INDUCTION AND CONTROL OF LARGE-SCALE GENE REGULATORY NETWORKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MEHMET TAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

JUNE 2009

Approval of the thesis:

INDUCTION AND CONTROL OF LARGE-SCALE GENE REGULATORY NETWORKS

submitted by **Mehmet Tan** in partial fullfillment of the requirements for the degree of **Doctor of Philosophy in Computer Engineering, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Müslim Bozyiğit
Head of Department, **Computer Engineering**

Prof. Dr. Faruk Polat
Supervisor, **Computer Engineering Dept., METU**

Prof. Dr. Reda Alhajj
Co-supervisor, **Computer Science Dept., Univ. of Calgary**

Examining Committee Members:

Prof. Dr. İsmail Hakkı Toroslu
Computer Engineering Dept., METU

Prof. Dr. Faruk Polat
Computer Engineering Dept., METU

Prof. Dr. Halil Altay Güvenir
Computer Engineering Dept., Bilkent Univ.

Prof. Dr. Kemal Leblebicioğlu
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Volkan Atalay
Computer Engineering Dept., METU

Date: 9/6/2009

ABSTRACT

INDUCTION AND CONTROL OF LARGE-SCALE GENE REGULATORY NETWORKS

Tan, Mehmet
Ph.D., Department of Computer Engineering
Supervisor: Prof. Dr. Faruk Polat
Co-Supervisor: Prof. Dr. Reda Alhajj

June 2009, 99 pages

Gene regulatory networks model the interactions within the cell and thus it is essential to understand their structure and to develop some control mechanisms that could effectively deal with them. This dissertation tackles these two aspects. To handle the first problem, a new constraint-based modeling algorithm is proposed that can both increase the quality of the output and decrease the computational requirements for learning the structure of gene regulatory networks by integrating multiple biological data types and applying a special method for dense nodes in the network. Constraint-based structure learning algorithms generally perform well on sparse graphs and it is true that sparsity is not uncommon. However, some domains like gene regulatory networks are characterized by the possibility of having some dense regions in the underlying graph and the proposed algorithm is capable of dealing with this issue. The algorithm is based on a well-known structure learning algorithm called the PC algorithm, and extends it in multiple aspects. Once a network exists, we could address the second problem, namely control of the regulatory network for various applications where the curse of dimensionality is the main issue. It is possible that hundreds of genes may regulate one biological activity in an organism and this implies a huge state space even in the case of Boolean models. The thesis proposes effective methods to find control policies for large-scale networks. The modeling and control algorithms proposed in this dissertation have been evaluated on both synthetic and real data sets. The test results demonstrate the efficiency and effectiveness of the proposed approaches.

Keywords: gene regulatory networks, induction, control

ÖZ

BÜYÜK-ÖLÇEKLİ GEN DÜZENLEYİCİ AĞLARIN MODELLENMESİ VE KONTROLÜ

Tan, Mehmet
Doktora, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Faruk Polat
Ortak Tez Yöneticisi: Prof. Dr. Reda Alhajj

Haziran 2009, 99 sayfa

Gen düzenleyici ağlar hücre içindeki etkileşimleri modellediğinden, yapılarını anlamak ve onları verimli bir şekilde kontrol edebilen mekanizmalar geliştirmek çok önemlidir. Bu tez, bu iki yönü ele almaktadır. İlk problemde, gen düzenleyici ağların yapısını öğrenmek için, çıktı kalitesini artırırken berimsel gereksinimleri azaltan, birden fazla biyolojik veri tipini birlikte kullanan ve yoğun düğümler için özel bir yöntem uygulayan yeni bir kısıt-tabanlı modelleme algoritması önerilmektedir. Kısıt-tabanlı yapı öğrenme algoritmaları, seyrek çizgeler için iyi performans gösterirler ve seyreklik de nadir görülen bir durum değildir. Bununla beraber, gen düzenleyici ağlar gibi bazı alanlarda, yoğun bölgeler içeren çizgelere rastlanabilir ve önerilen algoritma bu durumla başa çıkabilir. Algoritma, iyi bilinen bir yapı öğrenme algoritması olan PC algoritması tabanlıdır ve onu birden fazla yönde geliştirmektedir. Elimizde bir ağ olduğunda ise, ikinci problem karşımıza çıkacaktır; gen düzenleyici ağların, çeşitli uygulamalar için, temel mesele ölçeklenebilirlik olmak üzere kontrolü. Bir organizmada yüzlerce genin tek bir biyolojik aktiviteyi düzenlemede rol alması mümkündür ve Boolean modellerde bile bu, muazzam büyüklükte bir durum uzayına karşılık gelir. Bu tez, büyük-ölçekli ağlara kontrol planları bulmak için verimli yöntemler önermektedir. Bu tezde önerilen modelleme ve kontrol algoritmaları hem sentetik hem de gerçek veri kümelerinde test edilmiştir. Test sonuçları, önerilen yaklaşımların etkin ve verimli olduklarını göstermektedir.

Anahtar Kelimeler: gen düzenleyici ağlar, modelleme, kontrol

ACKNOWLEDGMENTS

I have to start by thanking my wife, love, best friend Oya Ünsal Tan for making all these years of graduate studying bearable. I am lucky to have a spouse who is also a graduate student and who never let me forget that I was loved. It is hard for me to express my deep gratitudes for her support and understanding.

Next, I must thank my parents, Kezban and Mustafa Tan for their endless support and encouragement. They helped me at every step of this study in countless ways from cooking us fantastic food, to taking care of my other half while I was away.

