iterative fashion [20]. Therefore, it is possible to use the perturbed MC and its SSD to derive the SSD-CP. Let $\tilde{\mathbf{s}}^g$ be the flipped state (with respect to control gene g) corresponding to state $\mathbf{s}$ (as with MFPT-CP), let π_U

be the original steady-state mass of the undesirable states and let $\tilde{\pi}_U(\mathbf{s})$ and $\tilde{\pi}_U(\tilde{\mathbf{s}}^g)$ denote the steady-state masses of the undesirable states resulting from altering the original state transition matrix by changing the starting state for the next transition from $\mathbf{s}$ to $\tilde{\mathbf{s}}^g$ and from $\tilde{\mathbf{s}}^g$ to $\mathbf{s}$, respectively. The SSD-CP policy is defined on pairs of states, $\mathbf{s}$ and $\tilde{\mathbf{s}}^g$, in the following manner: if both $\tilde{\pi}_U(\mathbf{s})$ and $\tilde{\pi}_U(\tilde{\mathbf{s}}^g)$ are larger than π_U, then control is applied to neither; otherwise, if $\tilde{\pi}_U(\mathbf{s}) \leq \tilde{\pi}_U(\tilde{\mathbf{s}}^g)$, then control is applied to $\mathbf{s}$, and if $\tilde{\pi}_U(\mathbf{s}) > \tilde{\pi}_U(\tilde{\mathbf{s}}^g)$, then control is applied to $\tilde{\mathbf{s}}^g$.

5. Growing subnetworks using Seed Genes

It is believed that a small number of genes are responsible for a set of functionalities within a cell. Additionally, from the computation prospective, modeling a network with a small number of genes is more feasible. These two reasons motivated Hashimoto et. al [15] to develop an algorithm for growing relatively small subnetworks, out of a large network. Their method is referred to as the *seed-growing algorithm*, because it starts with a small set of genes called *seed*, consisting of one or more genes that are known to be related to the phenotype of interest or are of interest.

The seed-growing algorithm considers the following two criteria, referred to as *principles of autonomy*, while growing subnetworks using gene expression measurements: 1- generating small subnetworks, out of large networks, where genes interact significantly 2- genes within a subnetwork, are not strongly conditioned by genes outside the network. The algorithm iteratively adjoins genes to the network to enhance the autonomy; and finally when the stopping criterion is satisfied, the process stops. The stopping criterion is usually the maximum number of genes that can be added to the network.

The algorithm uses *graph-theoretic* context and models the GRN as a *directed*

graph, where $\mathbf{X}$ is a set of genes, Y is a gene, and the strength of connection from $\mathbf{X}$ to Y is denoted by $\sigma_{\mathbf{X}}(Y)$, which is determined by either CoD (section 2), or *influence*. The *influence* of a variable relative to a Boolean function is found by the partial derivatives of the Boolean function [3]. This variable is among the Boolean variables of the Boolean function. The partial derivative of f w.r.t the variable x_j is 0 if toggling the value of variable x_j does not change the value of f, and 1 otherwise. The influence of x_j on f, is the expectation of the partial derivative w.r.t. the distribution of the variables. The strength from a set $\mathbf{X}$ of genes to the target set $Y = \{Y_1, Y_2, ..., Y_m\}$ of genes, is defined by:

$$\sigma_{\mathbf{X}}(Y) = \psi[\sigma_{\mathbf{X}}(Y_1), \sigma_{\mathbf{X}}(Y_2), ..., \sigma_{\mathbf{X}}(Y_m)] \tag{2.10}$$

where ψ is a function such as the sum, maximum or minimum.

The goal is to grow a subnetwork $\mathbf{S}$ that has a strong collective strength of connections among the genes within $\mathbf{S}$, and has the minimum collective strength of the connections from outside the subnetwork. To achieve this goal, the following definitions are used.

Let $\mathbf{U}$ be the set of all genes under study, and let the target set Y be such that $Y \notin \mathbf{S}$. The equation 2.11 is defined to measure the sensitivity of Y from $\mathbf{S}$, i.e. the collective strength of connection from the network $\mathbf{S}$ to the target Y:

$$\sigma_{\mathbf{from},\mathbf{S}}(Y) = \sigma_{\mathbf{S}}(Y) \tag{2.11}$$

and equation 2.12, measures the impact of Y to $\mathbf{S}$, i.e. the strength of connection from the target to the network:

$$\sigma_{\mathbf{to},\mathbf{S}}(Y) = \sigma_{\{Y\}\cup\mathbf{S}}(\mathbf{S}) \tag{2.12}$$

The strength of connection from genes external to $\mathbf{S}$ to Y is defined as:

$$\sigma_{\mathbf{out},\mathbf{S}}(Y) = \max_{X \subset U-(S \cup \{Y\}), \|X\| \leq m} \sigma_X(Y) \tag{2.13}$$

where m is the maximum number of genes allowed in the strength function.

The gene $\hat{Y}$, which most enhances the network autonomy, will be adjoined the network. $\hat{Y}$ satisfies:

$$\hat{Y} = arg \max_{Y \notin \mathbf{S}} \Xi[\sigma_{\mathbf{from},\mathbf{S}}(Y), \sigma_{\mathbf{to},\mathbf{S}}(Y), \sigma_{\mathbf{out},\mathbf{S}}(Y)] \tag{2.14}$$

where Ξ is a function to return the collective value of three parameters: $\sigma_{\mathbf{from},\mathbf{S}}(Y)$ and $\sigma_{\mathbf{to},\mathbf{S}}(Y)$ and $\sigma_{\mathbf{out},\mathbf{S}}(Y)$. $\hat{Y}$ maximizes Ξ.

After a certain number of genes are added to the subnetwork(s), the algorithm stops and returns a graph, representing the GRN [15].

6. Generating Boolean networks from prescribed attractor sets

Modeling GRNs with an intention to intervene in their long-run behavior is considered the key problem for genomics [14]. In the opposite direction, synthetically generating networks, which possess the key characteristics of the model, is another important task. The synthetic networks can be used to examine the inference and intervention of the GRNs. The inverse problem of generating networks, with a given set of characteristics, is addressed by Pal et. al. [14]. This problem is ill-posed, meaning that it is possible to generate many, or none, networks with the desired properties.

As provided earlier in this chapter, section 1, the transient states of the Markov chain transition to one of the attractor states with j transitions and j defines the level of the state. Attractors define the long-run behavior of the network. Considering these facts, two algorithms are developed by [14] to generate Boolean networks. The first

algorithm works with truth table by incorporating the attractors structure, predictor sets and the constraints on the levels in filling out the TT. The second algorithm randomly generates a state transition diagram, which dynamically represents the network, and then checks the validity of the constraints on the generated network. The first algorithm is widely used in PBN modeling and is chosen as the method for generating synthetic Boolean networks in this dissertation.

The method assumes for a given set V consisting of n genes, a family of n subsets $W_1, W_2, ..., W_n$ of V with cardinalities not less than m and not larger than M, $0 < m \leq M$, a set A containing k states, and two positive integers $l \leq L$, accordingly construct a BN with node set V, having predictor set $W = (W_1, W_2, ..., W_n)$, possessing attractor cycles $A_1, A_2, ..., A_r$, where $A = \cup_{j=1}^{r} A_j$, and containing between l and L level sets. Algorithm 1 starts by generating a set of k attractor states. Then, it randomly picks a predictor set W, where $m \leq \|W_i\| \leq M$ for all i. The next step is to checks if the selected attractor set is compatible with W. The compatibility refers to the matching of truth table entries for the set W with the state transition diagram. It then, fills out the truth table entries for generated attractors. Afterwards, if there exists any cyclic attractors, it goes back to the previous step. This is necessary because this method generates BNs with singleton attractors. An extension to the current algorithm is provided to generate networks with cyclic attractors. And finally, the levels of the networks are checked to ensure that they meet the minimum and maximum allowed levels.

It is claimed that most of microarry data represent the steady-state behavior. The majority of the probability mass in the steady-state comes from the attractors [13] and is it expected that modeling BNs, with a given attractor structure, represents the models that possess the key components of the real networks. Assuming that the sampling is coming from steady-state, checking the validity of the networks generated

by their proposed model [14] is done by checking the SSD mass in the observed sample states. They showed that in the case of a well-studied *WNT5A* network, the SSD probabilities of the model generated by the algorithm matches closely with the frequencies observed form sampling the data.

CHAPTER III

REDUCTION MAPPINGS*

A. Introduction

A key objective for modeling gene regulatory networks is to derive intervention strategies for beneficially altering cell dynamics [5] toward more desired stages. To address the issue of intervening in the long-run behavior, stochastic control has been used to find stationary control policies that affect a network's steady-state distribution [4]. However, optimal control policy methods are computationally complex [6, 7] and often it is not feasible to design optimal control policies for large networks. A possible approach to complexity reduction in the finite-horizon model is to use a discrete linear model [23]; however, this dissertation focuses on the more general nonlinear and infinite-horizon case where network dynamics are described by a Markov chain. Even with the Boolean model, where gene states are binary, the state transition matrix (STM) grows exponentially as the number of genes grow.

As a solution to handle the complex networks, approximation via re-inforcement learning [24] and greedy-control methods [8, 9] are proposed, but they are restricted in the size of networks they can handle. For instance, rather than doing a full optimization relative to some objective function and face the "curse of dimensionality" associated with dynamic programming, greedy methods utilize statistical characteristics of the network, including MFPT-CP [8] and SSD-CP [9]. But these still require

* Part of this chapter is reprinted, with permission, from "A cod-based reduction algorithm for designing stationary control policies on boolean networks", N. Ghaffari, I. Ivanov, X. Qian, and E. Dougherty, *Bioinformatics*, vol. 26, no. 12, pp. 1556-1563, 2010.

manipulating the state transition matrix, which effectively limits their use on large networks working on the current computational power.

As a solution for working with large GRNs, this dissertation takes the approach of reducing the size of the network, designing a control policy on the reduced network, and then inducing that control policy on the full network. It is motivated by a previously proposed network reduction algorithm that removes genes in such a way that the deleted gene induces a specific collapsing of pairs of states from the state space of the original network [25]. While other reduction algorithms have been developed to obtain reduced models for Boolean or probabilistic Boolean networks to maintain either the structural consistency [26] or the dynamical behavior of the original network [27], the specific intent in the current method is to find a reduction strategy that can provide beneficial stationary control policies for the original network. Boolean networks with perturbation, $BN_p s$, are used to model gene regulatory networks. The key point for choosing $BN_p s$ is that their dynamics can be modeled using Markov chains; thereby facilitating the development of control policies that can shift the network steady-state distribution towards desirable states.

As typically formulated, the intervention is characterized by a *target* gene whose expression is to be altered by the control policy and one *control* gene whose expressions is altered by intervention. The control policy acts by observing the state of the network at each time point and, based on the state, decides whether to alter the value of the control gene.

B. Proposed method: CoD-Reduce

This section described a new method that focuses on large networks where it is not possible to derive the optimal control policy. This method deletes one or more genes

from the network so that a policy on the reduced network can be designed, which would induce a sub-optimal policy on the original network. There are four basic steps in this procedure:

1. apply an algorithm to the network to select a gene for deletion
2. apply an algorithm to construct the gene logic for the reduced network
3. apply a control algorithm to the reduced network to derive a control policy on the reduced network
4. induce a control policy on the original network based on the control policy derived for the reduced network

The method proposed herein employs the coefficient of determination (CoD) [10] to choose genes for deletion, adapts the collapsing heuristic of [25] to construct the wiring of the reduced network, designs a control policy on the reduced network using either the MFPT-CP [8] or SSD-CP [9], and finishes with a procedure to induce a control policy on the original network. Performance of the CoD-based reduction procedure is evaluated by its effects on the steady-state distribution to shift the probability mass towards the desirable states. This shift is computed as the absolute value of the difference between the steady-state distributions of the network before and after applying the control policy. Additionally, the effects of the reduction on the control is studied using a new measure called *relative effect*, which compares the control policy designed on the original network with the control policy induced from the reduced network.

1. Selecting the Best Gene for Deletion

The first step of the algorithm is to select the gene to be deleted, based on two criteria:

1. If there are genes *isolated* from the rest of the genes, then the algorithm randomly selects one of them as the candidate for deletion. A gene is called *isolated* if it does not predict any other genes and no other gene predicts it.

2. Otherwise, the combination of 3 genes that has the smallest steady-state CoD in determining the target gene is found and the gene chosen for deletion is the one with the weakest influence in terms of CoD value on the target gene from that triple of genes.

The CoD is based on triples of genes because, as Kauffman points out, the average connectivity of the model cannot be too high if its dynamics is not chaotic [28], and 3-predictor connectivity is commonly assumed in Boolean network and PBN modeling [26]. The procedure ensures that the candidate gene for deletion from the network has small influence on the target gene if the model has reached its steady-state distribution. The deletion procedure is described in detail in algorithm 1.

2. Reduction Mappings using Selection Policy

After selecting the gene for deletion, called d, a reduction mapping is used to define the transition rules for states in the reduced network [27]. The design of the reduction mapping is based on the notion of a *selection policy* [29]. A *selection policy* determines the transitions for the states of the reduced network and is formally defined as:

Definition 1. *A* selection policy ν^d *corresponding to the deleted gene* d *is a* 2^n *dimensional vector,* $\nu^d \in \{0,1\}^{2^n}$*, indexed by the states of* S *and having components equal to* 1 *at exactly one of the positions corresponding to each pair* $(\mathbf{s}, \tilde{\mathbf{s}}^d)$*,* $\mathbf{s} \in S$*.*

Algorithm 1 CoD-Reduce: Selecting Best Gene For Deletion

Create connectivity table of the BN on n genes
Exclude self-predictions from the connectivity table
Compute set $C = \{c_1, c_2, ..., c_n\}$, where each c_i is the total number of genes that predict g_i or being predicted by g_i
Find all $c_i = 0$ and put their corresponding g_i in the constant gene set: *CONSTANT*
if $CONSTANT \neq \emptyset$ **then**
    GENE For DELETION $\leftarrow$ randomly selected gene from the *CONSTANT*
else
    Compute set *COMBINATIONS*: includes all 3-gene combinations, excluding the target gene
    for all the sets in *COMBINATIONS* **do**
        $\Theta_j \leftarrow$ CoD of the 3-gene set j w.r.t. target gene
    end for
    Find a 3-gene combination with minimum Θ_j: *MINCOD*
    GENE For DELETION $\leftarrow g_i \in$ *MINCOD* with minimum individual CoD w.r.t target gene
end if
return (*GENE For DELETION*)

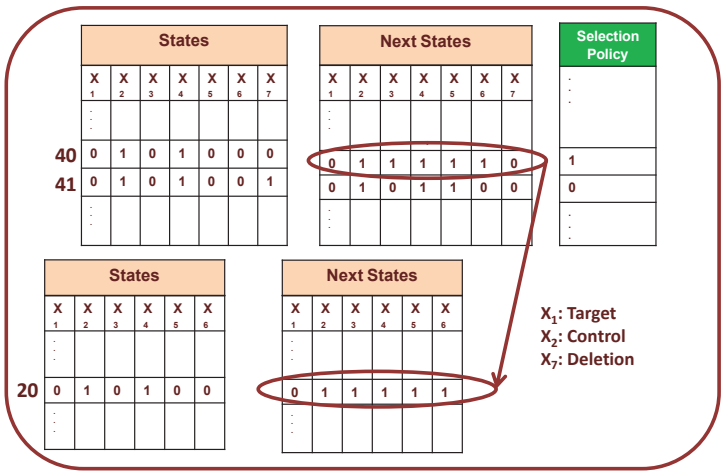

Fig. 2. Selection policy. For the two states that only differ in the gene for deletion, the one that has 1 in selction policy vector, defines the structure of the reduced network.

Finding an optimal selection policy would require testing each of the $2^{2^{n-1}}$ possible reduced networks, which is computationally infeasible. In the present dissertation, a heuristically chosen selection policy is combined with an inducement procedure to design a control policy on the original network. The selection policy used here is designed by considering the steady-state distribution of the network. The intuition behind this approach relies on two facts: first, attractors are an essential part of the network and therefore should be preserved during the reduction; second, states with larger steady-state probability are more likely to be visited during the long run transitions of the network. Based on these considerations, the selection policy proceeds as follows: for states $\mathbf{s}$ and $\tilde{\mathbf{s}}^d$ that only differ in the deleted gene d, the functions of the state possessing larger steady-state probability will be kept for the reduced state, excluding its gene for deletion; however, if either $\mathbf{s}$ or $\tilde{\mathbf{s}}^d$ is an attractor and the

other is not, then the attractor state is chosen to determine the function structure. Algorithm 2 represents the steps of reduction mapping. This selection policy has 1 for the states whose functions are kept as the result of reduction and 0 for the rest. Figure 2 represents the concept of selection policy graphically.

3. Inducement

Suppose the original network has n genes, the reduced network has $m < n$ genes based on $n - m$ deletion-reduction applications, and, without loss of generality, suppose the last $n - m$ genes have been deleted. Then, for any state $(x_1, x_2, ..., x_m)$ in the reduced network, there are 2^{n-m} states in the original network of the form $(x_1, ..., x_m, z_1, ..., z_{n-m})$. If μ_{red} is the control policy designed on the reduced network, then the induced policy on the original network is defined by

$$\mu_{ori}(x_1, ..., x_m, z_1, ..., z_{n-m}) = \mu_{red}(x_1, x_2, ..., x_m) \tag{3.1}$$

for any $z_1, ..., z_{n-m} \in \{0, 1\}$.

C. Discussion

Several simulations are carried out to study the performance of the *CoD-Reduce* algorithm. Two approaches are used to measure the goodness of the reduction mappings: first, the changes in the control policy of the network, and second, the shift of the steady-state distribution towards the desired states. This section describes each approach and represents the results

Algorithm 2 CoD-Reduce: Reduction Mapping

Put all the attractor States in a set called: *ATTRACTORS*
Find the SSD of the network: π
for all the States **s** in the State space **do**
    find its flipped State w.r.t. gene For deletion: $\tilde{\mathbf{s}}^d$
    if ((**s** $\in$ *ATTRACTORS*) & ($\tilde{\mathbf{s}}^d \notin$ *ATTRACTORS*)) **then**
        *Selection Policy (**s**) = 1*
        Selection Policy ($\tilde{\mathbf{s}}^d$) = 0
    else if ((**s** $\notin$ *ATTRACTORS*) & ($\tilde{\mathbf{s}}^d \in$ *ATTRACTORS*)) **then**
        *Selection Policy (**s**) = 0*
        Selection Policy ($\tilde{\mathbf{s}}^d$) = 1
    else
        if ($\pi(s) > \pi(\tilde{\mathbf{s}}^d)$) **then**
            *Selection Policy (**s**) = 1*
            Selection Policy ($\tilde{\mathbf{s}}^d$) = 0
        else
            *Selection Policy (**s**) = 0*
            Selection Policy ($\tilde{\mathbf{s}}^d$) = 1
        end if
    end if
end for
for all the States **s** in the State space that have *(Selection Policy (**s**) = 1)* **do**
    Keep the transitions of the **s** excluding the ***d*** coordinate as the
    transitions of $\check{\mathbf{s}}$: reduced State
end for

1. Relative effect

As the first measure, one can study the effects of the reduction mappings framework on the control policy of the network. When interpreting the deletion of gene d as creation of a latent, or non-observable, variable, it is desirable that there is a stationary control policy $\check{\mu}_g$ for the reduced network $B\check{N}_p$ that is as close as possible to μ_g-the one designed for the original network. In this way, one can achieve similar control actions for every state $\mathbf{s}$ and its corresponding reduced state $\check{\mathbf{s}}$. For example, if one considers the action of a stationary control policy $\check{\mu}_g$ on the state $\check{\mathbf{s}}$ in the reduced network the similarity of control could be fully achieved only if $\mu_g(\mathbf{s}) = \mu_g(\tilde{\mathbf{s}}^d)$. The following two definitions set up the framework about how to measure the effects of a selection policy-induced reduction mapping on a stationary control policy [29].

Definition 2. *Given a stationary control policy μ_g and a gene d to be deleted from the BN_p, the policy μ_g is called* d-inconsistent *at the state $\mathbf{s} \in S$ if and only if $\mu_g(\mathbf{s}) \neq \mu_g(\tilde{\mathbf{s}}^d)$. The state $\mathbf{s}$ is called* μ_g^d-inconsistent. *The ratio $r_{\mu_g}^d$ of the number of μ_g^d-inconsistent states $\mathbf{s} \in S$ to the total number of states in S is called the* d-relative inconsistency *of the control policy μ_g.*

As a consequence of this definition, one can measure the effects of a selection policy-induced reduction mapping by comparing the control actions for the subset $C^d \subseteq S$ of states that are not d-inconsistent to the control actions for their corresponding reduced states in the reduced network $B\check{N}_p$.

Definition 3. *Given a stationary control policy μ_g and a gene d to be deleted from the BN_p, the policy μ_g is called* ν^d-affected *at the state $\mathbf{s} \in C^d$ if and only if the control action for the reduced state $\check{\mathbf{s}} \in \check{S}$ is different from $\mu_g(\mathbf{s})$. The ratio $\varepsilon_{\mu_g}^{\nu^d}$ of the number of states $\mathbf{s} \in C^d$ where the control policy is ν^d-affected to the total number of*

states in C^d *is called the* relative effect *of the selection policy* ν^d *on the control policy* μ_g.

Because there is only a finite number of selection policies ν^d, there exists a selection policy $\nu^d_\circ$ that is *optimal* with regard to minimizing the relative effect $\varepsilon^{\nu^d}_{\mu_g}$ among the all possible selection policies ν^d on the stationary control policy μ_g.

In general, the shift in the steady-state distribution and the relative effect of the selection policy on the control policy follow inverse trends. This is to be expected because a big relative effect on the control policy implies significant difference in the control actions for the states on the larger network and their respective reduced states in the smaller network which could ultimately lead to a significant difference in the shifts induced by those control policies in the steady-state distributions of the models.

2. Effects on the steady-state distribution

This section represents the effects of reduction mappings on the SSD of the networks. Letting $\pi_D = (\pi_{D1}, \pi_{D2}, ..., \pi_{Dm})$ and $\omega_D = (\omega_{D1}, \omega_{D2}, ..., \omega_{Dm})$ denote the probability vectors composed of steady-state masses of the desirable states before and after control, respectively, the shift is defined by

$$\Delta = \sum_{k=1}^{m} \omega_{Dk} - \pi_{Dk} \tag{3.2}$$

Δ provides a measure of the effectiveness of the overall algorithm – deletion, reduction, and inducement – the goal being to decrease the probability of being in undesirable states and increase the probability of being in desirable states in the long-run. Also the induced policies arising from reductions to $n-1, n-2, ..., 4$ genes are applied and the mass shifts are computed.

3. Randomly Generated Networks

To study the performance of the *CoD-Reduce* algorithm,a simulation study is performed on sets of 100 randomly generated Boolean networks with perturbation, with $7, 8, 9$, and 10 genes. These have been generated using the algorithm developed in [14], subject to the constraint that for each network half of its attractors are among the desirable states. For each network of size $n \in \{7, 8, 9, 10\}$ genes, the original network is reduced to $n-1, n-2, ..., 4$ genes and for each reduction designed either the MFPT-CP or SSD-CP on the reduced network and induced a policy on the original network of size n. The networks are limited to 10 genes because the control policy on each originally generated network needs to be computed in order to make the comparisons. However, in general *CoD-Reduce* is not restricted to any number of genes and can handle large networks, as long as the steady-state distribution is obtainable. In such large networks *CoD-Reduce* is used to make reductions by deleting genes until the point that it is feasible to compute the control policy of the reduced network. Figure 3 shows, the average shift of the steady-state distribution under the MFPT-CP on the original network and the average shift for the induced policies arising from reductions to from n to $n-1, n-2, ..., 4$ genes. Figure 4 gives analogous results using the SSD-CP. The salient point regarding the 9- and 10-gene networks is that, after an initial drop off for a few-gene reduction, the shift tends to stabilize for further reduction and, in all cases, the induced policy achieves significant beneficial results.

Note that the beneficial steady-state-shift when designing the control policy on the original network is, on average, slightly better for the SSD-CP in comparison to the MFPT-CP, and that this agrees with the findings in [9]. On the other hand, the induced policy arising from the MFPT-CP designed on the reduced network slightly

outperforms the induced policy arising from the SSD-CP designed on the reduced network.

Another meaningful comparison between the original and reduced networks is the effect of the reduction measured by relative effect. The expected behavior of successful reduction is the inverse pattern for the SSD shift and the relative effect.

Figure 5 illustrates that *CoD-Reduce* does not have a significant effect on the amount of the shift of the steady-state distribution of the network towards the desirable states if a single gene is removed from the model. Similarly, there is a small change (on average) in the relative effect of the selection policy on the MFPT control policy when moving from a network to its reduced version. Thus on average, *CoD-Reduce* does not have much of an impact on the controllability of the network when a single gene is removed from it. However, the effects of reduction tend to accumulate with the removal of more genes, and ultimately the reduction mappings produce poor results when a very few genes remain in the network.

In random-network simulations one also has the issue of how to choose the target gene, since the randomly generated networks are of a purely computational nature. In practice, the target gene is chosen in such a way that its behavior is closely related to the phenotype of interest and the control gene can either be selected via biological knowledge or according to some criterion related to its ability to control the target gene. Since methods described in this dissertation are interested in networks in which the target gene is controllable, one needs to chose a target gene for which there exists a non-target gene that can exercise control over it. A simple way to do that is to consider all gene-to-gene CoDs and pick the control and target genes to be the two genes possessing the largest gene-to-gene CoD, the former being the control gene and the latter being the target gene. While it is true that this choice provides greater controllability than would normally be expected in a real biological problem, it affords

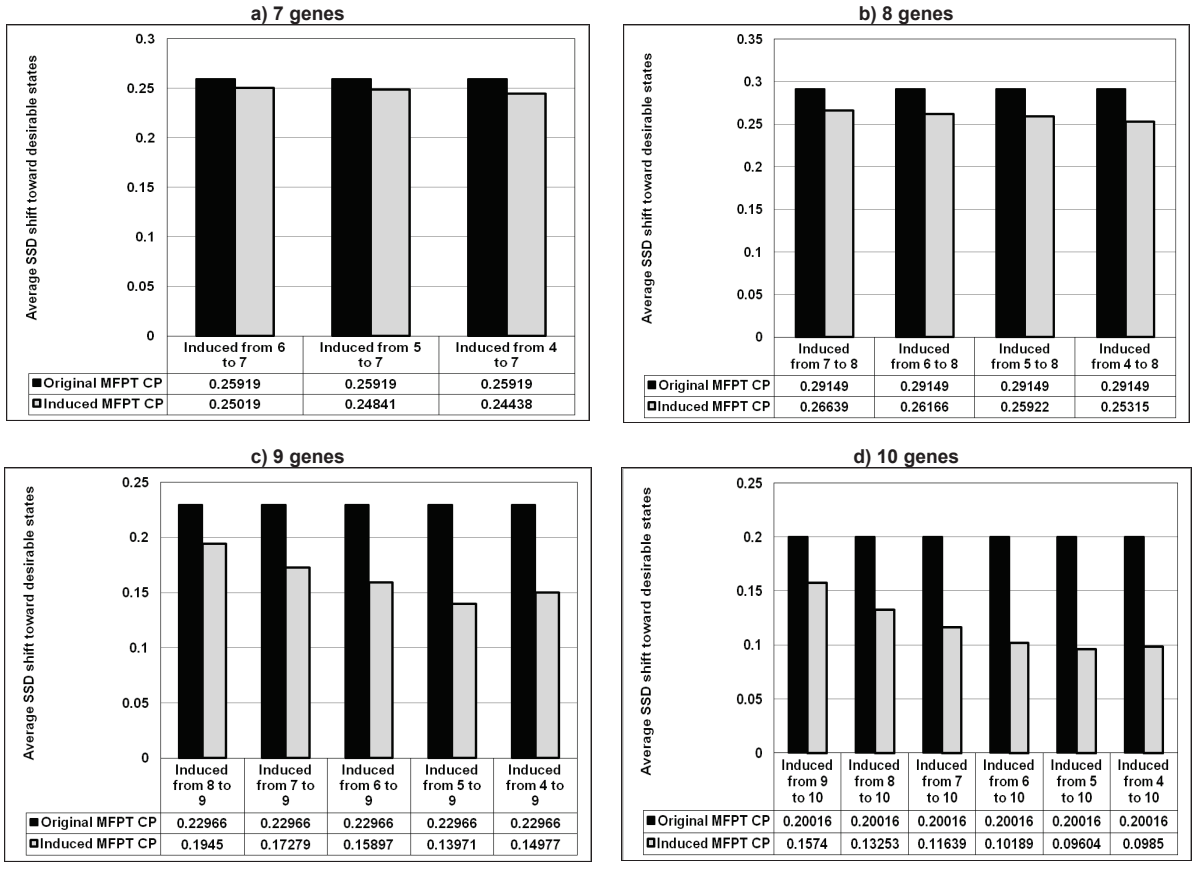

Fig. 3. The average shifts of the steady-state-distribution produced by applying the original MFPT and the stationary induced control policies, using different number of genes. The original MFPT control policies were obtained before any reductions. The induced control policies were designed on the reduced networks after applying reduction several times and then inducing the control policy of the reduced networks back to the original network. Each one of the four sets of 100 BN_ps was generated using randomly generated attractor sets; attractors are evenly distributed between desirable and undesirable states.

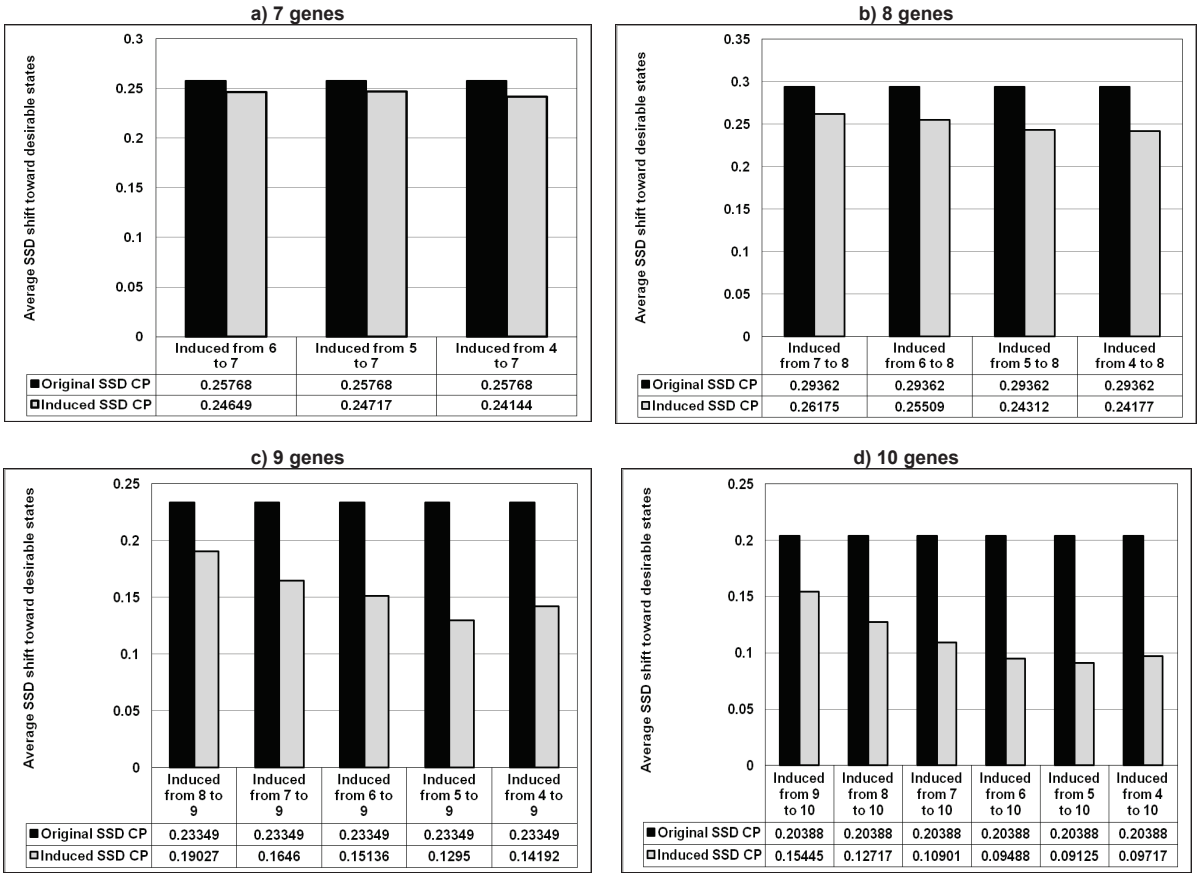

Fig. 4. The average shifts of the steady-state-distribution produced by applying the original SSD and the stationary induced control policies, using different number of genes. The original SSD control policies were obtained before any reductions. The induced control policies were designed on the reduced networks after applying reduction several times and then inducing the control policy of the reduced networks back to the original network. Each one of the four sets of 100 $BN_p s$ was generated using randomly generated attractor sets; attractors are evenly distributed between desirable and undesirable states.

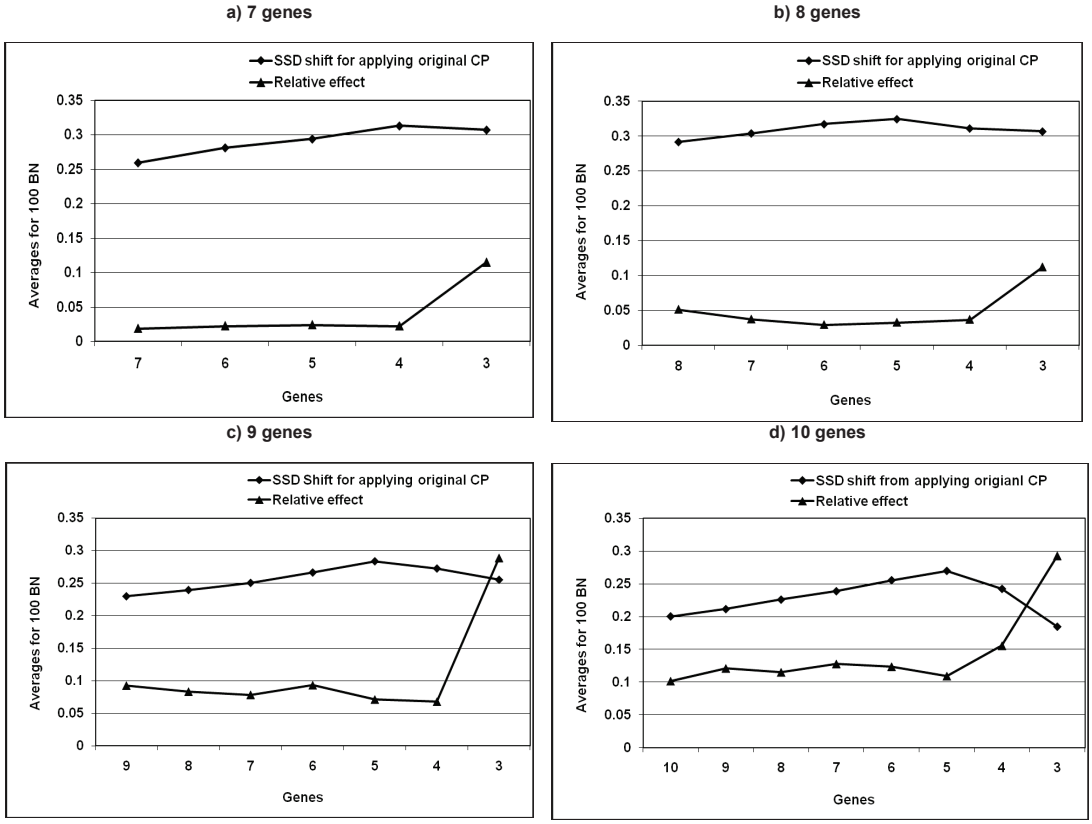

Fig. 5. The average SSD shift toward Desirable states and the relative effects on the control policies of successive reductions of 4 sets of 100 BN_p. Each set has randomly generated attractors which constrained to be evenly distributed between the Desirable and Undesirable states. At each step the MFPT-CP is designed on the network and applied to itself. As the figure shows the effect on the SSD is similar in the original and reduced networks by applying their own control policies. Also, SSD shift and relative effect curves follow inverse patterns.

us the opportunity to get a good measure of the loss of controllability that results from *CoD-Reduce*, that is, from deletion, reduction, and inducement. This implies that the Target-Control pair needs a direct connection in the GRN. This condition relaxed on the later experiments in Chapter IV. Appendix A provides a flowchart that displays all the steps for the algorithm of selecting a *Target-Control* pair.

D. Alternative Algorithm: CoD-Reduce II

In order to simplify the method for reduction mappings, this section represents a modified version of *CoD-Reduce* and calls it *CoD-Reduce II*. The main idea is that after fixing the target and control genes, all the other genes in the network will be considered as candidates for deletion and their corresponding CoD w.r.t target gene is calculated. Using these CoDs, genes will be sorted in the ascending order and will be deleted from the network. The procedure for finding the best target and control genes is the same as *CoD-Reduce*.

The main difference between *CoD-Reduce* and *CoD-Reduce II* is the deletion process. The *CoD-Reduce* algorithm finds all 3-gene sets predicting the target gene, excluding control genes, and finds their CoDs w.r.t the target gene. Then, it finds the 3- gene predictor set with minimum CoD predicting the target, and calculates the individual CoDs of its genes w.r.t the target gene. The gene with minimum individual CoD is set as the candidate gene for deletion. This procedure is repeated in each reduction step. Therefore, for the reduction of a n-gene BN to a 2-gene BN, this process needs to be done $n-2$ times. On the other hand, *CoD-Reduce II* initially finds the individual CoDs of all genes, excluding control gene, for predicting the target. It then sorts the genes based on the their CoDs and in the reduction steps deletes these genes accordingly. This way finding the best gene for deletion needs to

be done only once using the original network. Algorithm 3 represents the steps of the *CoD-Reduce II* process.

Algorithm 3 CoD-Reduce II

1: Create connectivity table of the BN on n genes
2: Exclude self-predictions from the connectivity table
3: Set the control gene C and target gene T
4: **for** all the genes, except C **do**
5: $\Theta_i \leftarrow$: CoD of the gene i For predicting target gene T
6: **end for**
7: Θ_j: Sort Θ_i by ascending order
8: Delete genes from Θ_j, until only T and C are left, or until a pre-specified number of genes remain

1. Simulation Results

A simulation study is performed on the effects of the alternative selection-induced reduction, *CoD-Reduce II*, on the SSD shift and the MFPT-CP. Using the algorithm developed in [14], several sets of 100 $BN_p s$ for $n = 7$ are randomly generated. Each set shares a common set of attractor states with the restriction that they form only singleton attractors. In addition, the number of predictor genes for each gene in the network is restricted to be ≤ 3 to keep the networks away from being chaotic.