My advisors, Faruk Polat and Reda Alhajj have been very valuable for me. I owe them much for helping me learn that working on the intersection of genes and computers could be fun. Their experience and wisdom definitely improved my skills in doing research. I have to also state how grateful I am to Reda Alhajj for his efforts in helping me live in Calgary without any problems other than being far from my home.

I am also grateful to my advisory committe members Rengül Çetin-Atalay and Volkan Atalay for their constructive comments in improving the research in this thesis.

I also wish to thank all people I met in my department. It has been a grate pleasure for me to work with them, I hope to keep in touch with all of them through the rest of my life.

The research in this thesis was supported by TÜBİTAK-BİDEB national PhD scholarship (2211) and TÜBİTAK-BİDEB PhD scholarship for research abroad (2214).

To my love and my family

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

TABLES

LIST OF ABBREVIATIONS

ADD	Algebraic Decision Diagram
APRICODD	Approximate Policy Construction using Decision Diagrams
BN	Bayesian Network
BoN	Boolean Network
cDNA	Complementary DNA
ChIP	Chromatin Immunoprecipitation
CLL	Chronic Lymphocytic Leukemia
COD	Coefficient of Determination
CPD	Conditional Probability Distribution
CPDAG	Complete Partially Directed Acyclic Graph
DAG	Directed Acyclic Graph
DBN	Dynamic Bayesian Network
DNA	Deoxyribo Nucleic Acid
DNF	Disjunctive Normal Form
EI	Edge Influence
FMDP	Factored Markov Decision Problem
FN	False Negative
FP	False Positive
GA	Genetic Algorithm
GGM	Graphical Gaussian Modelling
GO	Gene Ontology
GRN	Gene Regulatory Network
gss	Greedy Separator Search
IS	Influence Score
KEGG	Kyoto Encyclopedia of Genes and Genomes
KNN	K-nearest Neighbor
MC	Markov Chain
MCMC	Markov Chain Monte Carlo
MDP	Markov Decision Problem
miRNA	MicroRNA
mRNA	Messenger RNA
PBN	Probabilistic Boolean Network
PCR	Polymerase Chain Reaction
PD	Partially Dense
PDAG	Partially Directed Acyclic Graph
PPI	Protein-protein interaction
PSO	Particle Swarm Optimization
riRNA	Ribosomal RNA
RNA	Ribo Nucleic Acid
SHD	Structural Hamming Distance
snRNA	Small Nuclear RNA
SPUDD	Stochastic Planning using Decision Diagrams
TF	Transcription Factor
TN	True Negative
TP	True Positive
tRNA	Transfer RNA

CHAPTER 1

INTRODUCTION

1.1 Problem Definition and Motivation

Protein synthesis is one of the most essential functions taking place within a cell. A large number of different proteins are produced and consumed inside the cells of living organisms. This is accomplished by the help of the protein encoders in the cell: *the genes.* Number of genes varies between organisms; thought to be between 20000 and 25000 in humans.

Gene expression is the name given to the process of decoding a gene into a protein, and gene expression level can be defined as the amount of protein produced from a certain gene at a given time. Since the amount of a protein in the cell may change the dynamics of the cell, proteins have to be produced in an organized manner. This organization is performed by the genes themselves; some proteins called transcription factors bind to some special regions on genes and initiate or accelerate gene expression. Also there are some protein-protein interactions in the cell that effect gene expression. All these interactions constitute a complex network that is called a *Gene Regulatory Network* (GRN).

There are two important questions about gene expression regulation in the cell:

1. How can we deduce the relationships between genes in the cell? (i.e., which gene regulates which other gene(s)?)
2. Can we devise an intervention (or control) strategy to modify the behavior of this mechanism by means of some external actions?

Determining the relationships between genes is an important issue for biology and medicine. A GRN model provides the researchers the opportunity to understand the insights of cellular processes and even to simulate these processes. Drug discovery research can also benefit from a GRN model in determining new drug targets or the best one for a disease. In addition to the experiments performed *in vivo* and *in vitro*, the relationships between the genes in a living cell are being determined by *in silico* studies recently. This is due to recent increase in both the quality and quantity of the available biological data with the help of new technologies such as microarrays. But, as expected, this also has some challenges involved, since biological data have some amount of noise, missing values and a small sample size.

Devising control strategies for GRNs that will effect the evolution of the network is important to avoid undesirable gene activity profiles. A control or intervention strategy for a GRN can be defined as a way to interact with the network in terms of some actions in order to reach some pre-defined objective(s). These interventions (or actions) are usually defined in terms of (in)activation of certain types of genes or proteins; the objective is to reach (or avoid) a set of state(s) (or gene activity profiles) [17, 65, 66]. There are so many different examples for this type of intervention strategy in biology and medicine; the most well-known ones are some of the methods that are used to treat certain types of cancer. For instance, Gefitinib is a drug used in the treatment of a type of lung cancer and inhibits (inactivates) the epidermal growth factor receptor (EGFR) tyrosine kinase enzyme, which stops uncontrolled cell proliferation of malignant cells. Without this inactivation of EGFR, the cells may continue to divide beyond normal limits. This and other targeted therapies that (in)activate certain types of molecules in malignant cells have recently had a significant impact on the treatment of some types of cancer [31].