Figure 6 represents the averages (over one of the sets of 100 networks) of both the SSD shift towards the desirable states and the relative effect of the respective selection policies on the MFPT-CP, using *CoD-Reduce II*. Figures 7 and 8 show the effects of applying the *CoD-Reduce II* on the SSD of the original network, after applying control policies induced form the reduced networks compared with the control policies

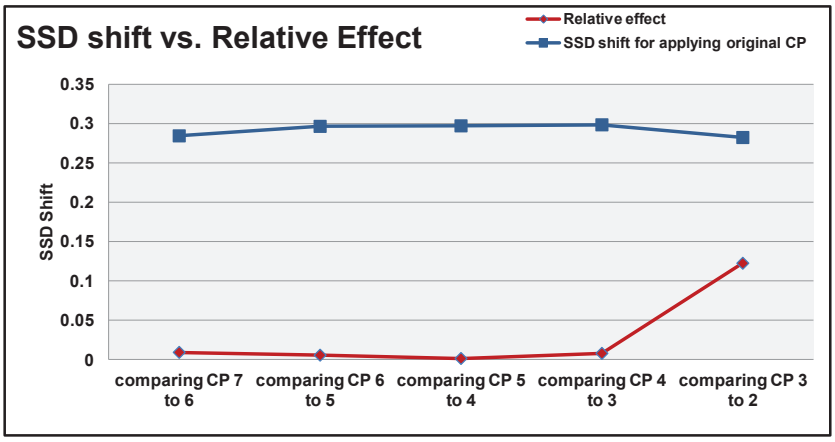

Fig. 6. The average SSD shift toward Desirable states and the relative effects on the control policies of successive reductions averaged for 100 BN_p, using *CoD-Reduce II*. At each step the control policy is designed on the network and applied to itself. After deleting each gene, the control policy designed on the reduced network and induced back to its original network. SSD shift and relative effect curves follow inverse patterns.

designed on the original networks.

These figures illustrate the typical performance of *CoD-Reduce* and its correspondence *CoD-Reduce II*: the shift of the steady-state distribution varies very little from a network to its reduced version. Moreover, the combination of selection policies based on the SSD and the CoD ranking of the genes provides us with reduction mappings that have very small relative effect on the MFPT-CP. The algorithm's performance deteriorates significantly only when the size of the network becomes very small (< 4) genes. This is to be expected because for small size networks removal of even one gene can significantly damage the dynamical behavior of the model.

E. Case study: A 4-gene BN and walk through of the concepts

This section presents a numerical example that covers the steps of the proposed algorithm, *CoD-Reduce*. The algorithm proposed by Pal et. al. [14] is used to

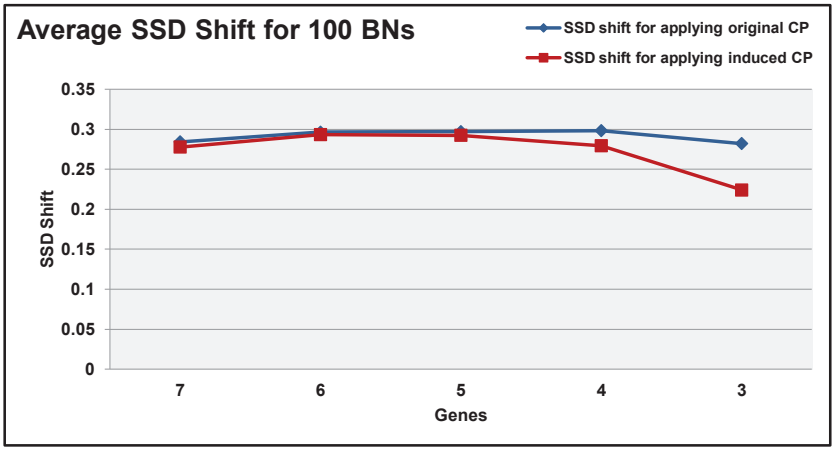

Fig. 7. The average SSD shift toward Desirable states by applying the original and induced control policies after each reduction, averaged for 100 BN_p, using *CoD-Reduce II*. At each step the control policy is designed on the network and applied to itself. After deleting each gene, the control policy designed on the reduced network and induced back to its original network. SSD shift toward Desirable states, generated by original and induced control policies, have very similar effects on the networks.

generate a 4-gene BN_p. This synthetic network has $2^4 = 16$ states. The corresponding Truth Table (TT) that defines the rules for one step transitions is shown as Figure 9. The underlying model of a BN_p is a finite Markov chain (MC) and its dynamics are completely described by its $2^4 \times 2^4$ State Transition Matrix (STM) that is usually represented by P. Also the perturbation probability p, the probability that each gene can be randomly flipped, makes the chain ergodic and therefore it possesses a Steady-State Distribution (SSD) that is represented by π. The SSD of the network is given by the equation 2.1 and can be found as an eigenvector of the P.

In a BN_p, the transition probability from state y to state x is given by the following equation [9]:

$$P_y(x) = 1_{[f(y)=x]}(1-p)^n + 1_{[x\neq y]}p^{\eta(x,y)}(1-p)^{n-\eta(x,y)} \tag{3.3}$$

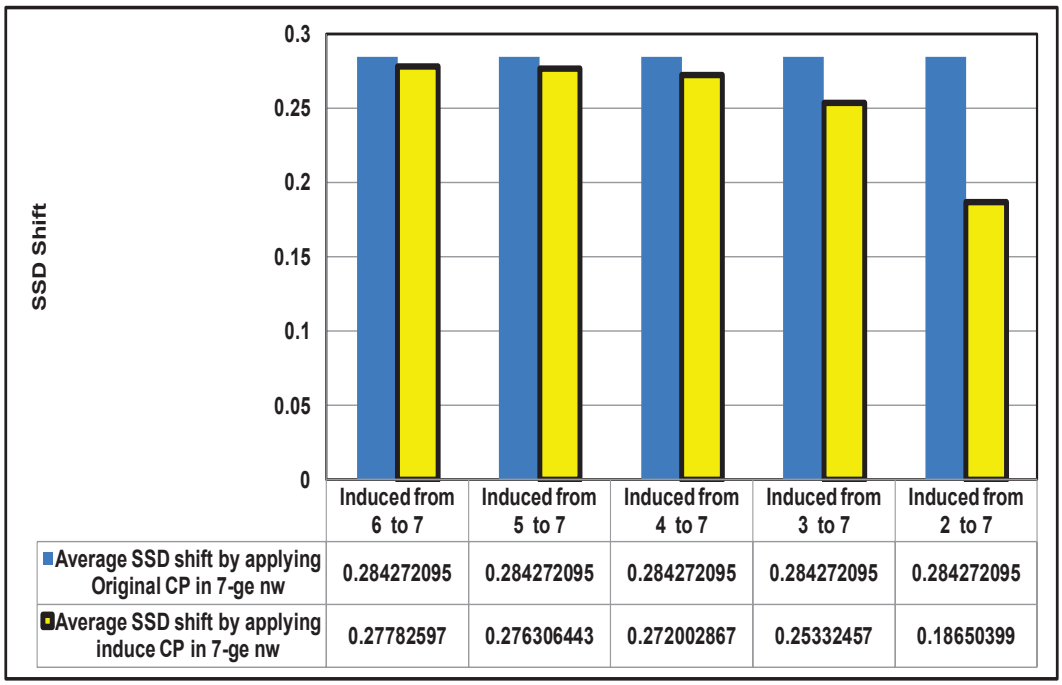

Fig. 8. The average SSD shift toward Desirable states by applying the original and induced control policies after all the reduction steps using *CoD-Reduce II*, averaged for 100 BN_p. After deleting genes, the control policy designed on the reduced network and induced back to the original network.

Decimal representation (states)	States				Next States				
	G1	G2	G3	G4	F1	F2	F3	F4	
0	0	0	0	0	1	0	1	1	
1	0	0	0	1	1	0	1	1	
2	0	0	1	0	0	0	1	0	Singleton Attractor
3	0	0	1	1	0	0	1	0	
4	0	1	0	0	1	1	0	0	
5	0	1	0	1	1	1	1	0	
6	0	1	1	0	0	1	0	0	
7	0	1	1	1	0	1	1	0	
8	1	0	0	0	1	0	1	1	
9	1	0	0	1	1	0	1	1	
10	1	0	1	0	0	0	1	0	
11	1	0	1	1	0	0	1	0	
12	1	1	0	0	1	1	0	0	Singleton Attractor
13	1	1	0	1	1	1	1	0	
14	1	1	1	0	0	1	0	0	
15	1	1	1	1	0	1	1	0	

Fig. 9. The truth table of the 4-gene network. The states 2 and 12 are the singleton attractors.

where $\eta(x, y)$ is the Hamming distance between x and y, and $1_{[f(y)=x}$ is the indicator function that equals to 1 if $f(y) = x$ based on truth table, and 0 otherwise. The transition probability for a singleton attractor to itself is: $(1-p)^n$.

In this example, the first gene from the left hand side g_1 is set to be the Target (T) gene and $g_1 = 0$ defines desirable states and $g_1 = 1$ defines undesirable states. The second gene from left hand side g_2 is defined as Control (C) gene. After these initial settings, the steps of *CoD-Reduce* are followed to reduce this network. A one step reduction is done which will select the best gene for deletion, then deletes it. After this deletion step, in the reduced 3-gene network a MFPT-CP is designed and induced back to the original 4-gene network and the result of applying this induced control policy is represented. The purpose of the current example is to demonstrate the algorithm presented in the current chapter and therefore, applying one of the CPs is sufficient for the reader to follow the details.

1. Selecting best gene for deletion

Initially, *CoD-Reduce* forms all possible 3-gene combinations that exclude the target gene T. Since there are only 4 genes in this example, then the only possible triple that excludes T is the set: g_2, g_3, g_4. Since g_2 is the control gene, therefore the candidate gene for deletion should be selected between: g_3 and g_4. To decide which gene to be deleted, the individual CoDs of g_3 and g_4 are needed. They are calculated as the following:

- CoD of g_3 w.r.t. T = 1
- CoD of g_4 w.r.t. T = 0

Since g4 has the minimum individual CoD w.r.t T, so it is selected as the gene for deletion. Hence, Gene for Deletion (D): g_4

2. Designing the Selection Policy

After selecting the gene for deletion, next step is to define the Selection Policy (SP) in order to reduce the network. For each candidate gene for deletion, there are $2^{2^{n-1}}$ SPs where n is the number of genes. As n grows, in the cases of large n, it is not computationally feasible to try all the possible SPs. Earlier in this chapter, a heuristic approach for designing a SP is introduced, which is highly dependent on the attractor structure and SSD probabilities of the network. This approach is represented as algorithm 2. In this example, with 4 genes there are 256 possible SPs which is a manageable number of possibilities. The effect of each one of these SPs on the dynamic behavior of the network can and has been examined. All the possible SPs are used to reduce the original network and then a CP is designed and applied to the reduced network; the SPs that are used for constructing the reduced networks

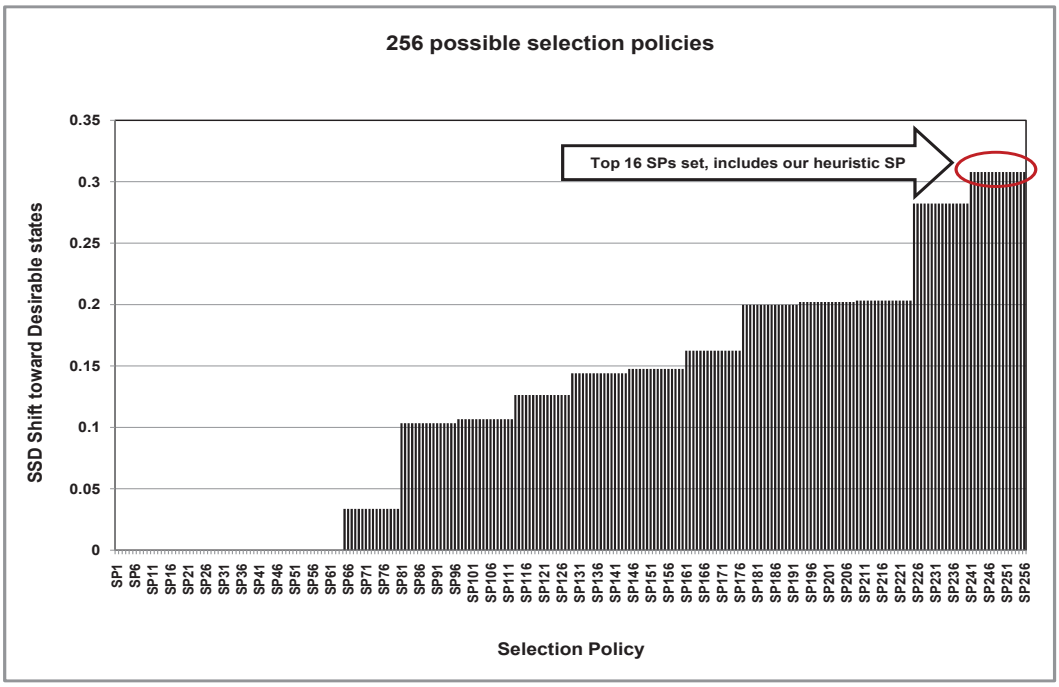

Fig. 10. All 256 possible SPs for 4-gene network and their shift of steady-state distribution toward more desirable states. Our heuristic SP is among the 16 optimal SPs that have maximum SSD shift toward desirable states.

that have maximum shift in their SSD after applying control are called optimal SPs. Figure 10 shows that our heuristic SP is among the 16 optimal SPs.

a. Proposed heuristic selection policy

A selection policy is a vector with the *number of components* $= 2^n$, where 2^n is the number of the states of the original network and $SP \in \{0,1\}^{2^n}$. It defines the reduction mapping to construct the TT of the reduced network and for each two states that differ only in the values of the gene for deletion, only one of them can have the SP action equal to 1.

As it is described in the algorithm 2, first the SP actions for the attractors and their flipped states w.r.t. the gene for deletion are defined; the attractor states get

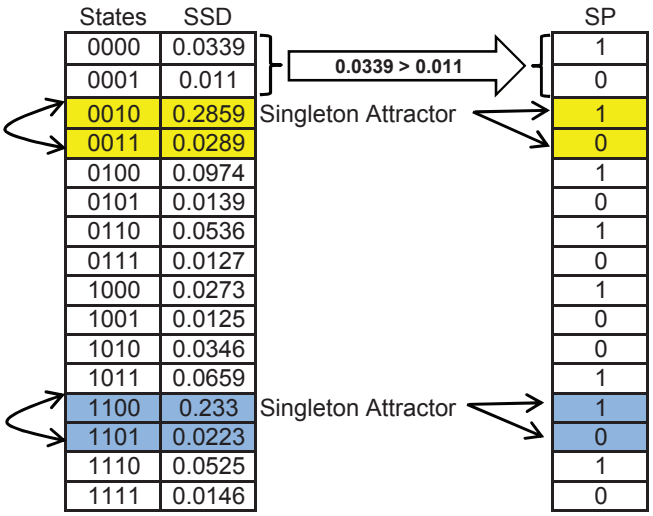

Fig. 11. The process for designing selection policy using the proposed heuristic algorithm

the SP equal to 1. Then steady-state probabilities are used to define the SP for the other states; the heavier weight of a state implies setting of the respective entry in the SP to 1. Each state is compared with the states that differ from it only in the value of the gene for deletion: g_4 for this example. Figure 11 graphically demonstrates the process of designing the SP for our 4-gene network.

3. Reducing the network, using selection policy

The next step is to define the TT of the reduced network after deleting g_4. For this purpose, the SP designed above is used. For each 2 states that differ only in the value of the gene for deletion, the state that has SP = 1 will be the one that defines the transitions for the reduced state. As an example, the state 0100 transitions to 1100 according to the TT:

$0100 \longrightarrow 1100$

The state that differs with 0100 only in g_4 is 0101 and the following is the rule for its transition:

States			Next States		
0	0	0	1	0	1
0	0	1	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
1	0	0	1	0	1
1	0	1	0	0	1
1	1	0	1	1	0
1	1	1	0	1	0

Fig. 12. The truth table of the 3-gene network

$0101 \longrightarrow 1110$

On the other hand by looking at the SP vector, one can see that the entry in the SP corresponding to the state 0100 has the value 1; therefore, following the deletion procedure, the gene g_4 is deleted from states 0100 and 0101 to form state 010 in the reduced network and the transition is selected based on the transition of the state 0100 in the original network, excluding the 4th coordinate. The resulting rule is:

$010 \longrightarrow 110$

By repeating this procedure for all of the states one can make the TT of the reduced network which has 3 genes. Figure 12 shows the resulting TT.

Up to this point, one gene is deleted from the network and the TT of the reduced network is found, therefore, the state transition matrix and steady-state distribution of the 3-gene network are obtainable. Using these items, one can design a control policy on the reduced network and induce it back to the original network. The following section explains this step.

4. Inducing control policy designed on the reduced network to the original network

After *CoD-Reduce* deletes a gene and constructs the TT of the reduced network, it designs a MFPT-CP [8] on the reduced network. That MFPT-CP has 8 coordinates which is the number of states in the 3-gene network. This CP has to be modified

to have the appropriate dimensions (vector of 16 control actions $\in \{0,1\}^{16}$) to be applicable to the original network. One needs to notice that the example here has very small number of genes and only one gene is deleted; in this case the MFPT-CP can be designed on the original network even without any reductions. The cases of having such small number of genes are not realistic. Therefore, the aim of this dissertation is to propose a reduction algorithm, *CoD-Reduce*, that enables us to decrease the complexity of large networks by sequentially deleting the genes. After deleting genes, the control policy that is designed on the reduced network is not applicable to the original network due to the different sizes. The induction of the control policy is the solution for enabling us to apply the control policy of the reduced network on the original network and examine its effects.

The definition for the induction procedure is given in this chapter, and here this process is explained, using the above 4-gene network as an example. Each state in the reduced 3-gene network, corresponds to two states in the original 4-gene network that collapse together. After designing the CP and assigning the control actions to the states of the 3-gene network, in the induction procedure, the control action of each state in the 3-gene network will be duplicated as the control actions for its parent states. For example, the states 0100 and 0101 of the original network form state 010 in the 3-gene network; if the control action of the control policy designed on the reduced network assigns control action of 1 to the state 010, then in the induced control policy, states 0100 and 0101 will have the control action of 1. Figure 13 represents the induction of MFPT-CP from 3-gene network to the 4-gene network.

5. Applying induced CP to the original network

After designing the CP on the 3-gene network and its induction to the 4-gene network, this CP is applied to the network. Applying CP means: for each state, if its control

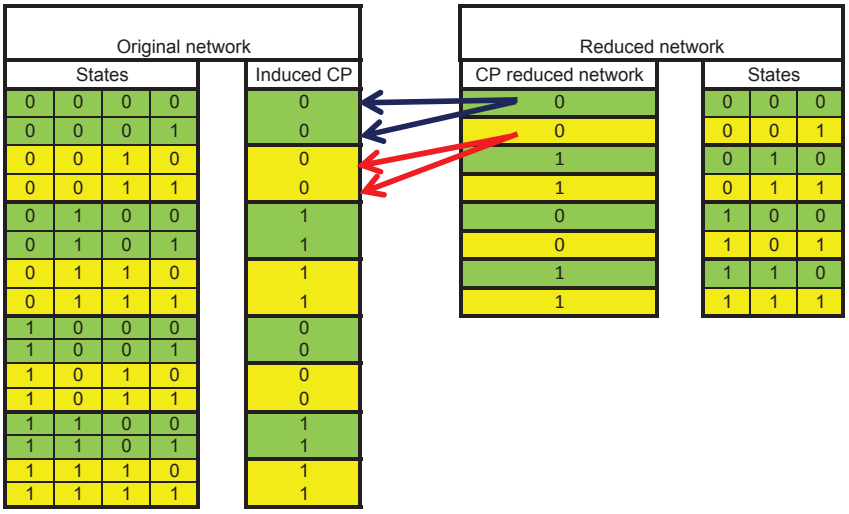

Fig. 13. The induction procedure. The color coding specifies the states that collapse to one state in the reduced network It also displays the duplication of the control actions during induction.

action is 1, then its row in the original STM is replaced by the row corresponding to its flipped state w.r.t. control gene. Following this procedure, the STM is changed and ultimately the SSD of the 4-gene network will be different. The SSD of the network is calculated again and the total probability mass for the desirable and undesirable states of the network are found. By comparing the total SSD probability mass of the desirable states before and after applying control, one can examine the effectiveness of the induced CP. The amount of the changes in the total probability mass of the desirable states is referred to as the shift of SSD toward desirable state. Figure 14 shows that after applying the induced CP, there is considerable shift (about 30%) toward desirable states.