Based on the above questions, there are two problems investigated in this thesis. The first can be stated as follows : devise a scalable modeling algorithm that, given a set of biological data, will derive a partially directed or undirected graph that represents the dependencies (or relationships) between the genes. The methods or algorithms proposed in this context will be discussed under the title "Gene Regulatory Network Modeling". To state the second problem, a few other concepts have to be defined first. For a given dynamic (temporal) model of a GRN, suppose we want to avoid the model reaching some of the states (or gene activity profiles); this is called the *objective* of the control problem. Objective is mapped to a control problem in terms of a *reward* function. This is usually defined in terms of some of the genes in the network [17, 65, 66], such as: avoid gene $ACE2$ being expressed. We call the gene(s) in terms of which the objective is defined, the *reward gene(s)* and the gene(s) that are intervened by external actions, the *control gene(s).* Each applied action has a certain cost; for example Gefitinib has a certain price in the market. Now the second problem can be stated as follows: given a GRN, an objective, reward gene(s) and control gene(s), devise a policy (or strategy) to intervene the GRN as effective as possible to reach the objective. The algorithms proposed in this context will be discussed under the title "Gene Regulatory Network Control".

For both of these problems, the focus is on *scalability* in addition to the quality of the output of the proposed algorithms. Scalability is one of the most important issues in GRN modeling and control as the number of genes in a genome is much larger than the number of variables that current modeling and control algorithms can handle.

1.2 Overview of the Proposed Approaches

This section briefly discusses the methods proposed in this dissertation. This overview of the contributions is divided into modeling and control sections whose details exist in Chapters 3 and 4, respectively.

1.2.1 Gene Regulatory Network Modeling

Gaussian Graphical Modeling is a method recently used in GRN modeling. Based on the multivariate normality assumption, this class of methods show promising performance on GRN modeling. The "PC algorithm" is one of the successful algorithms that can be used in this context. It can be used to derive a graphical model of the genes in a given data set by using statistical conditional independence tests. The graphical model output by the PC algorithm is a (partially) directed graph where the nodes are genes and there exists an edge between gene g_i and g_j if there is a direct relationship between g_i and g_j. We can also say that there is an indirect relationship (or dependency) between the genes if there is a path between them in the resulting graph.

The power of conditional independence tests depends on the sample size and the number of elements in the conditioning set which we will name hereafter as *order* following the naming in the PC algorithm literature. So as the order increases, the power (probability of making the right decision) of these tests decreases, in the usual case of a limited sample size. As the number of genes in gene expression data far exceeds the number of samples, the order in the PC algorithm can increase to a very large value. In addition to decrease in power, this also causes the algorithm to consume too much computational resource since there is an exponential number of subsets to be tested as conditioning sets.

There are two methods proposed in this dissertation to overcome the problems associated with using the PC algorithm (or other constraint-based structure learning algorithms) in GRN modeling. The first method is a procedure to integrate multiple types of biological data through conditional independence tests. This way, the method aims at making better decisions in the tests by using the evidence coming from more than one source. The idea here is to adapt the significance level in the tests toward more easily accepting (or rejecting) the null hypothesis according to the evidence coming from the other source. Two different sources of information used are the *gene expression data* and *transcription factor binding location data* (ChIP-chip) (see Chapter 2).

There are various structure learning algorithms that perform well for sparse graphs. These are the graphs where the expected number of connections for the nodes is small. While GRNs are also thought to be sparse, there also exist some dense nodes (genes). These nodes constitute a problem for structure learning algorithms due to (again) the exponential number of conditioning sets in the conditional independence tests of the PC algorithm. For this, we proposed a second method to identify the possible dense genes before executing the modeling algorithm and treat those nodes differently from others during modeling. This different procedure basically identifies the connections (or dependencies) by applying a greedy algorithm for the dense nodes instead of the original exponential one.

1.2.2 Gene Regulatory Network Control

The two frameworks within which control problems have been investigated are Markov Decision Problems (MDPs) and Factored Markov Decision Problems (FMDPs). Both have different advantages which may be the reason that they are both widely investigated and used in the machine learning community. While MDPs are easier to implement and understand, FMDPs can be used for some of the problems which are practically very hard to attempt with MDPs.

The method proposed in this work for scalability in MDPs considers the observation that the effects of all genes in a given data set for the control problem are not equal. Let V be the set of genes in a given data set and assume the control problem is defined as in Section 1.1. Given control and reward genes, g_c and g_r, respectively, we argue that some genes in the set $V \setminus \{g_c, g_r\}$ have a negligible effect on the solution of the problem. To estimate those genes, a score is assigned to each of the genes in $V \setminus \{g_c, g_r\}$; the smaller the score the more negligible that gene is. The proposed score is based on the *Influence* concept discussed in [78]. There is one important property of this reduction procedure that the irrelevant genes are eliminated from the given data even before deriving a model from the data. So, consider the steps of the procedure as: "derive a model from the given data, formulate the control problem as an MDP, solve the control problem"; this method is useful for the scalability of the employed procedure as a whole.

A GRN is naturally factorized, i.e., each gene corresponds to a factor in the model. But to the best of our knowledge, the GRN control problem has not been formalized in an FMDP framework before. In this work, we defined the problem as an FMDP for the first time. In an FMDP, the transition probabilities of the network are modeled using factored representations; by a dynamic Bayesian network for instance. And this usually saves both space and time. But for some of the problems, FMDPs also require exponential resources. So a reduction method for FMDPs may also have a significant effect on the requirements. Based on this argument, a decomposition method that can output good approximate solutions is proposed for FMDPs. Given an FMDP, this method simply decomposes the dynamic Bayesian network associated with the transition probabilities into a number of networks without changing the relationships between reward and control genes. This way, by simplifying the problem but preserving the "power" of the control gene, the FMDP solver can focus only on important parts of the problem and this saves significant computational resources.

1.3 Organization of the Thesis

The rest of this dissertation first covers the basic background required to understand the proposed methods. Then the methods are thoroughly described and their power is demonstrated by a number of experiments. The rest of this section briefly overviews the content of the remaining chapters.

The next chapter discusses some preliminary concepts required to understand the proposed methods. In addition to some biological background for interactions of genes in the cell, microarray technology and existing data types, Chapter 2 also includes basics of graphical modeling of GRNs, the modeling algorithms used (the PC algorithm and Probabilistic Boolean networks) and Markov decision problems.

Chapter 3 includes the proposed methods for GRN modeling. First, the method for dense nodes is introduced; the greedy procedure applied for these nodes and estimating them from prior knowledge is discussed. Then using prior knowledge in conditional independence tests by adapting the significance level in these tests is introduced. The related work in this field is discussed and the gap covered by these methods is explicitly stated in the last two sections of Chapter 3, respectively.

Chapter 4 gives the details of the reduction algorithms proposed for control. It is divided into two main sections where the first one discusses the method for the MDP framework and the second one is about the method proposed for FMDPs. Related work on GRN control is also included in the chapter. The contributions of the chapter are explicitly stated in the last section.

Experimental results are reported and discussed in Chapter 5. Naturally, the results for modeling and control are given in two separate sections and those sections are also divided into subsections for each experiment. Results for both synthetic and real data sets are given in this chapter.

Finally, Chapter 6 includes the summary of the thesis and the future research directions that are planned to be investigated to extend the work discussed.

1.4 Publications

The contributions described in this dissertation have been validated by the experimental study detailed in Chapter 5. Further, different parts of this dissertation have been published in reputable conferences and high quality journals covered by Science Citation Index with high impact factor. Here is a partial list of the already published papers.

- M. Tan, R. Alhajj and F. Polat, "Automated Large-Scale Control of Gene Regulatory Networks," *IEEE Transactions on Systems, Man, and Cybernetics-B,* (forthcoming).

- M. Tan, F. Polat and R. Alhajj, "Large-Scale Approximate Intervention Strategies for Probabilistic Boolean Networks as Models of Gene Regulation," *Proceedings of IEEE Symposium on Bioinformatics and Bioengineering,* Oct. 2008.
- M. Tan, M. Alshalalfa, F. Polat and R. Alhajj, "Combining Multiple Types of Biological Data in Constraint-Based Learning of Gene Regulatory Networks," *Proceedings of IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology,* Sep. 2008.
- M. Tan, F. Polat and R. Alhajj, "Feature Reduction for Gene Regulatory Network Control," *Proceedings of IEEE Symposium on Bioinformatics and Bioengineering,* Oct. 2007.

CHAPTER 2

PRELIMINARIES

This dissertation investigates and proposes novel approaches for gene regulatory network modeling and control. The proposed approaches integrate concepts and techniques from different areas, including molecular biology, graphical modeling, conditionally dependent and independent variables, Bayesian networks, probabilistic Boolean networks and Markov Decision Problems. All these are covered in this chapter in a step to turn the dissertation into a self contained document.

To cover the aforementioned topics, this chapter is organized as follows. Section 2.1 covers the biological background, including the basic molecular components of a cell, the microarray technology and existing biological data types. Characteristics of the graph modeling techniques are discussed in Section 2.2. MDPs are presented in Section 2.3. Section 2.4 discusses how the GRN modeling and control approaches proposed in this thesis benefit from these concepts and techniques.

2.1 Biological Background

This section briefly covers the basics of biology as required to understand the context of this thesis. It gives an overview of the cell and its structure. We dig a bit deeper to understand the molecules (protein, DNA, RNA) in the cell and how they control the cell functions. Then, we introduce the biology of genes, transcription, and gene expression. The microarray technology might be considered as a major constituent of the advance in gene expression data analysis. DNA microarray technology has attracted tremendous interest in both the scientific community and the industry. The data generated by microarray based experiments has been used for disease classification and class prediction.

2.1.1 Genes, Proteins and Their Interactions in the Cell

Giving rise to offsprings is essential for all living organisms. Each offspring inherits the properties of its parent cell or organism. This passing of traits is called heredity and genes are the basic units in a living cell/organism that are responsible for heredity. A gene is encoded in nucleic acids in most of the living organisms. This nucleic acid is called deoxyribonucleic acid (DNA). The other important nucleic acid in the cell is called ribonucleic acid (RNA).

DNA is composed of the long chain of four different bases: adenine (A), cytosine (C), guanine (G) and thymine (T) where these bases together with a sugar molecule and a phosphate group are called nucleotides. These nucleotides are the same in all living organisms but their sequences and amount are different in each organism. DNA exists as a double helix structure in a cell, where the nucleotides pair up; A pairs with T and G pairs with C.

RNA is the other important nucleic acid in the cell. Instead of a double strand in DNA, it is composed of a single strand of nucleotides where T is replaced by Uracil (U). There exist different types of RNA in a cell; each of these perform a different function. Messenger RNA (mRNA) is the most important one which "carries" the genetic information from DNA to the ribosome, the organelle in the cell that produces proteins. The other types of RNA also play important roles in protein synthesis process and gene regulation, and these are transfer RNA (tRNA), MicroRNA (miRNA), small nuclear RNA (snRNA) and ribosomal RNA (riRNA).

DNA has all the necessary information for the functioning of a cell, i.e., it includes all the genes of an organism. Genes correspond to small segments on DNA where each gene encodes a protein essential for the cell. In addition to this, DNA has long sequences of non-coding regions as well, corresponding to no known function.