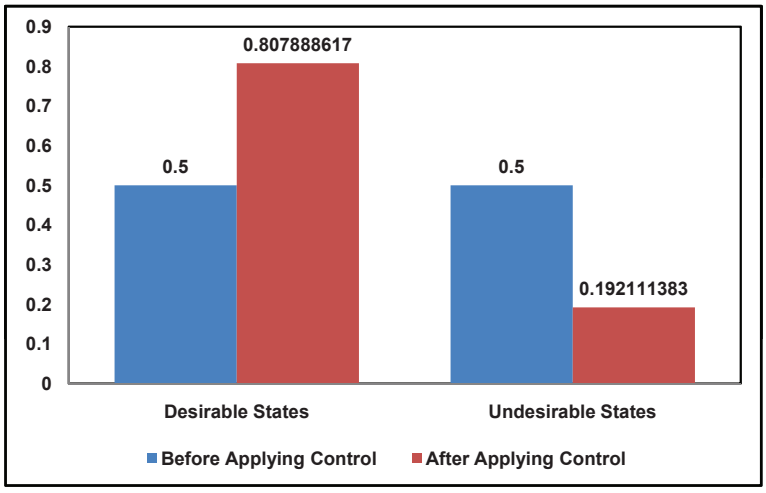

Fig. 14. SSD shift before and after applying the induced MFPT CP

CHAPTER IV

GREEDY CONTROL POLICY*

A. Introduction

To date, the majority of the research regarding intervention in GRNs has been carried out in the context of probabilistic Boolean networks (PBNs) [2]. Assuming random gene perturbation in a PBN, the associated Markov chain is ergodic, and thus it possesses a steady-state distribution (SSD), and theoretically one can always change the long-run behavior using an optimal control policy derived via dynamic programming [4, 5]. However, in practice, the computational requirements of dynamic programming limit this approach to small networks [6, 7]. As an alternative to such optimal intervention, greedy control approaches, e.g. *MFPT-CP* or *SSD-CP* have been proposed [8, 9]; nonetheless, these algorithms have their own computational issues owing to their need to use the state transition matrix (STM) of the Markov chain.

To overcome the computational problems associated with the design of control policies for larger PBNs, Chapter III proposed reduction mappings that delete genes sequentially from the network and finally induce the control policy designed on the reduced network to the original network. However, reduction mappings themselves are computationally demanding [30], [31], and consist of many steps. Additionally, deletion of network components compresses large networks, but it could be at the cost of information loss.

---

* Part of this chapter is reprinted, with permission, from "A cod-based stationary control policy for intervening in large gene regulatory networks", N. Ghaffari, I. Ivanov, X. Quian, and E. Dougherty, *BMC Bioinformatics*, vol. 12, no. S10, 2011.

As a solution, this chapter introduces a new greedy control policy method for designing intervention directly on large Boolean networks. The new method utilizes the CoD and SSD of the model. Thus, the proposed algorithm is called *CoD-CP* because the CoD is the main tool and CP stands for Control Policy [32]. The main advantage of *CoD-CP* in comparison with the previously proposed methods is that it does not require any compression of the original model, and thus can be directly designed on large networks.

The control approach taken in this chapter circumvents many of the computational impediments of previous methods by basing its intervention strategy directly on inter-predictability among genes. Referring to a gene that characterizes a particular phenotype as a *Target (T)* gene and a gene used to alter the long-run behavior of the network by controlling the expression of T as a *Control (C)* gene, the method proposed herein relies on the predictive power of a small group of genes, which includes the control gene, and designs a stationary control policy that alters the steady-state distribution of the model. The algorithm is designed for the specific class of networks where there is a path from the C to T gene – an assumption having a natural interpretation in terms of the biochemical regulatory pathways present in cells. This method simplifies the procedure of designing the stationary control policy and eliminates the need to have a complete knowledge about the STM. It only requires knowledge about the SSD of the network which can be estimated without inferring the STM. The coefficient of determination (CoD) is used for measuring the power of gene interactions [10]. Thus, our new algorithm is optimized for and performs especially well on network models that are inferred from data using CoD-based approaches, e.g. the well-known seed-growing algorithm [15] or the method developed by Chapter VI in this dissertation.

All of the previously proposed methods for working with large GRNs, e.g. *CoD-*

Reduce in Chapter III or state reduction [31], require 'deletion' of network components to achieve a compressed model, which allows for the design of the control policy. An induction step is then required in order to induce those control policies back to the original networks. In general, the benefits of *CoD-CP* are the followings:

- Reduces the complexity of the CP design
- Does not need the state transition matrix of the Markov Chain
- Takes advantage of the power of CoD in measuring the influence of the genes on each other
- Suitable for designing intervention policies in large networks

A series of simulation are performed to validate *CoD-CP* performance. These experiments show that in small networks, where it is possible to derive the currently available greedy MFPT-CP [8] and SSD-CP [9], *CoD-CP* achieves a similar performance. Most importantly, when the size of the network is large and MFPT-CP or SSD-CP cannot be designed directly on the original model, *CoD-CP* is easily constructed and applied to the network without any reduction mappings and induction of the control policy. Section C describes the simulation results. When the network is large, a reduction step is needed before designing the MFPT-CP or SSD-CP. In these cases, *CoD-CP* can be designed directly on the large networks and performs better than the induced MFPT-CP and SSD-CP on average for networks with singleton attractors only or models where cyclic attractors are allowed. The *CoD-CP* is examined for two different perturbation probabilities and the results show consistent patterns.

B. Proposed methodology

This section describes the new algorithm, *CoD-CP*. The algorithm takes advantage of the predictive power of triplets of genes that include the control gene to predict the expression of the target gene with a small estimated error. To achieve the best performance of the algorithm, it is necessary to have a *direct connection* or a *path* from the control gene to the target gene in the regulatory network. The algorithm uses the CoD to measure that predictive power and to design a control policy.

CoD-CP is a greedy technique for designing a stationary control policy. The target gene defines the phenotype and divides states into two mutually disjoint sets, D (desirable) and U (undesirable). The gene with the most predictive power over T among the genes connected with a *path* to T is used as the control gene. The goal of the algorithm is to increase the total probability mass of desirable states in the long-run by controlling C.

CoD-CP starts by generating all 3-gene combinations that include C. These three genes are used for predicting T. *CoD-CP* uses the CoDs for determining the strength of the connection between a target gene and its predictors. The CoDs are calculated using the SSD of the network and the respective conditional probability distribution (CPD) tables. After examining all 3-gene combinations, they are sorted based on their CoDs. The triple that has the maximum CoD with respect to T and its corresponding CPD is stored and used for designing the control policy. If there is more than one such triple, one can uniformly randomly decide to use one of them. This triple is referred to as *MAXCOD* and its CPD is called *MAXCPD*. Table I represents an example of a *MAXCPD* table, where the first three columns contain the binary combinations of the *MAXCOD* genes. Using T and the *MAXCOD* genes, the state space of the network is broken down into *blocks* with 2^{n-4} states. All states

in a *block* share the same values for T and the *MAXCOD* genes. The details about the entries of the *MAXCPD* table are given in the Example 1, part a.

Table I. MAXCPD Table: the first three columns represent the binary combinations of the three MAXCOD genes. The last two columns are filled by summing up the SSD probabilities of states in each corresponding block.

	MAXCOD			*T*	
	C	*Predictor 1*	*Predictor 2*	0	1
row 1	0	0	0	P_{10}	P_{11}
row 2	0	0	1	P_{20}	P_{21}
row 3	0	1	0	P_{30}	P_{31}
row 4	0	1	1	P_{40}	P_{41}
row 5	1	0	0	P_{50}	P_{51}
row 6	1	0	1	P_{60}	P_{61}
row 7	1	1	0	P_{70}	P_{71}
row 8	1	1	1	P_{80}	P_{81}

Example 1, part a: This example explains the entries of the *MAXCPD* table using a 7-gene network with 128 states. Without loss of generality, assume that x_1 and x_2 are the T and C genes, respectively, and $x_1 = 0$ defines the desirable states. After examining all the triples, *MAXCOD* is found to be $\{x_2, x_3, x_4\}$, which has maximum CoD for predicting x_1. The first three columns of the *MAXCPD* table contain 8 binary combinations of x_2, x_3 and x_4, as Table I shows. The last two columns of the table contain the summation of the SSD probabilities of the states with common value for *MAXCOD* genes. The only difference in columns four and five is the value of the

T gene. The size of each *block* of states is $2^{n-4} = 2^3 = 8$. The first *block* is $Block(1) = \{0000000, 0000001, 0000010, 0000011, 0000100, 0000101, 0000110, 0000111\}$, where all have $\{x_2, x_3, x_4\} = 000$ and $x_1 = 0$. The second *block* is $Block(2) = \{1000000, 1000001, 1000010, 1000011, 1000100, 1000101, 1000110, 1000111\}$, where $\{x_2, x_3, x_4\} = 000$ and $x_1 = 1$. Each entry of the forth and fifth columns of the CPD table are represented by P_{ij}, where $i \in \{1, ..., 8\}$ represents a row and $j \in \{0, 1\}$ is the T value. Each P_{ij} is the summation of the SSD probabilities of the states in a *block*. For columns four and five of the first row ($i = 1$), all the SSD probabilities for the states in $Block(1)$ are summed up to find P_{10}. The summation of the SSD probabilities of *Block(2)* forms P_{11}. The rest of the P_{ij}s are calculated similarly.

In the PBN setting, control of the network is achieved by toggling the value of the control gene. The derivation of a stationary control policy, $\mu \in \{0, 1\}^{2^n}$, means defining control actions for each state $\mathbf{s} \in \{StateSpace\}$. If the control action for the state $\mathbf{s}$ is set to 1, it means that the network should transition from its flipped state with respect to C: $\tilde{\mathbf{s}}^c$. Otherwise the network transitions as specified by its STM. The *CoD-CP* algorithm finds the *MAXCPD* table in order to specify the control actions. It uses the total probabilities P_{ij} to define the control actions. Algorithm 4 details all the steps of *CoD-CP*. In the binary representation of each state $\mathbf{s}$, the values of *MAXCOD* genes are found. The decimal conversion of the values of *MAXCOD* genes determines the row of the *MAXCPD* table corresponding to state $\mathbf{s}$. Then, the total probabilities P_{ij} are used to find $D(.)$, as described by algorithm 4, where $D(.)$ defines the difference between the total probability of a *block* of states to be desirable from that of being undesirable in the long run. Using this difference one can define the control actions: if $D(\mathbf{s}) > D(\tilde{\mathbf{s}}^{\mathbf{c}})$, then flip the value of C in $\tilde{\mathbf{s}}^{\mathbf{c}}$ to start the transition

from $\mathbf{s}$; otherwise, flip the value of C in $\mathbf{s}$ and start the next transition of the Markov chain from $\tilde{\mathbf{s}}^{\mathbf{c}}$. If $D(\mathbf{s}) = D(\tilde{\mathbf{s}}^{\mathbf{c}})$, then one can select one of them uniformly randomly. Example 1, part b illustrates how control actions are assigned to the states.

Algorithm 4 CoD-CP - Part 1

Find *MAXCOD* genes and their corresponding *MAXCPD* table
for all the States $\mathbf{s} \in S$, **do**
    Find its flipped State w.r.t. C: $\tilde{\mathbf{s}}^c$
    Find row i of *MAXCPD* by mapping *MAXCOD* genes in $\mathbf{s}$ to the *MAXCPD*
    Find row k of *MAXCPD* by mapping *MAXCOD* genes in $\tilde{\mathbf{s}}^c$ to the *MAXCPD*
    if *T = 0* defines *Desirable States* **then**
        $D(\mathbf{s}) = P_{i0} - P_{i1}$
        $D(\tilde{\mathbf{s}}^c) = P_{k0} - P_{k1}$
    else
        $D(\mathbf{s}) = P_{i1} - P_{i0}$
        $D(\tilde{\mathbf{s}}^c) = P_{k1} - P_{k0}$
    end if
    if $(D(\mathbf{s}) > D(\tilde{\mathbf{s}}^c))$ **then**
        $\mu(\mathbf{s}) = 0$
        $\mu(\tilde{\mathbf{s}}^c) = 1$
    else if $(D(\tilde{\mathbf{s}}^c) > D(\mathbf{s}))$ **then**
        $\mu(\mathbf{s}) = 1$
        $\mu(\tilde{\mathbf{s}}^c) = 0$

Algorithm 5 CoD-CP - Part 2

19: **else**

20: Uniformly randomly assign control actions For $\mathbf{s}$ and $\tilde{\mathbf{s}}^c$ such that only one has $\mu(.) = 1$

21: **end if**

22: **end for**

23: return (μ)

Example 1, part b: Following the same 7-gene example, consider state $\mathbf{s} = 0000000$. The $D(\mathbf{s}) = P_{10} - P_{11}$ is calculated. The flipped state with respect to the control gene is $\tilde{\mathbf{s}}^{\mathbf{c}} = 0100000$. Looking at the *MAXCOD* genes in the binary representation of $\tilde{\mathbf{s}}^{\mathbf{c}}$, one can see that $\{C = 1, Predictor1 = 0, Predictor2 = 0\}$, which maps to row 5 of the *MAXCPD* table. Similarly, $D(\tilde{\mathbf{s}}^{\mathbf{c}}) = P_{50} - P_{51}$. If $D(\mathbf{s}) > D(\tilde{\mathbf{s}}^{\mathbf{c}})$, then it is beneficial to flip $\tilde{\mathbf{s}}^{\mathbf{c}}$ and force the Markov chain to start the next transition from $\mathbf{s}$, but if $D(\tilde{\mathbf{s}}^{\mathbf{c}}) > D(\mathbf{s})$, then it is better to start the next transition from $\tilde{\mathbf{s}}^c$, in which case the control action for $\mathbf{s}$ is set to 1. For all the states in *Block(1)* the same control action is applied. This greatly simplifies the design of the control policy. Figure 15 shows a numerical example of how the *CoD-CP* can be designed on this 7-gene example network.

C. Performance Comparison

This section compares the performances of *CoD-CP*, SSD-CP, and MFPT-CP, first with respect to run time and then to shift of the steady-state distribution.

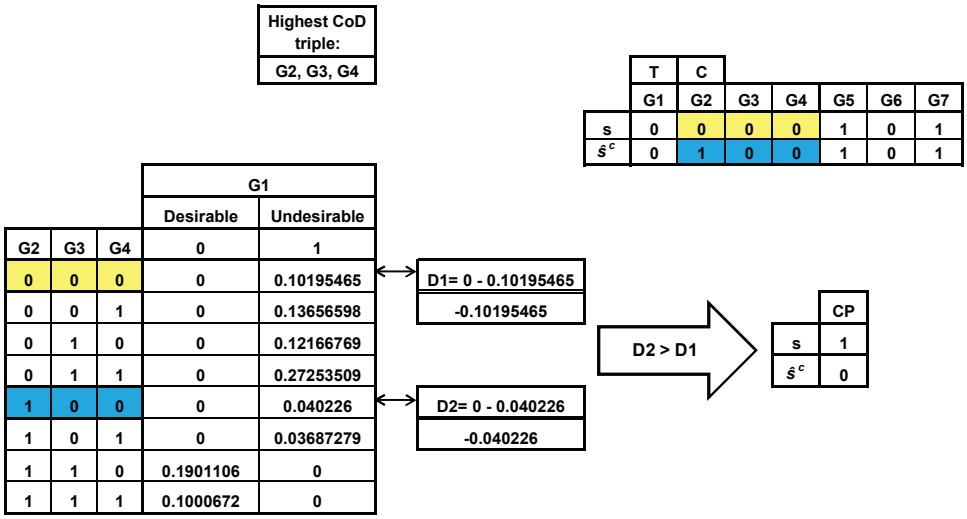

Fig. 15. Deriving CoD-CP for a small 7-gene network. The x_1 and x_2 genes are the T and C genes, respectively. $x_1 = 0$ defines *Desirable* states. The *MAXCOD* genes are: $\{x_2, x_3, x_4\}$. The control action for state $\mathbf{s}$ is 1 and the control action for state $\tilde{\mathbf{s}}^{\mathbf{c}}$ is 0, because *D(2) > D(1)*.

1. Run-time Comparison

The dynamics of a GRN and its associated Markov chain are determined by its state transition matrix. The STM provides the full knowledge about the states and their transitions in the network; however, inferring the STM is difficult, especially when available data about the network are limited or the size of the network is large. The main advantage of the *CoD-CP* algorithm is that it can be directly designed on large networks without inferring the STM and only needs an estimation of the SSD of the Markov chain. This section provides a comparison of *CoD-CP* with MFPT-CP [8] and SSD-CP [9].

The necessary reduction and induction steps increase the computational time associated with MFPT-CP and SSD-CP. To compare the three algorithms, the running times needed for designing control policies are measured on gene networks containing 7, 8, 9, and 10 genes, averaged for 100 randomly designed $BN_p s$. For MFPT-CP and

SSD-CP, the best gene for deletion was selected and then the original network was reduced by deleting that gene, according to the methodology introduced in Chapter III. Consequently, the control policies were designed on the reduced networks and then induced back to the original networks. *CoD-CP* was designed directly on the original network as described by this chapter. All computations were performed on a computer with 4GB of RAM and Intel(R) Core(TM) i5 CPU, 2.53 GHz. Figure 16 shows the average running times for 100 BN_ps in seconds. The running times tend to grow exponentially as the number of genes increases.

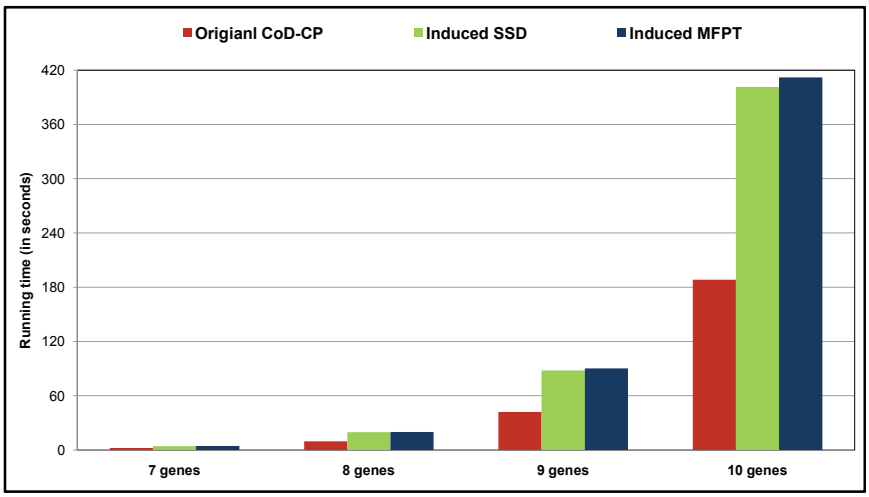

Fig. 16. Comparing the average running times (in seconds) for designing stationary control policy for 100 randomly generated 10-gene, 9-gene, 8-gene and 7-gene BN_ps. Running time for CoD-CP algorithm is always less than MFPT-CP and SSD-CP. The running time grows exponentially as the number of genes increases.

For comparing the performance of the three algorithms one needs to keep in mind their important characteristics. The *CoD-CP* algorithm needs the SSD to design the control policy. In cases when the SSD is known, one can directly proceed to the CoD calculations and design the control policy for the network. When the SSD is

not known, it can be calculated using equation (2.1) or can be estimated by methods described in [33]. The model-dependent version of the MFPT algorithm requires an extra step to infer the STM. It then uses matrix inversion to find the mean-first-passage-time vectors K_D and K_U, this step having the same time complexity as finding the SSD. The model-free version of MFPT-CP requires time-course measurements to estimate the necessary mean-first-passage-time vectors. In such a case the algorithm can skip the inference of the STM, and the complexity of estimating MFPT vectors is constant with respect to the number of genes. However, the availability of time-course data is very limited in practice. The other available greedy approach, SSD-CP, also requires the SSD and STM of the network. Moreover, the SSD-CP algorithm needs to find the perturbed SSD for each state, which increases the time spent for designing the control policy.