Proteins are one of the most important macromolecules and participate in every kind of activity in the cell. The basic unit that forms a protein is an amino acid. All proteins are composed of 20 different types of amino acids. A gene encodes a protein by determining which of these amino acids will be used for the production of a protein. In addition to the amino acid sequence, 3D structure of a protein is also important in determining its function in the cell. Proteins can be classified into two groups based on their function; structural and regulatory. Structural proteins, as the name implies, have roles in forming the shape of a living organism and regulatory proteins include enzymes and transcription factors (TFs) that catalyze the reactions and bind to DNA to control protein synthesis, respectively.

The *central dogma of molecular biology* states that the information transfer in a cell is mainly divided into three stages; *replication*, *transcription* and *translation*. Replication is the stage where DNA duplicates itself for a new offspring of the cell. Transcription is the process of "copying" the information on DNA to a mRNA. Then, after transcription is completed, mRNA is translated into a protein by the help of ribosome and some enzymes.

The property of proteins that makes them essential for the cell is their ability to bind to other molecules. For example they can bind to other proteins forming complex proteins or they can bind to specific regions on DNA called promoters to control gene expression. A gene is said to be expressed if it is transcribed into mRNA. TFs are the proteins that bind to promoters and regulate gene expression. A TF can either be monomeric or be the result of binding of more than one protein. The interaction between a TF and the promoter region of a gene on DNA can be named as a gene-protein interaction. All these protein-protein and gene-protein interactions constitute a large network of interactions and the gene expression is controlled by this network. This network is usually referred to as a gene regulatory network (GRN) and can be represented as a graph (See section 2.2) where the nodes are genes/TFs and the edges are the relationships between them.

2.1.2 Microarray Technology

Microarray is the name of the technology that gave researchers the opportunity to determine the expression levels of large numbers of genes in parallel. The data produced by a microarray experiment provides the ability to see a large proportion of the genes in the genome of an organism. This section discusses the complementary DNA (cDNA) microarray experiments which is the most widely performed one [96].

Hybridization is the process of binding of two complementary single stranded nucleic acids. DNA can be produced from mRNA by a process called reverse transcription with the help of the enzyme, reverse transcriptase. This DNA is called cDNA and it can hybridize to mRNA. These two concepts form the basis for the cDNA Microarray experiments. The steps of a cDNA microarray experiment are given in Figure 2.1 and can be enumerated as follows:

1. Target DNA preparation
2. Slide preparation
3. Printing of DNA on chips
4. cDNA preparation and labeling
5. Hybridization
6. Scanning

Microarray chips are constructed by commercial companies by polymerase chain reaction (PCR) methodology. PCR produces single stranded DNAs to spot on a glass slide. So each spot contains numerous identical copies of a gene from the organism used. The genes corresponding to each spot are recorded. Then by reverse transcription, the cDNAs of interest are produced. cDNA microarray experiments are usually performed to compare two types of conditions of two different cells where one represent the experimental conditions and other represent the reference conditions. As shown in Figure 2.1, these can be cancer and normal cells. cDNAs for both conditions should be labeled by incorporating fluoresecently labeled nucleotides during reverse transcription. Usually they are labeled with either a red or a green dye, where each

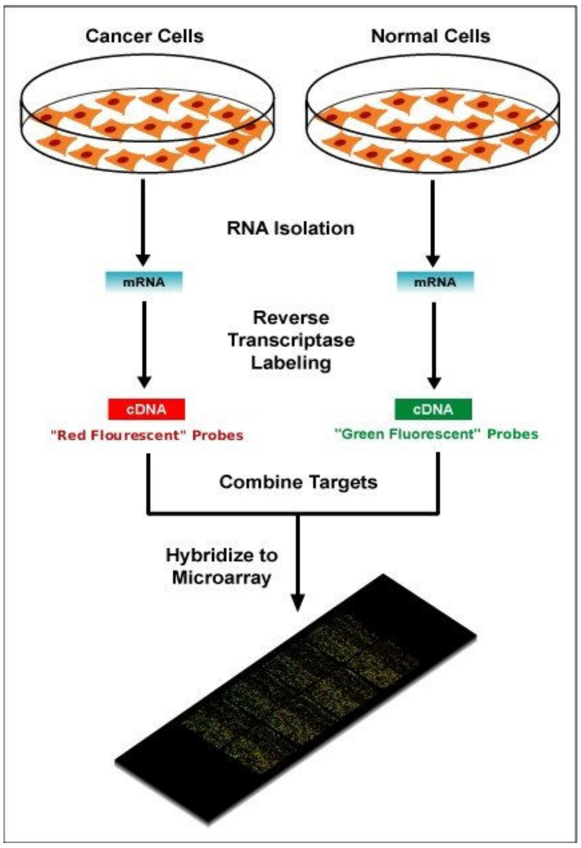


Figure 2.1: Steps of a microarray experiment (adapted from [94])

color labels either experimental or reference conditions. Then both types of cDNA are put on the same slide for hybridization with the DNA on the microarray chip. This hybridization process generally lasts for one night and after this, in addition to hybridized cDNA, there will also be some amount un-hybridized cDNA on the chip of experimental or reference conditions. So the chips are washed in this step to remove any remaining unbound cDNAs. Two images are then produced from the chip by scanning, where one image is for one color and the other is for the other color. A merged image is also produced from the two images. This image is further processed by image processing techniques to produce readings for the green and red labels. The ratio of red to green or vice versa outputs the relative expression level of the genes in the experimental conditions.

2.1.3 Existing Biological Data Types

In this section, we review a set of data types that are widely used by the bioinformatics community. This set also includes the biological data types analyzed in this thesis.