As described in the section B, *CoD-CP* uses the *MAXCPD* table to design the control policy, which divides the state space into *blocks* of size 2^{n-4}. These *blocks* are used to assign the same control actions to all of the states in a given *block* and the complement control action for the *block* of flipped states. This significantly reduces the complexity of the control policy design and leads to shorter run times.

2. Generating Synthetic Networks and Their Characteristics

This section provides simulation experiments to demonstrate the performance of the *CoD-CP* algorithm with respect to its main goal, to shift undesirable steady-state mass to desirable steady-state mass. The algorithm is applied to randomly generated networks. *CoD-CP* has been designed for networks that are too large for direct application of greedy algorithms such as MFPT-CP and SSD-CP while at the same time not suffering from loss of information when designing control polices on reduced networks and then inducing them to the corresponding original networks. Hence, the

desire is to demonstrate the improved performance of *CoD-CP* in comparison to the induced greedy control policies when reduction-inducement is necessary; otherwise, one can simply use the previously developed greedy policies directly. This section discusses the results of a simulation study that compares the performance of *CoD-CP* to MFPT-CP and SSD-CP on a set of $BN_p s$ that are randomly generated using the algorithm from [14], for two different perturbation probabilities: $p = 0.1$ and $p = 0.01$. The latter probability is the one most commonly used in GRN control studies [4, 34, 8, 9]; nevertheless, also $p = 0.1$ is used to see the effect, if any, of a less stable network where less mass is concentrated in the attractors. In order to examine how the attractor structure affects performance, the *CoD-CP* algorithm is tested on two model classes:

1. networks with singleton attractors only

2. networks that allow cyclic attractors

In the first class, 100 unique attractor sets are chosen randomly for a different number of genes n, where $n \in \{7, 8, 9, 10\}$. The attractor sets are restricted to be evenly distributed between the desirable and undesirable states. In the second class, the attractor sets are unique, but the criteria for evenly distribution between D and U is no longer required and attractors are allowed to be cyclic and of unequal length. The algorithm's performance is measure by *absolute shift* of the SSD, defined by:

$$\lambda = \sum_{d \in D} \tilde{\pi_d} - \sum_{d \in D} \pi_d, \tag{4.1}$$

where $\sum_{d \in D} \tilde{\pi_d}$ and $\sum_{d \in D} \pi_d$ are the total probability masses of the desirable states after applying control and before applying any control, respectively, a larger λ being desirable.

In real-world situations the target and control genes are often pre-selected by the biologists/clinicians, the basis for choice being that a phenotypically related target is to be up- or down-regulated and the control gene is known to be related to the target. However, in the simulation studies, where knowledge about T and C does not exist, a procedure is needed to identify reasonable target and control genes. The objective of the procedure is to select a (C, T) pair such that there is a *direct connection*, or *path*, from C to T, which would be a natural constraint in applications. The strength of connection between C and T is measured by the CoD. The selected pair is called *CoD-strongly-connected* pair. To select this pair, all two-gene combinations are constructed such that each gene in a given pair is treated as both the candidate target and candidate control gene, and the CoD of the candidate C for predicting candidate T is calculated. The pair with the maximum CoD of C candidate for predicting candidate T is picked. Then the algorithm checks if there is a path from C to the T. If such a *path* exists, then the (C, T) pair is chosen. If no *path* exists, then the pair is discarded and the next highest CoD pair is considered as the candidate (C, T) pair.

For checking of the existence of a *direct connection* or *path* between candidate T and C genes, the connectivity table of the network is used. This table is built using the truth table of the BN_p as follows: if toggling the value of a predictor gene affects the value of a target gene, then the corresponding entry of the table has 1, otherwise it has 0. Therefore, if there is a *direct connection* between C and T genes, then the corresponding entry in the connectivity table has value 1. This implies that if for the pair T-C, a *path* from C to T exists, it also means that control gene can affect the target gene, based on the truth table.

If there is no connection between the T-C pair, there is still a possibility of having a *path* which consist of more than these two genes. For checking the existence

of an *indirect path(s)*, the Breadth-First-Search(BFS) algorithm [35] is used. The BFS finds all the nodes that are *reachable* from the given source node.

This procedure is repeated until the *CoD-Strongly-Connected T-C pair* is found.

3. Effect on the SSD of the networks

To compare *CoD-CP* to the reduction-inducement versions of MFPT-CP and SSD-CP, the reduction method in Chapter III and [30] is used, called *CoD-Reduce*. The *CoD-Reduce* algorithm is designed for the networks with singleton attractors only because its selection policy heuristically uses the singleton attractors to generate the structure of the reduced network. Therefore, in this chapter, when reduction of the network is needed for comparison of the control policies, the networks with singleton attractors only are used. Figure 17 illustrates that the *CoD-CP* designed on the original network outperforms the induced MFPT-CP and SSD-CP policies when there is significant network reduction in the case of a 10-gene network and $p = 0.1$. Each set of bars in the graph shows the average SSD shifts for the three policies with different amounts of reduction for the MFPT-CP and SSD-CP policies, beginning no reduction-induction, then reduction to 9 genes and induction back to 10, and so on. The performance of *CoD-CP* is invariant because it is designed directly from the original network. In the absence of reduction, the *CoD-CP* is outperformed by the induced polices and continues to be outperformed with a 2-gene reduction. But after that, for reductions of 3 ore more genes, *CoD-CP* outperforms the induced policies, with its superiority increasing as the extent of the reduction grows. This is precisely the desired behavior. While both the MFPT-CP and SSD-CP policies can be used directly for 10-gene networks, they must be induced from reductions for large networks and, as it is observed, the reduction-induction paradigm provides decreasing SSD shift as the amount of reduction increases. Figure 18 shows a similar

Table II. Using a *CoD-Strongly-Connected T-C pair*: Comparing the MFPT-CP and SSD-CP with the proposed CoD-CP. The absolute SSD shift toward desirable states, averaged for 100 $BN_p s$ with 10 genes, 100 $BN_p s$ with 9 genes, 100 $BN_p s$ with 8 genes and 100 $BN_p s$ with 7 genes. Singleton attractors with perturbation probability $p = 0.1$

Control Policy	7 genes	8 genes	9 genes	10 genes
CoD-CP	0.322014658	0.256250706	0.205475527	0.162102211
MFPT CP	0.323669615	0.278172941	0.235560338	0.215092042
SSD CP	0.31548592	0.274780118	0.235272982	0.217780913

phenomenon with $p = 0.01$.

4. Effect of cyclic attractors and the selection of target-control pairs

Having demonstrated the advantage of *CoD-CP* over the induced polices as the degree of reduction (and, therefore, induction) increases, this section turns to two other aspects of *CoD-CP*: the effect of cyclic attractors and the selection of target-control pairs. For each issue, two cases are considered. For attractors, as previously noted, there are two cases: (a) only singleton attractors and (b) cyclic attractors allowed. Regarding target-control pairs, there are two possibilities: (a) *CoD-strongly-connected* target-control pairs and (b) randomly selected target-control pairs. Combining these choices, there are four factors to consider: network size (n), perturbation probability (p), attractor structure, and target-control structure. Tables II, III, IV, V, VI, VII, VIII, IX provide the SSD shifts for network size $n \in \{7, 8, 9, 10\}$, $p \in \{0.1, 0.01\}$, and the two possibilities for attractors and target-control pairs.

The first point to recognize is that using *CoD-strongly-connected* target-control pairs are more realistic because in practice one would control a target with a gene

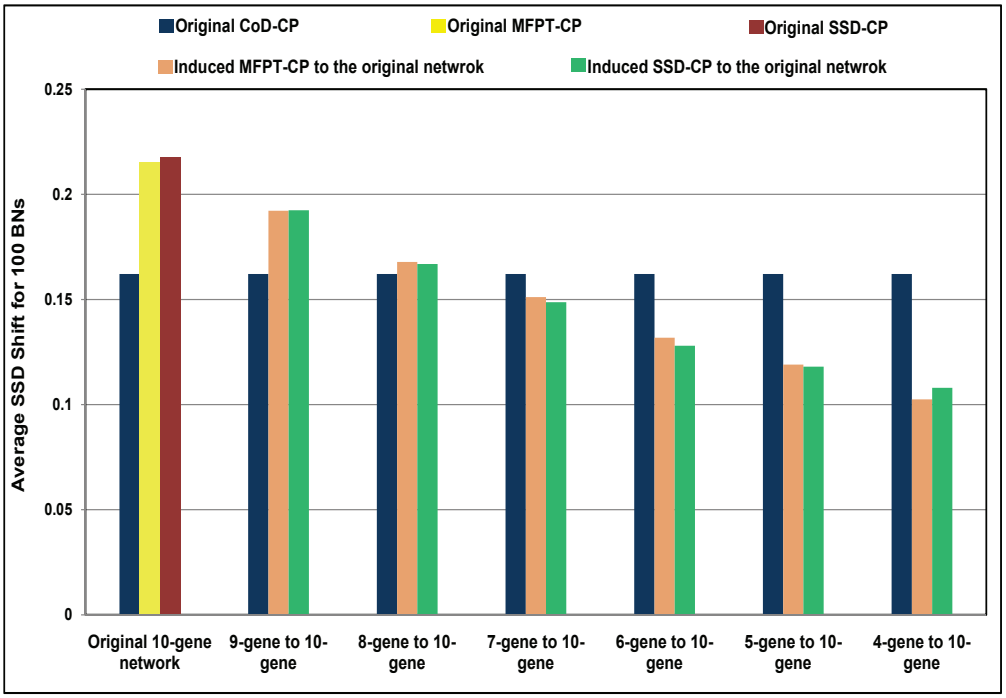

Fig. 17. Comparing original CoD-CP to the original and induced MFPT-CP and SSD-CP for 100 randomly generated 10-gene $BN_p s$ with half of the attractors in D states. In the first set of bars, CoD-CP, MFPT-CP and SSD-CP are designed on the 10-gene networks. In the next sets, the CoD-CP was designed on the original 10-gene networks and compared to the induced MFPT-CP and SSD-CP. At each step, one gene was deleted, and then MFPT-CP and SSD-CP were designed and induced back to the original network, until each BN_p had only 4 genes. The perturbation probability is 0.1.

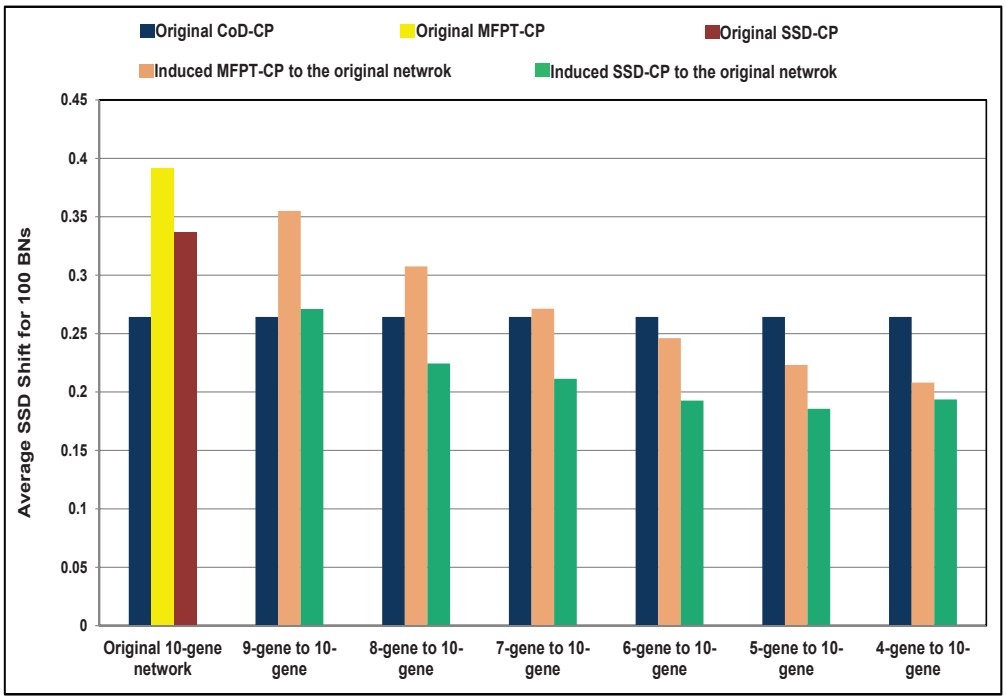

Fig. 18. Comparing original CoD-CP to the original induced MFPT-CP and SSD-CP for 100 randomly generated 10-gene $BN_p s$ with half of the attractors in D states. In the first set of bars, CoD-CP, MFPT-CP and SSD-CP are designed on the 10-gene networks. In the next sets, the CoD-CP was designed on the original 10-gene networks and compared to the induced MFPT-CP and SSD-CP. At each step, one gene was deleted, and then MFPT-CP and SSD-CP were designed and induced back to the original network, until each BN_p had only 4 genes. The perturbation probability is 0.01.

Table III. Randomly choosing the target and control genes: Comparing the MFPT-CP and SSD-CP with the proposed CoD-CP. The absolute SSD shift toward desirable states, averaged for 100 $BN_p s$ with 10 genes, 100 $BN_p s$ with 9 genes, 100 $BN_p s$ with 8 genes and 100 $BN_p s$ with 7 genes. Singleton attractors with perturbation probability $p = 0.1$.

Control Policy	7 genes	8 genes	9 genes	10 genes
CoD-CP	0.072929047	0.069787742	0.069085431	0.043901727
MFPT CP	0.108511867	0.119247253	0.125959351	0.125947219
SSD CP	0.116194205	0.133758479	0.142706132	0.141898815

Table IV. Using a *CoD-Strongly-Connected T-C pair*: Comparing the MFPT-CP and SSD-CP with the proposed CoD-CP. The absolute SSD shift toward desirable states, averaged for 100 $BN_p s$ with 10 genes, 100 $BN_p s$ with 9 genes, 100 $BN_p s$ with 8 genes and 100 $BN_p s$ with 7 genes. Cyclic attractors with perturbation probability $p = 0.1$

Control Policy	7 genes	8 genes	9 genes	10 genes
CoD-CP	0.315366842	0.237692755	0.191486782	0.132842645
MFPT CP	0.320002278	0.254221118	0.225470299	0.192352519
SSD CP	0.320124817	0.257447572	0.229185616	0.19785175

Table V. Randomly choosing the target and control genes: Comparing the MFPT-CP and SSD-CP with the proposed CoD-CP. The absolute SSD shift toward desirable states, averaged for 100 BN_ps with 10 genes, 100 BN_ps with 9 genes, 100 BN_ps with 8 genes and 100 BN_ps with 7 genes. Cyclic attractors with perturbation probability $p = 0.1$.

Control Policy	7 genes	8 genes	9 genes	10 genes
CoD-CP	0.078443655	0.045092185	0.057163771	0.044105186
MFPT CP	0.110781996	0.091997738	0.116346892	0.124227404
SSD CP	0.124815432	0.114307555	0.136428735	0.141501075

Table VI. Using a *CoD-Strongly-Connected T-C pair*: Comparing the MFPT-CP and SSD-CP with the proposed CoD-CP. The absolute SSD shift toward desirable states, averaged for 100 BN_ps with 10 genes, 100 BN_ps with 9 genes, 100 BN_ps with 8 genes and 100 BN_ps with 7 genes. Singleton attractors with perturbation probability $p = 0.01$

Control Policy	7 genes	8 genes	9 genes	10 genes
CoD-CP	0.442813327	0.43722164	0.343500124	0.26431826
MFPT CP	0.444933823	0.474439814	0.417636081	0.391189057
SSD CP	0.429896505	0.431245347	0.368024051	0.337230131

Table VII. Randomly choosing the target and control genes: Comparing the MFP-T-CP and SSD-CP with the proposed CoD-CP. The absolute SSD shift toward desirable states, averaged for 100 $BN_p s$ with 10 genes, 100 $BN_p s$ with 9 genes, 100 $BN_p s$ with 8 genes and 100 $BN_p s$ with 7 genes. Singleton attractors with perturbation probability $p = 0.01$.

Control Policy	7 genes	8 genes	9 genes	10 genes
CoD-CP	0.117863711	0.148585675	0.160006514	0.102484183
MFPT CP	0.170811757	0.282492387	0.317528279	0.330404616
SSD CP	0.185170107	0.315283799	0.34151411	0.318055021

Table VIII. Using a *CoD-Strongly-Connected T-C pair*: Comparing the MFPT-CP and SSD-CP with the proposed CoD-CP. The absolute SSD shift toward desirable states, averaged for 100 $BN_p s$ with 10 genes, 100 $BN_p s$ with 9 genes, 100 $BN_p s$ with 8 genes and 100 $BN_p s$ with 7 genes. Cyclic attractors with perturbation probability $p = 0.01$

Control Policy	7 genes	8 genes	9 genes	10 genes
CoD-CP	0.438436274	0.309464685	0.288786759	0.204716543
MFPT CP	0.451949369	0.341847358	0.353542661	0.324093223
SSD CP	0.42385878	0.299621849	0.288726556	0.265043692

Table IX. Randomly choosing the target and control genes: Comparing the MFPT-CP and SSD-CP with the proposed CoD-CP. The absolute SSD shift toward desirable states, averaged for 100 $BN_p s$ with 10 genes, 100 $BN_p s$ with 9 genes, 100 $BN_p s$ with 8 genes and 100 $BN_p s$ with 7 genes. Cyclic attractors with perturbation probability $p = 0.01$.

Control Policy	7 genes	8 genes	9 genes	10 genes
CoD-CP	0.069442199	0.068265255	0.116798596	0.06892594
MFPT CP	0.175905537	0.202021369	0.26174396	0.265980677
SSD CP	0.196300574	0.233026013	0.274143933	0.265980677

that is strongly connected to it via prediction and the CoD is a measure of prediction. On the other hand, one could hardly expect to achieve as good results by randomly selecting targets and controls. In addition, as the tables show, using *CoD-strongly-connected* target-control pairs results in decreasing SSD shift for increasing network size, whereas this trend is replaced by sporadic behavior for randomly selected target-control pairs. Finally, one can observe the better performance for $p = 0.01$ than for $p = 0.1$. This reflects the more random network behavior for higher perturbation probability because the control algorithm utilizes the predictive structure in the network (as measured by the CoD) and this structure is less determinative when perturbations are more likely. In this regard, it is noted that both MFPT-CP and SSD-CP also perform better for $p = 0.01$ than for $p = 0.1$, in both their non-induced and induced modes.

5. Statistical Testing

Furthermore, to examine the effects of the attractor structure of the network on the performance of each control policy, the *CoD-CP*, MFPT-CP and SSD-CP are designed

for and applied to 100 randomly generated $BN_p s$ with 10 genes. The *two-sample t-test assuming unequal variance* is performed on the absolute SSD shift of the 100 networks with two major classes of attractors and for two perturbation probabilities. These tests show that the performances of all of the control policies are statistically different (in the significance level of 0.05) for the networks with only singleton attractors from the ones that permit cyclic attractors, for perturbation values $p = 0.1$ and $p = 0.01$. In the case of perturbation value $p = 0.1$, the *p-values* for the *CoD-CP*, MFPT-CP and SSD-CP are 4.8×10^{-4}, 1.3×10^{-4} and 3.2×10^{-4}, respectively. For $p = 0.01$, the the *p-values* are 4.8×10^{-3}, 1.1×10^{-5} and 1.4×10^{-8}, for the *CoD-CP*, MFPT-CP and SSD-CP respectively. Comparing the *p-values* illustrates that the performance of the *CoD-CP* has the least change among the three policies when the attractor structure of the networks changes. This means that the *CoD-CP* is more robust with respect to the attractor structure of the network.

CHAPTER V

ALGORITHMS FOR GENERATION OF SYNTHETIC BOOLEAN NETWORKS AND NETWORKS ISOMORPHISMS

A. Introduction

Representing GRNs via mathematical models with an ultimate intention of perturbing the long-run behavior toward more desirable states is the main focus of translational genomics. The mathematical models need to be constructed from the available data such as microarray measurements. This approach is addressed in [15] and a new inference method is proposed in Chapter VI of this dissertation. At the same time, it is important to understand and study the properties of Boolean network model, and the impact they might have on the process of designing control policies. In this context, simulation studies based on synthetically generated BNs have attracted significant research interest in recent years.

It is believed that the attractors play an important role in the the long-run behavior of the network and the majority of the probability mass of the SSD is concentrated in the attractors [13]. Considering this fact, Pal et. al [14] addressed the problem of generating GRNs from a prescribed set of attractors. The algorithm proposed in [14] generates synthetic Boolean networks with prescribed attractors, while enforcing more constraints on the connectivity and structure of the networks. They developed two algorithms:

1. Algorithm 1: randomly generates the truth table according to the prescribed set of attractors and incorporates the compatibility between the attractors structure and gene predictors. It also limits the minimum and maximum number of levels in the networks, where the level is the distance of each state from the root.

2. Algorithm 2: randomly generates the state transition diagram, which dynamically represents the states and their transitions, and then checks the validity of the constraints on the generated network.

Algorithm 1 is widely used in studies that focus on large sets of synthetically generated PBNs. However, this process could generate networks that have identical dynamical structure, after mapping based relabelings of their genes. Thus, there is no grantee that when one generates large sets of BNs with a given set of properties, one will not bias that set toward a certain class of networks. It becomes important to find sufficient conditions that allow an unbiased generation of sets of synthetic networks by algorithm 1 in [14]. In this dissertation, two Boolean networks are called *isomorphic* if there exists a gene relabeling that maps them to each other. This chapter is dedicated to developing of sufficient conditions to avoid isomorphic networks, while using algorithm 1 in [14].

The *state transition diagram* of a Boolean network is a graph and therefore, BNs can be studied in the context of *graph theory*. This chapter starts by general definitions of *graphs* and *trees* and adapts these concepts for Boolean networks. The isomorphism in the context of Boolean networks is studied in section B, which also introduces two new algorithms for discarding isomorphic BNs: 1- with singleton attractors only; 2- with cyclic attractors.

B. Isomorphism in the context of the Boolean networks

This chapter studies *isomorphisms* in the context of Boolean networks, without being concerned about gene perturbations. The state transition diagram of a BN is a *graph*. The following section provides the general definitions of the *graphs*, *trees* and *isomorphisms* between them. It utilizes these concepts to define the *isomorphism* of

Boolean networks.

1. Definitions

Definition 4. *A directed graph* $\mathbf{G}$ *is a pair,* $(\mathbf{V}, \mathbf{E})$*, where* $\mathbf{V}$ *is a finite set and* $\mathbf{E}$ *is a binary relation on* $\mathbf{V}$*. The set* $\mathbf{V}$ *is called the* ***vertex set*** *of* $\mathbf{G}$*, and its elements are called* ***vertices****. The set* $\mathbf{E}$ *is called the* ***edge set*** *of* $\mathbf{G}$*, and its elements are called* ***edges****[35].*

According to the above definition, the state transition diagram of a BN is a *directed graph*, consisting of nodes that represent the states and edges that represent the transitions.

Definition 5. *Two graphs* $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ *and* $\mathbf{G}' = (\mathbf{V}', \mathbf{E}')$ *are* ***isomorphic*** *if there exists a bijection* $f : \mathbf{V} \to \mathbf{V}'$ *such that* $(u, v) \in E$ *if and only if* $(f(u), f(v)) \in E'$*[35].*

Thus, the isomorphisms of graphs could be thought as a relabeling of vertices of $\mathbf{G}$, maintaining the corresponding edges in $\mathbf{G}$ and $\mathbf{G}'$ [35].

Trees are special form of graphs and can be defined as:

Definition 6. *A directed graph with no cycles is called a* ***directed acyclic graph****. A (directed)* ***tree*** *(sometimes called a rooted tree), is a directed acyclic graph satisfying the following properties[36]:*

1. *There is exactly one vertex, called the* ***root****, which no edges enter.*
2. *Every vertex except the root has exactly one entering edge.*
3. *There is a path (which is easily shown unique) from the root to each vertex.*

Boolean networks can be divided into two classes, based on their attractor structure: 1- BNs with singleton attractors only; 2- BNs with cyclic attractors. Class 1 of

the BNs can be viewed as a collection of trees if the direction of each edge is reversed, and the self-referencing attractor edges are removed. Therefore, in this dissertation, a BN with singleton attractors only is referred to as a ***k-BN-tree***.

For defining isomorphisms for *k-BN-tree*s, let's start by describing the general tree isomorphisms. Aho et al. [36] define two trees to be *isomorphic* if one can map a tree into the other by permuting the order of the sons of vertices. They developed an algorithm that determines if two trees are isomorphic in *O(n)* time, where n is the number of vertices [36]. The algorithm works through the trees level-by-level toward the roots and assigns integers to vertices in two trees. The two trees are *isomorphic* if and only if their roots have the same integer at the end of the process.

It is important to observe that this graph-theoretical definition of isomorphism between trees has little to do with the dynamic of a BN as represented by its state transition diagram. One has to keep in mind that *k-BN-tree*s consist of vertices that are the states of the respective Boolean network. In this dissertation, the concept of isomorphism for *k-BN-tree*s relates to the possibility of mapping one *k-BN-tree* to another *k-BN-tree*, by a permutation of their gene labels, assuming that both networks are defined on the same collection of genes. The following section discusses this concept.

C. Discussion

In order to generate unbiased sets of the synthetically generated BNs, one needs to ensure that the networks included in that set, cannot be mapped to each other by relabeling their genes. This section starts by defining the isomorphism in the context of *k-BN-tree*s. Afterwards, it provides two algorithms that ensure that the sets of synthetically generated BNs are unbiased in the sense that they do not contain any

isomorphic BNs.

1. Isomorphism of *k-BN-tree*s

It is important to note that a gene relabeling not only changes the enumeration of the states of a *k-BN-tree*, but can also move a state to a different basin of attraction, or it can change the position of a state within the same basin of attraction; Figure 19 discusses such an example. Furthermore, while *gene relabeling* changes the enumeration of the states, the attractor structure and the number of states within a basin of an attractor will not change.

In Figure 19 two *2-BN-tree*s are mapped to each other by relabeling of their genes. In part (a), a given BN and its corresponding state transition diagram are provided. The relabeling is based on the following permutation: $(1, 2, 3, 4) \rightarrow (3, 2, 1, 4)$. The state transition diagram shows that the network is essentially the same as the *2-BN-tree* in part a, in terms of attractor structure and the basin of the attractors.

The following definition provides the concept of isomorphic *k-BN-tree*s:

Definition 7. *Two k-BN-trees care called isomorphic if they can be mapped to each other by a gene relabeling.*

The objective of this section is to find sufficient conditions that allow for elimination of isomorphic *k-BN-tree*s, from a set of synthetically generated *k-BN-tree*s by algorithm 1 in [14]. In order to describe such sufficient conditions, the notion of *semi-isomorphism* for *k-BN-tree*s are introduced. The *semi-isomorphism* is used in a novel algorithm that ensures that no isomorphic *k-BN-tree*s remain in a set generated by algorithm 1 in [14]. In what follows, it is assumed that the set of all *k-BN-tree*s on fixed number of genes, is *partially ordered*. In particular, a *k-BN-tree* has its attractor states $\{S_1, \cdots S_k\}$, always ordered by non-decreasing cardinality of their respective

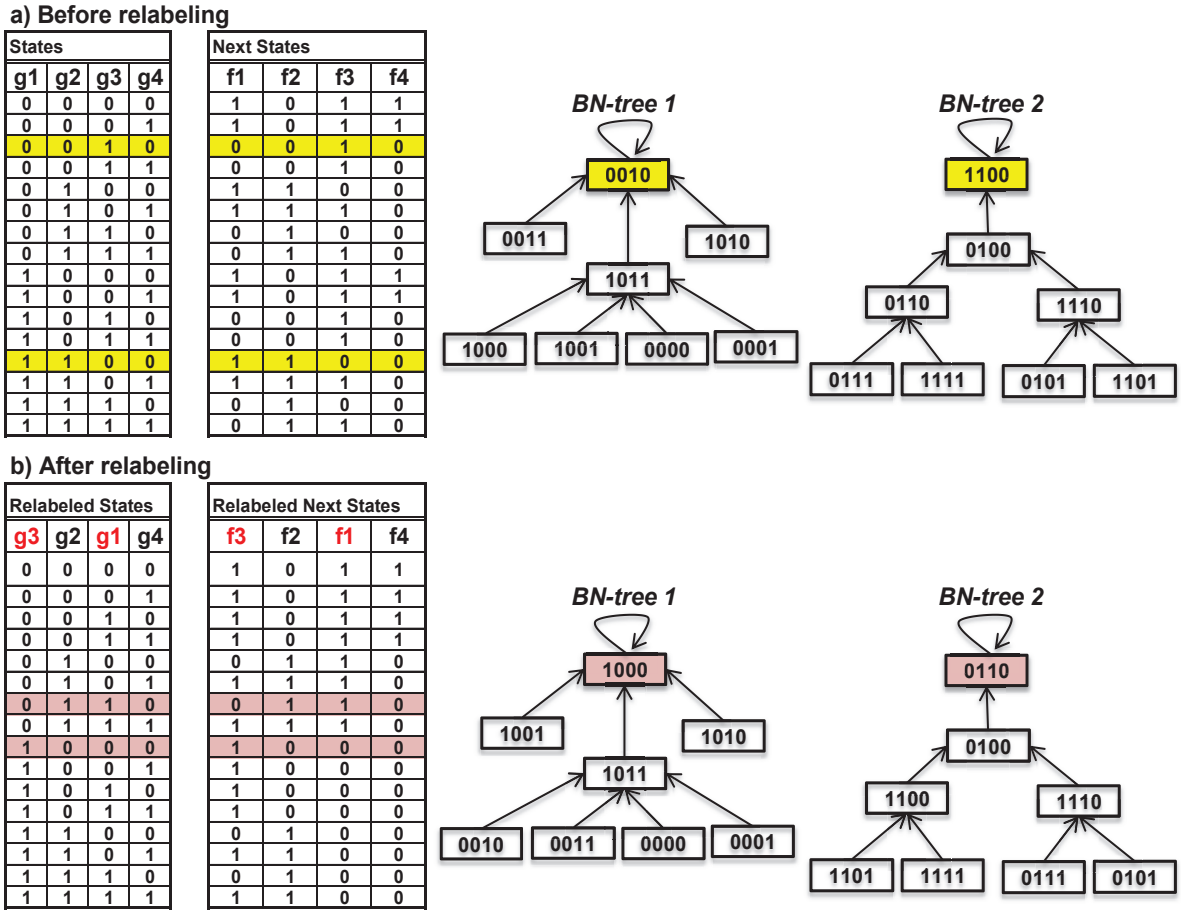

Fig. 19. Two isomorphic *2-BN-trees*. a) A Boolean network with two singleton attractors, states {0010, 1100}, represented as two *BN-trees* called *BN-tree 1* and *BN-tree 2*. b) A relabeling exchanges columns one and three in the truth table; then the truth table is re-ordered, creating a new *2-BN-tree*, with attractors {1000, 0110}. The *BN-tree 1* and *BN-tree 2* in parts a and b, have a common attractor structure and matching basin for their attractors, therefore, they are isomorphic.

basins $\{B_1, \cdots B_k\}$, i.e. $\|B_1\| \leq \|B_2\| \leq ... \leq \|B_k\|$. Under this convention, the partial order is defined as: *k-BN-tree(1)* $\preceq$ *k-BN-tree(2)* if $\|B_i(1)\| \leq \|B_i(2)\|$ for $i = 1, ..., k$. The following defines the concept of *semi-isomorphism* for *k-BN-tree*s:

Definition 8. *Two k-BN-trees, k-BN-tree(1) and k-BN-tree(2) are semi-isomorphic if* $\|B_i(1)\| = \|B_i(2)\|$*, for* $i = 1, ..., k$.

Table X represents two *4-BN-tree*s with *semi-isomorphism*. There are 7 genes and 128 states in each network and four states are randomly selected to be the attractor states.

Table X. Two networks with 7 genes and 128 states: *4-BN-tree (1)* and *4-BN-tree (2)*, are *semi-isomorphic*. Four states are randomly chosen to be the attractor states. The number of states within the basin of attractors vary, but their partial order is the same.

Network	Attractor	Number of States within the Basin
4-BN-tree (1)	5	16
	33	16
	80	48
	98	48
4-BN-tree (2)	3	48
	46	16
	77	48
	112	16

The notion of *semi-isomorphism* provides sufficient condition to avoid isomorphic BNs, while generating synthetic networks using algorithm 1 outlined in [14]. It is important to note that if two *k-BN-tree*s are isomorphic, then they are neces-

sary semi-isomorphic. This means that if the semi-isomorphic BNs are discarded, no isomorphic networks are left in the set of synthetically generated BNs. As a conservative approach, this dissertation proposes discarding one of the networks for every semi-isomorphic pair of *k-BN-tree*s. Section 2 provides more details.

2. An algorithm for avoiding isomorphic *k-BN-tree*s

This section introduces an algorithm for avoiding synthetic isomorphic *k-BN-tree*s, generated by the method outlined in [14]. The notion of semi-isomorphism, defined in this chapter, provides sufficient condition for avoiding the networks that can be mapped to each other by a gene relabeling. Therefore, one can ensure that by filtering out semi-isomorphic *k-BN-tree*s, the remaining networks are *non-isomorphic*. The algorithm conservatively eliminates one of the networks for every pair of semi-isomorphic *k-BN-tree*s, which makes the algorithm very fast and efficient.

Algorithm 6 provides the steps that are needed for ensuring the synthetically generated *k-BN-tree*s are non-isomorphic. In the case when N non-isomorphic BNs are needed, nN networks are produced. This is necessary because for every pair of networks that are semi-isomorphic, only one of them is saved and the other one is discarded. The parameter n can be selected by the user, and should be at least 2. The total of nN *k-BN-Tree*s are generated to expect that N networks will be non-isomorphic. The algorithm starts by randomly generating nN sets of k non-repeating numbers, where k is the number of prescribed singleton attractors. Then, the algorithm in [14] is used to generate nN networks, using the sets of prescribed attractors. In the next step, for every pair of semi-isomorphic networks, one of them is discarded and the other one is saved. And finally, the algorithm ensures that the total number of saved networks is more than or equal to the needed number of networks. It is possible that even by generating nN *k-BN-Tree*s, at least N non-

Algorithm 6 Discarding isomorphic Boolean networks with singleton attractors

Input N: total number of needed non-isomorphic BNs
Input M: total number of iterations
Input k: total number of attractor States for each BN
Input n
while M $\neq 0$ **do**
    Generate nN sets of numbers, where each set consists of exactly k non-repeating numbers
    BN-tree $\leftarrow$ Generate nN networks by the algorithm outlined in [14], using the sets with exactly k unique numbers as prescribed singleton attractors
    BN-tree-UNIQUE $\leftarrow$ For any two networks in *BN-tree* that are semi-isomorphic, discard one of them
    if $|$*BN-tree-UNIQUE*$| \geq N$ **then**
        return(*BN-tree-UNIQUE*)
        BREAK
    else
        M $\leftarrow$ M - 1
    end if
end while

isomorphic networks were not generated. Therefore, the process is repeated M times. However, if the total of at least N non-isomorphic networks are generated in any of the iterations, then, the algorithm stops.

3. Isomorphism for Boolean networks with cyclic attractors

Boolean networks have another important class of attractor structure: *cyclic attractors*. In this class, the attractor sets of the networks consist of l states that can transition to each other. If any of the l states has only self-referencing, then that state is a singleton attractor. This section provides the conditions to verify isomorphic BNs with cyclic attractors, up to gene relabelings. Figure 20 shows an example of a BN with cyclic attractors, where $l = 4$.

Clearly the Boolean networks with cyclic attractors, cannot be represented as *k-BN-tree*s. The general definition of the BNs, provided in section 1 of the Chapter II, states that the BNs have sets of attractors, where the attractors can be cyclic, singleton or a mixture of them. The Boolean networks with cyclic attractors are the general case and their state transition diagram can be represented as a directed graph. The definition of the partial order for *k-BN-tree*s, defined in section 1 of this chapter, easily extends to the case of Boolean networks with cyclic attractors. Similarly, the notion of semi-isomorphism can be extended to any case of BNs. Attractors with different lengths of cycles allows for stronger sufficient condition for eliminating semi-isomorphism in the general case. This section provides an algorithm 7 that eliminates semi-isomorphic Boolean networks with cyclic attractors, while generating networks with algorithm 1 in [14].

The algorithm 7 takes a conservative approach by keeping only one network with a given cycle length and partial order of basins. Similar to algorithm 6, nN networks are generated to ensure that the process results in at least N non-isomorphic networks.

The algorithm stops if: 1- in any of iterations, at least N non-isomorphic networks are generated; 2- the algorithm is repeated for a total of M iterations.

The two algorithms presented in this chapter, are used in the simulation studies of Chapters III and IV. These algorithms ensured that only non-isomorphic BNs are used and thus, the sets of synthetically generated networks are not biased.

Attractor	17	36	42	121
Cycle Length	2	2	2	2

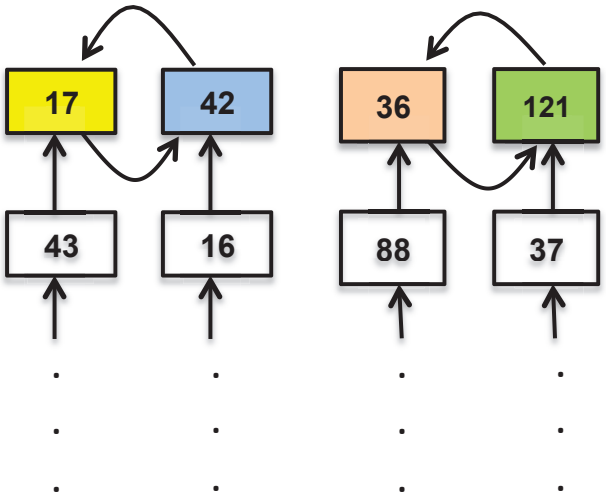

Fig. 20. Boolean network with 4 cyclic attractors. There exists 7 genes and 128 states; therefore, the truth table consists of 128 rows. Four states {17, 36, 42, 121} are the attractors. The cycle length for each attractor is 2 for all 4 attractors.

Algorithm 7 Discarding isomorphic Boolean networks with cyclic attractors - Part 1

Input N: total number of needed non-isomorphic BNs
Input M: total number of iterations
Input l: total number of attractor States for each BN
Input n
while M $\neq 0$ **do**
    Generate nN sets of numbers, where each set consists of exactly l non-repeating numbers
    BN-Cyclic-Attractors $\leftarrow$ Generate nN networks by the algorithm outlined in [14], using the sets with exactly l unique numbers as prescribed cyclic attractors
    for all the $BN(i) \in$ *BN-Cyclic-Attractors* **do**
        $CycleLength(i) \leftarrow$ Length of attractor cycles
        $BasinStructure(i) \leftarrow$ The number of States within the basin of each attractor
        if The first appearance of $CycleLength(i) == True$ **then**
            Add $BN(i)$ to the *BN-Cyclic-Attractors-UNIQUE* set
        else
            if The first appearance of $BasinStructure(i) == True$ **then**
                Add $BN(i)$ to the *BN-Cyclic-Attractors-UNIQUE* set
            else
                Discard the $BN(i)$
            end if
        end if
    end for

Algorithm 8 Discarding isomorphic Boolean networks with cyclic attractors - Part 2

```
21:     if |BN-Cyclic-Attractors-UNIQUE| ≥ N then
22:         return(BN-Cyclic-Attractors-UNIQUE)
23:         BREAK
24:     else
25:         M ← M -1
26:     end if
27: end while
```

CHAPTER VI

AN INFERENCE METHOD WITH AN INTERVENTION INTENT

A. Introduction

One of the important goals of translational genomics is to model genomic regulatory networks to alter the time evolution of their gene activity profile toward desired states. Synthetically generated networks are widely used to model GRNs and study the effects of available control mechanisms, [14] and Chapter V of this dissertation. Another approach aims to *infer* gene regulatory networks using the measurements from patients' samples. This chapter introduces a new method that considers the ultimate task of controlling, while inferring a Boolean network. This new algorithm is called: *CoD-Control-Embedded-inference (CoD-CE-Inference)*.