Gene Expression Data

Gene expression data D^{exp}, obtained usually from microarray experiments, is an $m \times n$ matrix of expression values. D^{exp}_{ij} entry of this matrix corresponds to the expression value of gene i under condition j. There are two types of expression data widely used; time series and classification (or sometimes referred to as static) data.

Time series data, as the name implies, has the change of expression values of genes over time. So the microarray experiment is designed to get the measurements of expression over a number of time steps. This type of data is extensively studied and also introduces some challenges in both experimental design and analysis [4].

Static or classification data is the snapshot of the expression levels of genes in different samples. These different samples are usually used to compare two or more different types of cells; cancer versus normal tissue samples for instance. Unlike the case in time series data, these are assumed to be independent and identically distributed.

Transcription Factor Binding Data

This type of data is obtained by a technique called genome-wide location analysis [70]. This method is a combination of modified chromatin immunoprecipitation(ChIP) and DNA microarray analysis. ChIP method used here provides the ability to detect the binding site of any protein *in vivo* [63].

TF binding data D^{tf}, at the end, is in the form of a $m \times n$ matrix where m is the number of genes in the experiment and n is the number of TFs. D^{tf}_{ij} entry of the data is a p-value indicating the level of confidence that TF j binds to the promoter region of gene i; the smaller the p-value, the larger the probability of binding [53]. This type of data is one of most effective in determining the associations between genes in an organism [6, 34, 53, 105].

Protein-protein Interaction Data

As mentioned before, after synthesis, proteins can form complexes with other proteins for functioning. Protein-protein interaction (PPI) data include physical interactions between proteins in an organism. A physical interaction here refers to the experimentally verified binding of two amino acid chains. Such data sets are useful for working on the specific proteins as well as whole genome interactions [39, 74, 84]. PPI data is usually in the form of a 0-1 matrix D^{pp} of interactions, where each entry D^{pp}_{ij} determines whether proteins i and j are experimentally determined to be interacting.

2.1.4 Pre-processing Gene Expression Data

Gene expression data may need some pre-processing before mining meaningful knowledge. The first reason for this is the lack of standardization in the experiments. This not only introduces differences in readings in the same conditions but also brings some noise to the data. The second reason is that data have many null entries; i.e., for a given gene some expression values may not be available. Normalization is the pre-processing method that helps remove the noise and make the expression values comparable for different experiments. Pre-processing methods also exist for imputing missing values in gene expression data by changing the original distribution of data as little as possible. It has also been shown that pre-processing the data has certain effects on gene network inference [56]. Since some methods work on discrete data rather than continuous, discretization of gene expression data can also be listed among the pre-processing steps.

Normalization

Several methods have been proposed for normalization of gene expression data [69, 100]. The first one is based on the concept of house-keeping genes. These genes are assumed to be always active at a certain level of expression. So the expression level of these genes is used as a reference for normalization. The expression levels of other genes are divided by the expression level of the house-keeping gene in this type of normalization.

The second one is called total intensity based normalization. This method is based on the assumptions that the mRNA amount for each sample compared is equal and the same number of labeled molecules hybridizes to the arrays for the samples, where the intensity here refers to the readings of green or red labeled spots on the slide after processing the final image of microarray chip. A normalization factor is calculated as the ratio of the sum of red to green intensities and each intensity value is multiplied by this factor such that the mean ratio of intensities become 1.

Missing value imputation

Most of the data mining algorithms require complete data. For this reason, several missing value imputation methods have been proposed for gene expression data [88]. Of several methods, the method based on the k-nearest neighbor (KNN) algorithm is the most widely used. The KNN algorithm simply chooses the k other genes that are most similar to the gene that has a missing value. Then a weighted average of these k genes are imputed as the value of the gene. The similarity metric used here is very important where Euclidean distance is proposed in [88].

Discretization

Discretization is one of the issues in gene expression data processing that still suffers from general consensus. There are numerous algorithms proposed for discretizing data [57]. The simplest one divides the interval between minimum and maximum values of a given attribute into a given number of sections (bins). Each bin is then assigned a different discrete number. Each expression level is mapped to the corresponding discrete number. The quantization level (number of discrete bins) becomes important in almost all of the discretization methods, where 2 or 3 is common for gene expression data corresponding to {ON,OFF} and {under-expression, baseline, over-expression}, respectively [29, 101].

2.2 Graphical Modeling

A graph G is defined as a pair (V, E), where V is a set of nodes and E is a list of (ordered or unordered) pairs (i, j) to represent that nodes i and j are connected in G. We will use E_{ij} to denote the edge between i and j. This connection may have many interpretations depending on the domain. For example, for GRNs considered in this work, the nodes are the genes and the existence of E_{ij} denotes that the expression level of gene i is in some way related to the expression level of gene j. Graph G can be undirected, which implicitly means that $E_{ij} \in G \Rightarrow E_{ji} \in G$, *i.e.*, direction is not important. On the other hand, a Directed Acyclic Graph (DAG) is a graph where the edges are directed and the graph does not contain any cycles. It is also possible for a graph to have both undirected and directed edges; such a graph is usually called a Partially Directed Acyclic Graph (PDAG), which also does not include any cycles. In a graph G, two nodes i and j are called *adjacent* if $E_{ij} \in E$ or $E_{ji} \in E$.

A DAG G and a probability distribution P are said to be *faithful* to each other if G denotes all and only the conditional independence relationships in P in the form of what is called *d-separations.* To better understand the definition of d-separation, it is necessary to first introduce the conditional independence relationship and some graph related concepts.

Two variables i and j are said to be conditionally independent with respect to a probability distribution given a set of variables S if and only if:

$$P(i, j|S) = P(i|S)P(j|S) \tag{2.1}$$

In this work, we use $Ind(i, j|S)$ to denote the independence relationship expressed in Eq (2.1).

A *path* P in a DAG G is a set of nodes $\{i_1, i_2, i_3, ..., i_n\}$, such that starting at node i_1 we can reach node i_n by following the sequence of edges $E_{i_k i_{k+1}}$ $(k = 1\ to\ n)$. In a DAG G, node i is called a *collider* in a path if there are two nonadjacent nodes j and k such that $E_{ji} \in E$ and $E_{ki} \in E$. In this case, the triplet (j, i, k) is called *v-structure* (see Figure 2.2). An undirected path U is said to be *blocked* by a set of nodes W if any of the following two conditions hold

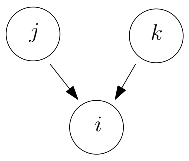


Figure 2.2: (j, i, k) v-structure

$\forall i \in U$:

- i is a collider, and neither i nor its descendants are in W
- i is a non-collider, and it is in W.

Two nodes i and j are said to be *d-separated* by a set of nodes S if and only if every undirected path between i and j is blocked by S.

It is possible to have more than one DAG generating the same probability distribution P [14]; this defines an equivalence class among DAGs with respect to P. The *skeleton* of a DAG is the undirected graph obtained by replacing directed edges with undirected ones. Two DAGs are equivalent if and only if they have the same skeleton and the same set of v-structures [91]. It is possible to represent such an equivalence class with a PDAG. A PDAG that completely represents an equivalence class of DAGs is called *Complete Partially Directed Acyclic Graph* (CPDAG). The aim of most of the structure learning algorithms is to find such a CPDAG representing the equivalence class of DAGs faithful to the underlying probability distribution P.

A Bayesian Network (BN) is a tuple (G, P), where $G = (V, E)$ is a DAG and P is a joint probability distribution on V. Both G and P satisfy the Markov condition in a BN; all the variables are independent of their non-descendants given their parents. A BN is said to be faithful if all and only conditional independence relationships are the ones that are entailed by Markov condition.

Structure learning algorithms for BNs refer to a set of algorithms that try to find the DAG component of the BN given some data sampled from a probability distribution. These algorithms basically fall into two categories. The first category is the search-and-score based algorithms, which search the space of DAGs (or CPDAGs) for the graph that maximizes a score function. The other class of algorithms is known as constraint-based algorithms [82]; the latter algorithms start with a fully connected graph and search for conditional independencies in the probability distribution, generally by means of statistical conditional independence tests.

In this thesis, we assume that the data are from a multivariate normal distribution. This assumption has been widely used recently in GRN modeling, *e.g.*, [11, 42, 75, 95], where the name GGM is given to this modeling framework. Under this assumption, vanishing partial correlations imply conditional independence [52]. Sample partial correlations can be calculated from the given data with various methods, including regression, inversion of covariance (correlation) matrix or recursion. Here, we use the method and mathematical notation of Kalisch *et al.* [43].

To test conditional independence, Fisher's z-transformation is applied to a partial correlation. This transformation can be expressed as follows:

$$Z(i,j|S) = \frac{1}{2}log\left(\frac{1+\hat{\rho}_{i,j|S}}{1-\hat{\rho}_{i,j|S}}\right) \tag{2.2}$$

where $\hat{\rho}_{i,j|S}$ denotes the sample partial correlation of i and j conditional on set S. Then, given a significance level α, the null-hypothesis $H_0(i,j|S) : \rho_{i,j|S} = 0$ is rejected against the two sided alternative hypothesis $H_1(i,j|S) : \rho_{i,j|S} \neq 0$ if,

$$\sqrt{n-|S|-3}|Z(i,j|S)| > \Phi^{-1}(1-\alpha/2) \tag{2.3}$$

where Φ is the cumulative distribution function of normal distribution with mean 0 and variance 1, *i.e.*, $N(0,1)$.

2.2.1 The PC Algorithm

One of the most well-known constraint-based structure learning algorithms is the PC algorithm [82]. The algorithm is composed of two parts; the first part constructs the skeleton of the graph, and the second part orients the undirected edges in the skeleton. Given in Algorithm 1 is the process which is usually referred to as the first part of PC algorithm.

The proposed methods in this thesis modify the first part of the PC algorithm. The edge orientation part does not need any modifications in order to be applied to the results presented here. If we assume a faithful distribution to a DAG G and a perfect knowledge of conditional independence relationships, the PC algorithm correctly infers the skeleton of the underlying DAG G [82]. The worst-case complexity of the PC algorithm is $O(p^{ord_m})$, where ord_m is the maximum value of ord (see Algorithm 1) and p is the number of variables. Moreover, given the above assumptions, if we denote the maximum number of neighbors of a node in G by q, $ord_m \in \{q-1, q\}$; and the algorithm is known to scale well for sparse graphs [43].

2.2.2 Probabilistic Boolean Networks

PBNs are probabilistic extensions of Boolean Networks (BoNs), which were first introduced by Kauffmann [46]. We will briefly discuss here basic concepts about BoNs and PBNs; the reader is referred to [46, 78, 79] for further details.