Previously, the inference of GRNs, from the experimental microarray measurements, had been addressed by the well-known *seed-growing algorithm* [15]. The seed-growing method starts by a set called *seed*, consisting of one or more genes. The functionality/regulatory role of the seed genes is known or be simply of interest. The objective of the algorithm is to grow subnetworks from the seed genes, that are strongly connected and have weak impact from the rest of the genes. This method remains in the context of graph theory and delivers directed graph, representing the topology of the GRN. A brief review of this method is provided in Chapter II, section 5.

The *CoD-CE-Inference* is an inference procedure that generates networks from gene expression measurements. It takes the a binarized input dataset in order to infer a Boolean network with strong inter-gene connections. The *CoD-CE-Inference* shares few similarities to the seed-growing method, such as using the seed gene and

adjoining genes sequentially. Nonetheless, it pursues a different goal and delivers only one Boolean network. The main difference between *CoD-CE-Inference* and seed-growing method is in their objective: while *CoD-CE-Inference* infers a BN with the ultimate goal of controlling its long-run behavior via applying control policies, the seed-growing algorithm is concerned with growing subnetworks in accordance with the *principles of autonomy* [15]. The *CoD-CE-Inference* utilizes the CoD for measuring the strength of gene connections; seed-growing algorithm uses CoD and *Influence of genes*, Chapter II section 5, to measure the strength of connections.

The *CoD-CE-Inference* initiates by setting one gene as the seed, called *target* (T) gene. This gene could be selected by a biologist/physician or could be a well-known gene related to the phenotype of interest. Currently, the algorithm starts with one gene as the target, but can be easily generalized to work with more genes. The ultimate goal of controlling the inferred BN plays an important role in all of the steps of the algorithm. It is known that for beneficially controlling the behavior of the T, the expression level of another gene called *control* (C), needs to be altered. The algorithm start growing a network by adding 2 genes that have strong CoD-measured gene connection to the target gene. Genes that are tightly connected to T measured by CoD, or to one of the other genes inside the growing-network are added sequentially. This criterion provides an opportunity for selecting a good candidate control gene, in the cases that this gene is not provided by prior knowledge. However, in practice, the control gene is chosen by a biologist/domain expert. The truth table of the inferred BN is constructed, based on the genes within the growing-network. Section B of this chapter describes the algorithm in more details.

As an important property, the *CoD-CE-Inference re-wires* the network after adding a new gene. *Re-wiring* means that the genes predicting any given gene, can be changed after adding a new gene to the network. Therefore, adding each gene can

affect the dynamics of the network and its corresponding truth table.

In summary, the seed-growing algorithm and the *CoD-CE-Inference* have four major differences:

1. *CoD-CE-Inference* aims to generate a network with the purpose of control, whereas the seed-growing algorithm is concerned only with the topology of the inferred subnetworks

2. *CoD-CE-Inference* generates the truth table of the GRN, however, the current implementation of the seed-growing method does not generate the truth table

3. After adding any new gene, *CoD-CE-Inference* re-wires the network, as opposed to the seed-growing method that keeps the structure of currently inferred subnetworks untouched

4. *CoD-CE-Inference* infers a unique BN, while the seed-growing algorithm generates many networks starting from the seed gene(s)

B. Proposed inference method

The *CoD-CE-Inference* algorithm grows a network by adjoining genes that are strong predictors of the genes within the already inferred network. The genes within the inferred network, including the C gene, are strong predictors of the T, therefore, altering the behavior of the network toward more desirable states can be easily achieved by using the available optimal/greedy control policy methods. This section provides the details of the new algorithm.

The process initiates by selecting the T as the seed gene. For selecting the first pair of genes to join the T, the 2-gene combinations of all of the available genes within the input dataset, excluding the seed, are found. For each 2-gene combination, its

CoD for predicting the T is calculated. The combination that has the maximum CoD w.r.t. T is added to the growing-network. In the next step, the the truth table (TT) is constructed, using the conditional probability distribution (CPD) table of the winning 2-gene combination. Table XI shows the general form of a CPD table. In Chapter IV, section B, the triple with maximum CoD for predicting T is called *MAXCOD* and the their corresponding CPD table is called *MAXCPD*. Herein, the same terminology is used to refer ro the gene combination with maximum CoD for predicting T and its CPD table. However, two genes are used to predict T, and thus, the *MAXCPD* table has 4 rows. Also, the values in columns 3 and 4 are the average frequencies (proportions) of the binary values of T, conditioned on the values of *MAXCOD* genes.

Table XI. Conditional Probability Distribution (CPD) Table: the first two columns represent the binary combinations of the 2 predictor genes. The last two columns represents the proportions of the frequencies of the target, conditioned on predictor 1 and predictor 2.

	Predictor genes		T	
	Predictor 1	*Predictor 2*	0	1
row 1	0	0	P_{10}	P_{11}
row 2	0	1	P_{20}	P_{21}
row 3	1	0	P_{30}	P_{31}
row 4	1	1	P_{40}	P_{41}

During the inference procedure, the steady-state distribution of the network is not available. Therefore, for calculating the CoDs, the binarized gene expression values are used. Figure 21 shows the process of generating the CPD tables from

the binarized gene expressions. A CPD table is generated, using a gene expression dataset that is binarized, normalized and filtered, using the methods in [37], where:

1. The first 2 columns of the table represent the binary values of the two genes, predicting the target gene.

2. For completing columns 3 and 4, the frequencies of the binary values of the T across all samples, conditioned on the 2 predictor genes, are averaged. Therefore, the values in these two columns represent the proportion of the T values, conditioned on predictor 1 and predictor 2. If a binary combination of the 2 predictor genes is not observed in the data, then, its corresponding values in columns 3 and 4 are set to zero.

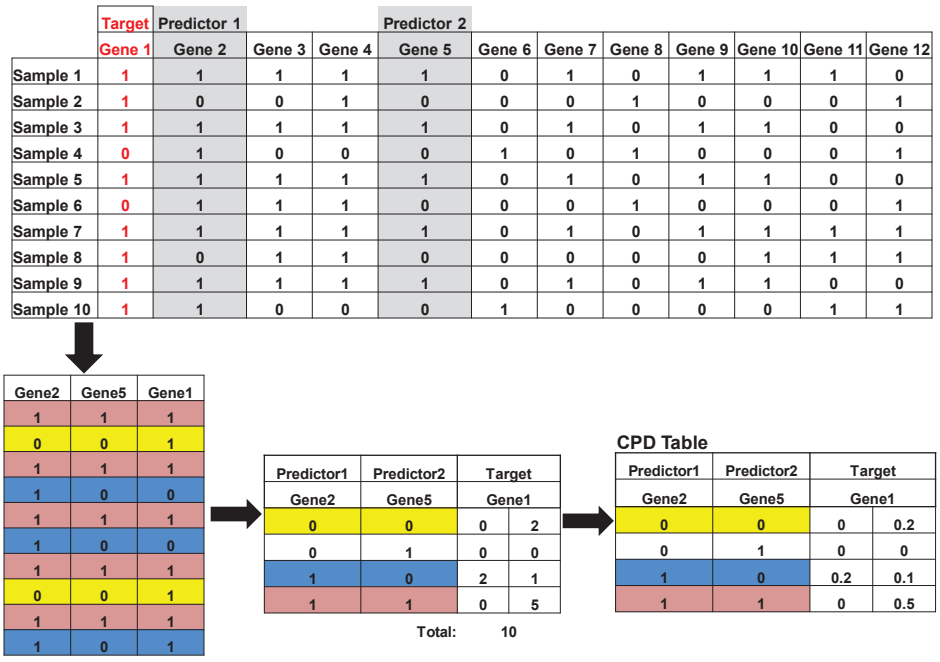

Fig. 21. Generating CPD tables from binarized microarry measurements

After adding the first 3 genes, the algorithm adds genes one by one to the growing-network. In each step, to select a gene to be added, one of the genes, within the growing-network, is set as the *temporary target* gene. Then, all possible 2-gene combinations are made such that one gene is from *inside* the growing-network and one gene is from *outside*. The goal of this step is to combine a gene from outside, with a gene from inside, to find a gene that can help in maintaining the strong CoD-measured connections among genes within the growing-network. This method for selecting a new gene differs the *CoD-CE-Inference* algorithm from the seed-growing algorithm. As provided in Chapter II, section 5, seed-growing algorithm does not consider gene combinations, instead it selects one gene form outside the growing subnetwork, where it maximize the equation 2.14. The *CoD-CE-Inference* algorithm takes another approach: the gene to be added, is from outside the growing-network, where it has maximum CoD for predicting the *temporary target* gene, when it is combined with a gene inside the growing-network. The gene, from the not added genes, in the winning 2-gene combination will be added to the network. The following example explains how the functions for the newly added gene will be completed. The algorithm 11 outlines the procedure for filling up the TT.

Example 1: This example fills up a column of the truth table for a 7-gene Boolean network with 128 states. The process goes through all the states of the network. For deciding which row of the CPD table to use, one needs to look at the values of the predictor genes in the state that is being considered. Without loss of generality, assume that x_2 and x_3 are the predictor genes for the T, which is x_1. Considering the state $\mathbf{s} = 0010000$, one can see that $x_2 = 0$ and $x_3 = 1$. This binary value corresponds to the row 2 of the CPD table. Therefore, for filling up the a

column of truth table for x_1, the row 2 of the CPD table is used. If $P_{20} < P_{21}$, then, the values of the T for the functions within the TT are set to be 0. If $P_{20} > P_{21}$, then, the T values are set to be 1. If none of these cases happen, it means $P_{20} = P_{21}$, therefore, on can choose the value uniformly randomly.

After choosing the gene to be added to the growing-network, the *CoD-CE-Inference* re-wires the network. For predicting each gene within the growing network, all of the other possible 2-gene combinations of the genes within the growing-network are examined. Two genes that predict this gene with maximum CoD are selected as its predictors. The CPDs and the strongest predictors for each gene are used to fill up the TT, using the process explained by example 1 and algorithm 11. After this step, the inferred network and its TT are completed. The procedure is represented in algorithm 9. The algorithm stops when a pre-specified number of genes are added to the network.

C. Discussion

This section uses the proposed *CoD-CE-Inference* method to infer Boolean networks from gene expression measurements. The data is from a gastrointestinal cancer study, where a 2-gene classifier is devised for accurately distinguishing the two types of the disease [1]. To infer the BN, the microarray data were normalized, filtered and binarized using methods from [37]. Two different networks were inferred, using two seed genes. The Boolean networks have 17 genes, where the dimension of the state transition matrix is $2^{17} \times 2^{17}$. Due to the limited computational power, more genes could not be added to the networks. However, there is no limit on the number of genes that *CoD-CE-Inference* can adjoin to a network. The following two sections provide more details about the two BNs that are inferred by *CoD-CE-Inference*.

Algorithm 9 CoD-CE-Inference, initiation and adjoining genes - Part 1

STEP 1*: Initiation by Adding First 3 Genes*

Input T

Input *Max-Number-of-Genes*

$\{POTENTIAL\text{-}GENES\} \leftarrow$ all the genes, except T

Find the 2-gene combination $\in$ *POTENTIAL-GENES* with maximum CoD For predicting T: *MAXCOD*

Call the *'Fill-Out-TT'* function using T and *MAXCOD*

$\{ADDED\text{-}GENES\} \leftarrow \{T, MAXCOD\}$

Remove *MAXCOD* from *POTENTIAL-GENES*

TOTAL-GENES $\leftarrow 3$

STEP 2*: Adding More Genes*

while *TOTAL-GENES* $<$ *Max-Number-of-Genes* **do**

    for all the genes $x_i \in$ *ADDED-GENES* **do**

        Set x_i as the *TEMPORARY-T*

        Find the set *MIXTURE*: includes all 2-gene combinations y_1y_2, where $y_1 \in$ *POTENTIAL-GENES* and $y_2 \in$ *ADDED-GENES*

        for all the combinations $\in$ *MIXTURE* **do**

            $\Theta_{iy_1y_2} \leftarrow$ CoD For predicting *TEMPORARY-T*

        end for

    end for

    Find maximum $\Theta_{iy_1y_2}$ and its corresponding 2-gene combinations y_1y_2

    Call the *'Fill-Out-TT'* For y_1

    Add y_1 to: *ADDED-GENES*

    Remove y_1 from: *POTENTIAL-GENES*

Algorithm 10 CoD-CE-Inference, initiation and adjoining genes - Part 2

23: *TOTAL-GENES* $\leftarrow$ *TOTAL-GENES* $+ 1$

24: **for** each gene within $\in$ *ADDED-GENES* **do**

25: Find its *MAXCOD*, using the 2-gene combinations $\in$ *ADDED-GENES*

26: Call the *'Fill-Out-TT'* function to complete TT

27: **end for**

28: **end while**

29: return (*ADDED-GENES*)

D. Gastrointestinal cancer network, OBSCN as the seed gene

The first network, presented in this section, uses the gene *OBSCN* as the seed gene. This gene is one of two genes composing the best classifier in [1]. The network is grown by adding genes that have strong connectivity to OBSCN, as measured by the CoD. The seed gene, OBSCN, is set as the target gene and the second gene added to the network, GREM2, is set as the control gene. Unless there is a biologically known relation between a target gene and a particular phenotype, as in the case of WNT5A and metastatic competence in melanoma, there is no standard way to select a target and control gene pair; however, it is reasonable to expect that the best 1-gene classifier, OBSCN, that discriminates between two types of cancer can also be a potential target for a possible therapeutic intervention. GERM2 has the strongest CoD connection to this gene and thus, could be viewed as a good candidate for a control gene. At each iterative step, a gene from outside the growing-network combined with a gene inside the network. The outside gene from the combination that has the strongest connectivity, measured by the CoD, to one of the genes from the current network is added to the network. Then, the network is re-wired taking

Algorithm 11 CoD-CE-Inference, Filling out the Truth Table, Fill-Out-TT

```
Input CPD table
Input TARGET-GENE
Input TOTAL-STATES as the total number of the States
Input Predictor1-COLUMN
Input Predictor2-COLUMN
TOTAL-ROWS-CPD ← 4
for i ≤ TOTAL-ROWS-CPD do
    Predictor1-VALUE ← CPD(i,Column1)
    Predictor2-VALUE ← CPD(i,Column2)
     for j ≤ TOTAL-STATES do
        if StateS(j,Predictor1-COLUMN) = Predictor1-VALUE &&
StateS(j,Predictor2-COLUMN) = Predictor2-VALUE then
            if CPD(i,P_i0) < CPD(i,P_i1) then
                FUNCTIONS(j,TARGET-GENE) ← 0
            else if CPD(i,P_i0) > CPD(i,P_i1) then
                FUNCTIONS(j,TARGET-GENE) ← 1
            else
                FUNCTIONS(j,TARGET-GENE) ← Randomly choose 0 or 1
            end if
        end if
     end for
end for
return (FUNCTIONS)
```

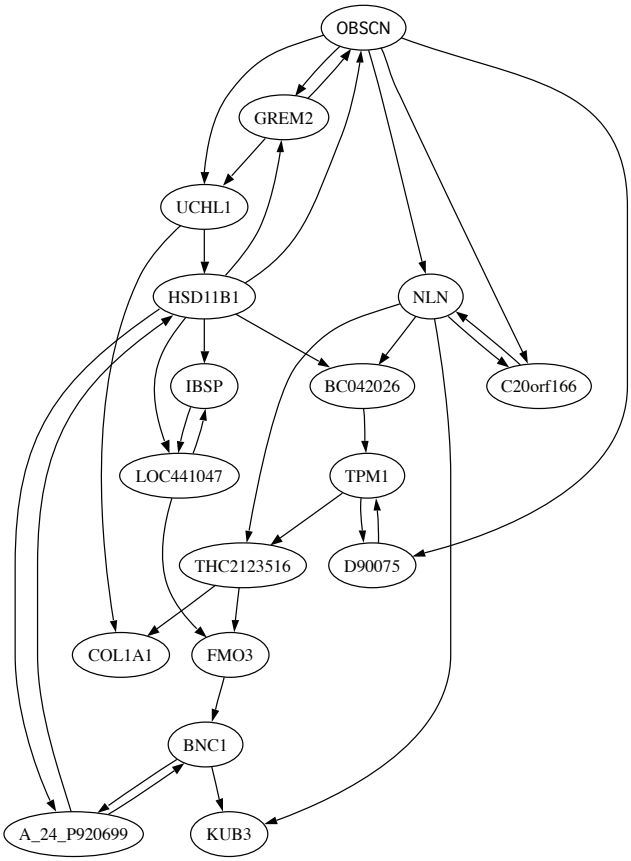

Fig. 22. 17-gene Gastrointestinal Cancer Network, with OBSCN as the seed gene, [30], [31]

into account that a new gene in the network can change the way genes influence each other. The 17-gene network includes the following genes: $OBSCN$, $GREM2$, $HSD11B1$, $UCHL1$, $A\_24\_P920699$, $BNC1$, $FMO3$, $LOC441047$, $THC2123516$, NLN, $COL1A1$, $IBSP$, $C20orf166$, $KUB3$, $TPM1$, $D90075$ and $BC042026$. Figure 22 shows this network.

1. Applying CoD-Reduce and CoD-CP

To demonstrate the performance of the *CoD-CP* algorithm, the 17-gene inferred network with OBSCN as the seed gene is used. The *CoD-CP* can be directly designed

on this large network, but MFPT-CP and SSD-CP cannot be derived. Therefore, the *CoD-Reduce* is used to delete genes consecutively until only 10 genes are left in the network: $OBSCN$, $GREM2$, $HSD11B1$, $BNC1$, $LOC441047$, NLN, $C20orf166$, $KUB3$, $D90075$ and $BC042026$. At that point it is possible to design the MFPT-CP and SSD-CP policies, after which they are induced back to the original 17-gene network. The resulting performance comparison of the *CoD-CP* policy with the induced MFPT-CP and SSD-CP policies is shown in Figure 23. The SSD shift for the CoD-CP is better than the shift for the induced MFPT-CP and SSD-CP policies, which are about the same. The perturbation probability used in this experiment is $p = 0.1$. Figure 24 displays the results for $p = 0.01$. It is important to point out that the small perturbation probability, $p = 0.01$, makes such a large network to be very deterministic. Thus, all of the three control policies produce significant shifts in the network SSD towards the desirable states. In addition, one can notice that the *CoD-CP* performs extremely well which can be attributed to the use of CoD to infer the network structure from data. This results illustrates the importance of the proper combination of network inference and control policy design methods.

E. Gastrointestinal cancer network, C9orf65 as the seed gene

For furthure demonstration of *the CoD-CE-Inference* performance, another large 17-gene network is generated using the gastrointestinal cancer data set from [1], and uses *C9orf65* gene as the seed gene. The total number of the genes in the final network is 17: $C9orf65$, $CXCL12$, $TK1$, $SOCS2$, $THC2168366$, $SEC61B$, $ENST00000361295$, $KCNH2$, $ACTB$, $RPS18$, $RPS13$, $THC2199344$, $SNX26$, $RPL26$, $SLC20A1$, $RPS11$, $THC2210612$, $THC2161967$, $IER2$ and $LAMP1$. Gene $CXCL12$ is set to be the control gene.

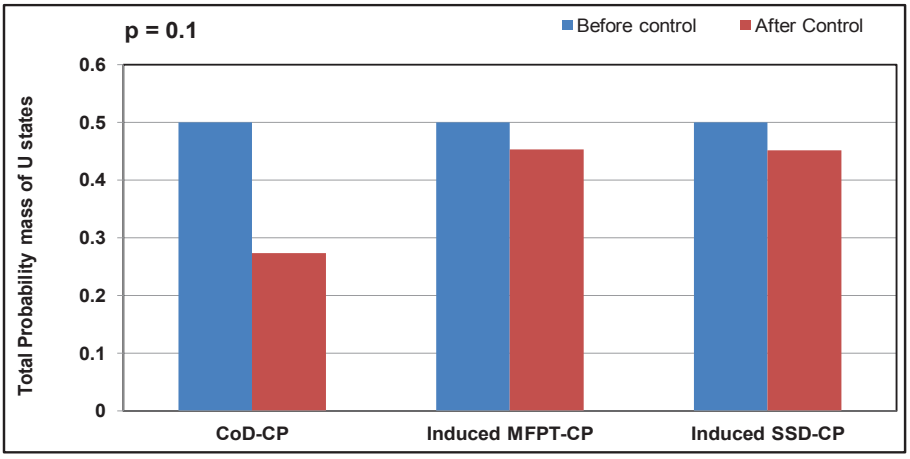

Fig. 23. Comparing the total SSD shift for the Undesirable states, before and after applying CoD-CP, Induced MFPT-CP and SSD-CP. The CoD-CP is designed on the 17-gene Gastrointestinal cancer network. The 17-gene network was reduced to 10 genes, the MFPT-CP and SSD-CP were designed for it and then these control policies induced back and applied on the original 17-gene network. The seed gene is OBSCN and p=0.1.

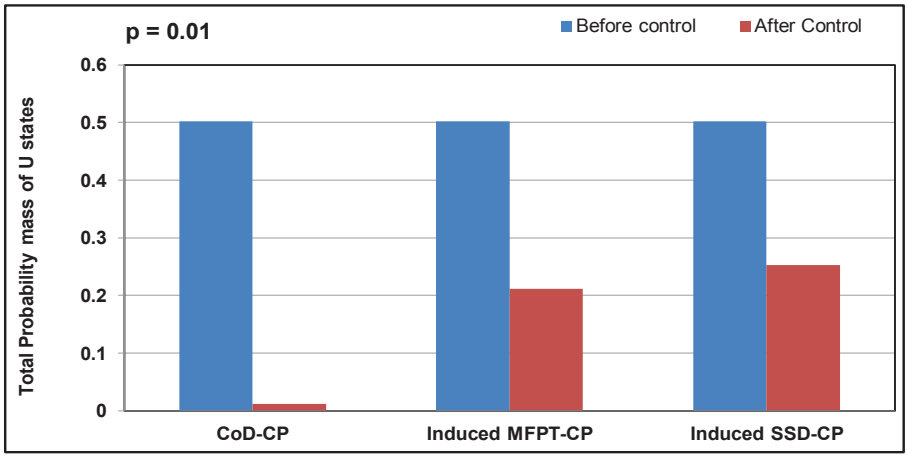

Fig. 24. Comparing the total SSD shift for the Undesirable states, before and after applying CoD-CP, Induced MFPT-CP and SSD-CP. The CoD-CP is designed on the 17-gene Gastrointestinal cancer network. The 17-gene network was reduced to 10 genes, the MFPT-CP and SSD-CP were designed for it and then these control policies induced back and applied on the original 17-gene network. The seed gene is OBSCN and p=0.01.

The network is very large and MFPT-CP cannot be directly designed for it. Therefore, initially the best gene for deletion is selected using *CoD-Reduce* and the network is reduced by deleting one gene, then *CoD-Reduce* is applied consecutively to reduce the network down to 10 genes: $C9orf65$, $CXCL12$, $RPS18$, $RPS13$, $THC2199344$, $SNX26$, $RPL26$, $SLC20A1$, $RPS11$ and $THC2210612$. After reducing the original network down to 10 genes, the MFPT control policy for the reduced network is designed and then induced back on the original 17-gene network. Table XII shows the total probability mass of the desirable and undesirable states before and after applying the induced control policy. As the table illustrates, there is about 13% shift in the steady-state distribution of the network toward more desirable states.

Table XII. SSD shift toward the Desirable states in Gastrointestinal Cancer Network, with C9orf65 as seed

Total Probability mass of Desirable states, before control	0.594127833
Total Probability mass of Desirable states, after control	0.722673833
Total Probability mass of Undesirable states, before control	0.405872167
Total Probability mass of Undesirable states, after control	0.277326167

REFERENCES

[1] N. D. Price, J. Trent, A. K. El-Naggar, D. Cogdell, E. Taylor, K. K. Hunt, R. E. Pollock, L. Hood, I. Shmulevich, and W. Zhang, "Highly accurate two-gene classifier for differentiating gastrointestinal stromal tumors and leiomyosarcomas," *Proceedings of the National Academy of Science*, vol. 104, no. 9, pp. 3414–3419, 2007.

[2] I. Shmulevich and E. R. Dougherty, *Probabilistic Boolean Networks: The Modeling and Control of Gene Regulatory Networks*. New York: SIAM Press, 2010.

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

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

[5] A. Datta, R. Pal, A. Choudhary, and E. Dougherty, "Control approaches for probabilistic gene regulatory networks," *IEEE Signal Processing Magazine*, vol. 24, no. 1, pp. 54 – 63, 2007.

[6] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA: Athena Scientific, 2005.

[7] T. Akutsu, M. Hayashida, W.-K. Ching, and M. K. Ng, "Control of boolean networks: Hardness results and algorithms for the tree structured nqtworks," *Journal of Theoretical Biology*, vol. 244, no. 4, pp. 670 – 677, 2007.

[8] G. Vahedi, B. Faryabi, J.-F. Chamberland, A. Data, and E. Dougherty, "Intervention in gene regulatory networks via a stationary mean-first-passage-time control policy," *IEEE Transactions on Biomedical Engineering*, pp. 2319–2331, 2008.

[9] X. Qian, I. Ivanov, N. Ghaffari, and E. R. Dougherty, "Intervention in gene regulatory networks via greedy control policies based on long-run behavior," *BMC Systems Biology*, vol. 3, no. 61, 2009.

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

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

[12] I. Shmulevich, E. Dougherty, and W. Zhang, "From boolean to probabilistic boolean networks as models of genetic regulatory networks," *Proceedings of IEEE*, vol. 90, no. 11, pp. 1778–1792, 2002.

[13] M. Brun, E. Dougherty, and I. Shmulevich, "Steady-state probabilities for attractors in probabilistic boolean networks," *Signal Processing*, vol. 85, no. 10, pp. 1993 – 2013, 2005.

[14] R. Pal, I. Ivanov, A. Datta, and E. Dougherty, "Generating boolean networks with a prescribed attractor structure," *Bioinformatics*, vol. 54, no. 21, pp. 4021 – 4025, November 2005.

[15] R. Hashimoto, S. Kim, I. Shmulevich, W. Zhang, M. Bittner, and E. Dougherty, "A directed-graph algorithm to grow genetic regulatory subnetworks from seed

genes based on strength of connection," *Bioinformatics*, vol. 20, no. 8, pp. 1241–1247, 2004.

[16] J. Norris, *Markov Chains*. Cambridge University Press, 1998.

[17] S. Kim, E. R. Dougherty, M. Bittner, Y. Chen, K. Sivakumar, P. Meltzer, and J. M. Trent, "A general framework for the analysis of multivariate gene interaction via expression arrays," *Biomedical Optic*, vol. 5, no. 4, pp. 411–424, 2000.

[18] D. Martins, U. Braga-Neto, R. Hashimoto, M. Bittner, and E. R. Dougherty, "A general framework for the analysis of multivariate gene interaction via expression arrays," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 3, pp. 424–439, 2008.

[19] E. Dougherty, M. Brun, J. Trent, and M. L. Bittner, "A conditioning-based model of contextual regulation," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 6, no. 2, pp. 310–320, 2009.

[20] X. Quian and E. Dougherty, "Effect of function perturbation on the steady-state distribution of genetic regulatory networks: optimal structural intervention," *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 4966–4975, 2008.

[21] P. Schweitzer, "Perturbation theory and finite markov chains," *Journal of Applied Probability*, vol. 5, pp. 401 – 413, 1968.

[22] J. Hunter, "Stationary distributions and mean first passage times of perturbed markov chains," *Linear Algebra and its Applications*, vol. 410, pp. 217 – 243, 2005.

[23] M. Ng, S.-Q. Zhang, W. Ching, and T. Akutsu, "A control model for markovian genetic regulatory networks," *Transactions on Computional Systems Biology*, pp.

36–48, 2006.

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

[25] I. Ivanov and E. Dougherty, "Reduction mappings between probabilistic boolean networks," *EURASIP JASP*, vol. 1, no. 1, pp. 125 – 131, 2004.

[26] I. Shmulevich and E. R. Dougherty, *Genomic Signal Processing*. Princeton: Princeton University Press, 2007.

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

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

[29] I. Ivanov, P. Simeonov, N. Ghaffari, X. Quian, and E. Dougherty, "Selection policy induced reduction mappings for boolean networks," *IEEE Transactions on Signal Processing*, vol. 58, no. 9, pp. 4871–4882, September 2010.

[30] N. Ghaffari, I. Ivanov, X. Qian, and E. Dougherty, "A cod-based reduction algorithm for designing stationary control policies on boolean networks," *Bioinformatics*, vol. 26, no. 12, pp. 1556–1563, 2010.

[31] X. Qian, N. Ghaffari, I. Ivanov, and E. Dougherty, "State reduction for network intervention with probabilistic boolean networks," *Bioinformatics*, vol. 26, no. 24, pp. 3098–3104, 2010.

[32] N. Ghaffari, I. Ivanov, X. Quian, and E. Dougherty, "A cod-based stationary control policy for intervening in large gene regulatory networks," *BMC Bioinformatics*, vol. 12, no. S10, 2011.

[33] S. Kim, H. Li, E. R. Dougherty, N. Chao, M. L. Bittner, and E. B. Suh, "Can markov chain models mimic biological regulation," *Biological Systems*, vol. 10, pp. 447–458, 2002.

[34] R. Pal, A. Datta, and E. Dougherty, "Robust intervention in probabilistic boolean networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 3, pp. 1280 – 1294, 2008.

[35] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2001.

[36] A. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, 1974.

[37] I. Shmulevich and W. Zhang, "Binary analysis and optimization-based normalization of gene expression data," *Bioinformatics*, vol. 18, no. 4, pp. 555 – 565, 2002.

[38] A. Naldi, E. Remy, D. Thieffry, and C. Chaouiya, "A reduction method for logical regulatory graphs preserving essential dynamical properties," *Lecture Notes in Bioinformatics*, vol. 5688, pp. 266–80, 2009.

[39] H. Conzelmann, J. Saez-Rodriguez, T. Sauter, E. Bullinger, F. Allgower, and E. Gilles, "Reduction of mathematical models of signal transduction networks: simulation-based approach applied to egf receptor signalling," *Systems Biology*, vol. 1, pp. 159–169, 2001.

[40] R. Wang, T. Zhou, Z. Jing, and L. Chen, "Modelling periodic oscillation of biological systems with multiple timescale networks," *IET Systems Biology Journal*, 2004.

[41] P. Indic, K. Gurdziel, R. Kronauer, and E. Klerman, "Development of a two-dimension manifold to represent high dimension mathematical models of the intracellular mammalian circadian clock," *Journal of Biological Rhythms*, vol. 21, pp. 222–232, 2006.

[42] N. Borisov, N. Markevich, J. Hoek, and B. Kholodenko, "Signaling through receptors and scaffolds: independent interactions reduce combinatorial complexity," *Biophysical Journal*, vol. 89, pp. 951–966, 2005.

[43] H. Conzelmann, J. Saez-Rodriguez, T. Sauter, B. Kholodenko, N, and E. Gilles, "A domain-oriented approach to the reduction of combinatorial complexity in signal transduction networks," *BMC Bioinformatics*, vol. 7, no. 34, pp. 159–169, 2006.

[44] B. L. Clarke, "General method for simplifying chemical networks while preserving overall stoichiometry in reduced mechanisms," *Journal of Chemical Physics*, vol. 97, pp. 4066–4071, 1992.

[45] K. Ball, T. Kurtz, L. Popovic, and G. Rempala, "Asymptotic analysis of multiscale approximations to reaction networks," *Annals of Applied Probability*, vol. 16, no. 4, pp. 1925–1961, 2006.

[46] O. Radulescu, A. Gorban, A. Zinovyev, and A. Lilienbaum, "Robust simplifications of multiscale biochemical networks," *BMC Systems Biology*, vol. 2, no. 86, 2008.

[47] L. Hartwell, J. Hopfield, S. Leibler, and A. Murray, "From molecular to modular cell biology," *Nature*, vol. 402, pp. 6761–supp, 1999.

[48] J. Saez-Rodriguez, A. Kremling, and E. Gilles, "Dissecting the puzzle of life: modularization of signal transduction networks," *Computers and Chemical Engineering*, vol. 29, pp. 619–629, 2005.

[49] A. Gorban, N. Kazantzis, I. Kevrekidis, H. ttinger, and C. Theodoropoulos, *Model Reduction and Coarse-Graining Approaches for Multiscale Phenomena*. Springer, 2006.

APPENDIX A

FLOWCHART OF SELECTING TARGET-CONTROL GENE WITH DIRECT CONNECTION

This appendix shows graphically the steps that are needed for selecting the *Target-Control* pair with *direct connection*, according to the truth table of the network. The requirement of the *direct connection* is relaxed in further experiments to be either *direct connection* or a *path* between target and control genes.

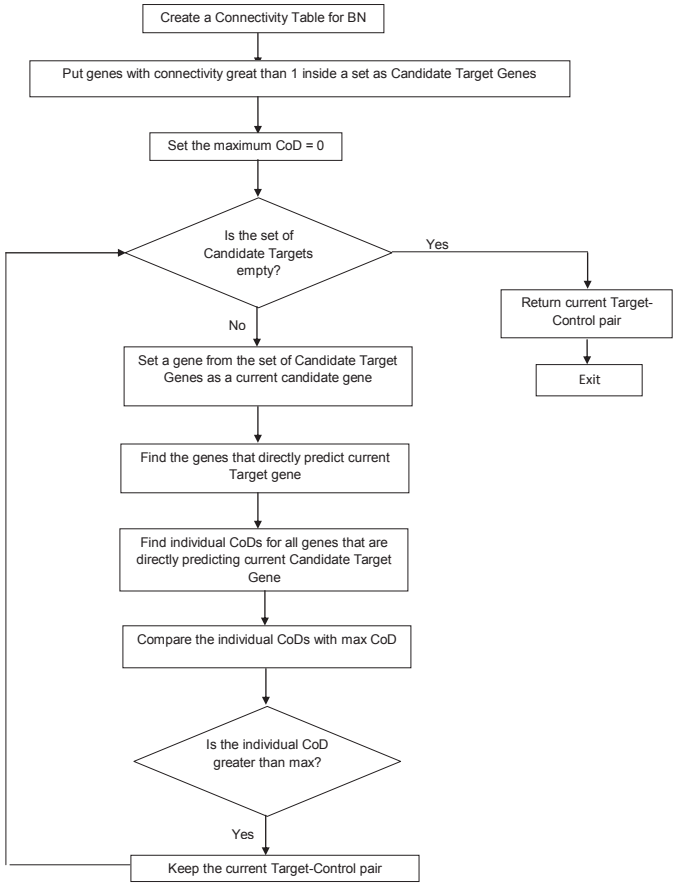

Fig. 25. Procedure for selecting Target-Control pair with direct connection

APPENDIX B

REDUCTION IN OTHER CONTEXTS

Complexity reduction has been considered for other classes of models, including discrete network models like Boolean networks [25, 27] or logical regulatory graphs [38], and continuous biochemical networks [39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49]. All past efforts focus on reducing the complexity while preserving network dynamics, either by maintaining the attractor structures as in discrete mathematical frameworks [27, 38] or partitioning large systems into smaller subsystems to enable better analysis and understanding for continuous network models as in [39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49]. Although the literature mainly focuses on reduction of biochemical networks in a continuous simulation framework [39, 43, 44, 46, 48, 49] while our work models gene regulatory networks in a discrete mathematical framework, the idea of partitioning large systems into multi-scale or hierarchical small subsystems could be an interesting future research direction on model reduction for Boolean-network-based gene regulatory networks if one can suitably abstract the relationships within subnetworks.

The following three reviewed papers, consider the networks which are deterministic. In comparison to the proposed methods, they do not consider stochastic networks and therefore the steady-state distribution is not considered as a dynamical property to be preserved after reduction. More importantly, the present dissertation has the essential objective of deriving a reduction for preserving the performance of potential intervention, while these do not.

Naldi, et. al. [38] implements a reduction algorithm to reduce the complexity of a logical regulatory graph by making one node "implicit" in the graph. By removing

one node, the paper proposes to connect the predictors and targets of the removed node and manipulate the regulatory functions to preserve the attractor structure as much as possible. The paper also provides the proofs that the reduction preserves the attractors in the original network but it can introduce spurious nontrivial attractors. In addition, state transition trajectories in the reduced model can be related to trajectories in the original model while it may lose important information regarding reachability properties. Due to these problems, it appears to be unclear whether there is good theoretical performance guarantee for the preservation of dynamical properties (attractor structures) for the iterative algorithm of removing more nodes from the graph. The proposed reduction is a kind of extension of the reduction mapping algorithm for Boolean networks in [27] to finer quantified logical regulatory graphs. In fact, both methods consider the preservation of attractor structures as important dynamical properties.

Conzelmann et. al. [39] consider the reduction of biochemical networks in a continuous simulation framework. The method first partitions large systems into "retroactive-free modules." It then studies each module by simulation and proposes to use a simpler linear system as a reduced model to approximate the I/O behavior of the original module. The partition of large systems into small modules in a hierarchical fashion is a popular and promising direction for model reduction; however, if the goal is to understand the original large system, then it is important to have accurate abstraction of relationships within partitioned modules, which is still on-going research.

Radulescu et. al. [46] and its references focus on understanding network dynamical behavior by finding small subsystems/modules to analyze their dynamical behavior. In this paper, the authors are specifically interested in "dominant subsystems," subsystems with different time scales (as they study continuous biochemical

networks), and identification of critical parameters in ODEs. They implement algorithms to reduce the complexity for both linear and nonlinear models. They also provide a review for model reduction of continuous biochemical networks, including trajectory-based techniques, singular perturbation techniques, and aggregation or lumping techniques. Generally, these techniques are designed for continuous models and are not applicable for discrete mathematical frameworks such as Boolean networks and logical regulatory graphs. However, it can be noted that, the idea of partitioning large systems into multi-scale or hierarchical small subsystems could be investigated further.

VITA

Noushin Ghaffari received her B.S. degree in software engineering from Tehran Central Azad University in 2002 and her M.S. degree in computer information systems from the University of Houston-Clear Lake, TX, in 2006. She finished her Ph.D. in computer engineering at Texas A&M University, College Station, TX in December 2011. Prior to joining Texas A&M University, she worked at Exagen Diagnostics, Inc. as a scientific software engineer.

Her research interests include genomic signal processing, systems biology, computational biology, and bioinformatics. She is a member of IEEE and International Society for Computational Biology (ISCB). She was a recipient of "Who's Who among American Universities and Colleges" in 2006 and the "Jesse H. and Mary Gibbs Jones Scholarship," University of Houston-Clear Lake, 2005-2006. She joined the bioinformatics team at AgriLife Genetics & Bioinformatics Services during the last year of her doctoral studies, starting October 2010.

She can be reached at "contact@noushinghaffari.com", or

c/o Dr. Edward R. Dougherty

Department of Electrical and Computer Engineering

Texas A&M University

214 Zachry Engineering Center,

TAMU 3128

College Station, Texas 77843-3128