A BoN $G(V,F)$ is defined as a set of nodes $V = (x_1, x_2, ..., x_n)$ and a set of Boolean functions $F = (f_1, f_2, ..., f_n)$. Every node in V has a k-ary ($k \leq n$) Boolean function f_i that determines its value. Without loss of generality, f_i can be considered as n-ary with some

Algorithm 1 PC algorithm (first part)

```
Input: Data $D$, Set of nodes $V$, Conditional independence test $Ind$
Output: Skeleton of the graph $G$, Separator information $Sep$
 1: Set $G$ to the fully connected undirected graph of $V$
 2: $ord = 0$
 3: repeat
 4:     repeat
 5:         Choose new adjacent ordered pair of nodes $i, j$ with $i$ having at least $ord$ neighbors
 6:         repeat
 7:             Choose new set $S$ of nodes adjacent to $i$, where $|S| = ord$
 8:             if $Ind(i, j|S)$ then
 9:                 Delete edge $i, j$ from $G$
10:                 $Sep(i, j) = S$
11:             end if
12:         until (edge $i, j$ is deleted) or (all different sets $S$ of length $ord$ have been tested for edge $i, j$)
13:     until all pairs of adjacent nodes have been tested
14:     $ord = ord + 1$
15: until number of neighbors for each node in $G$ is less than $ord$
16: return $G, Sep$
```

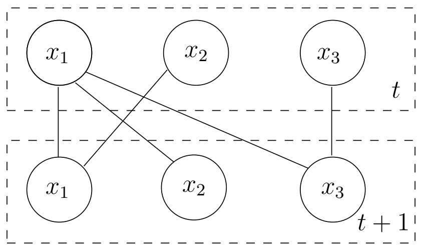


Figure 2.3: Wiring diagram of a BoN.($x_1(t+1) = f_1(x_1(t), x_2(t)); x_2(t+1) = f_2(x_1(t)); x_3(t+1) = f_3(x_1(t), x_3(t))$).

fictitious (unnecessary) variables. A variable x_i is fictitious for f if,

$$f(x_1, ..., x_{i-1}, 0, x_{i+1}, ..., x_n) = f(x_1, ..., x_{i-1}, 1, x_{i+1}, ..., x_n) \tag{2.4}$$

A variable that is not fictitious is called essential. Wiring diagrams are useful in representing a BoN [3]. Figure 2.3 shows an example wiring diagram.

As can be seen, the dynamics of a BoN is completely deterministic. The only probabilistic aspect of a BoN is the selection of the initial starting state. If we represent the initial state of the network with a joint probability distribution $D(x)$ where $x \in \{0,1\}^n$, it can be shown that the dynamics can be modeled by the equation below that resembles a Markov chain;

$$D^{t+1} = \psi D^t \tag{2.5}$$

where ψ is a mapping of the form $\psi : \{0,1\}^{2^n} \rightarrow \{0,1\}^{2^n}$.

A BoN represents gene expression by using only two levels: ON and OFF. The expression level for a gene g_i at time step $t+1$ is related to the expression level of k_i other genes at time t by a Boolean function, $f^{(i)}(g_{i_1}, ..., g_{i_k})$, where genes g_{i_1} to g_{i_k} are called *parents* of g_i. So, a BoN is defined by a set of genes $V = (g_1, ..., g_n)$ and a set of Boolean functions $F = (f^{(1)}, ..., f^{(n)})$. On the other hand, PBNs assign a set of functions to each gene instead of a single function. At each time step, a function from that set is chosen to determine the next-step value for a gene. Formally, a PBN is defined by a set of genes $V = (g_1, ..., g_n)$ and a set $F = (F_1, ..., F_n)$, where each F_i is a set of functions for g_i, $F_i = \{f_j^{(i)}\}_{j=1,...,l_i}$. Each $f_j^{(i)}$ is one of the possible functions to determine the next state of g_i, and l_i is the number of such functions. The probability of choosing $f_j^{(i)}$ in F_i to predict the next state of g_i is denoted $c_j^{(i)}$.

Given binary quantized gene expression data, deriving a PBN model requires finding F and $c_j^{(i)}$ for all i and j. To do this, a measure of how well a function predicts the value of a gene is needed. Coefficient Of Determination (COD) [23] is one such measure. COD compares the prediction performance of a function with the best constant estimator in the absence of other information. Assume that we are given the parents P_i of g_i and a function $f_j^{(i)}(P_i)$ to predict

g_i. The COD θ_j^i of $f_j^{(i)}$ is defined as follows:

$$\theta_j^i = \frac{\varepsilon_i - \varepsilon(g_i, f_j^{(i)}(P_i))}{\varepsilon_i} \tag{2.6}$$

where ε_i is the error of the best constant estimate of g_i and $\varepsilon(g_i, f_j^{(i)}(P_i))$ is a probabilistic error measure [79]. Given θ_j^i values, it is straightforward to define $c_j^{(i)}$ [79]:

$$c_j^{(i)} = \frac{\theta_j^i}{\sum_{m=1}^{l_i} \theta_m^i} \tag{2.7}$$

Best-Fit extension paradigm

For a given set of parent genes P_i, $f_j^{(i)}$ can be derived using various methods. In our work, we use best-fit extension paradigm of Lähdesmäki *et al.* [51].

Lähdesmäki *et al.* [51] try to derive a model for the gene regulatory networks by using Boolean networks. They study on two different aspects of the problem, that are called the consistency problem and the best-fit extension problem. The consistency problem (or Extension problem) is concerned with deriving a consistent function that is a Boolean function f from a class of functions C that perfectly separate the given true and false examples in the given data. A partially defined Boolean function $pdBf(T,F)$ is defined by two sets, T and F that denote the true and false examples in the given data, respectively. If, for a Boolean function f, we define the true and false examples as $T(f) = \{x \in \{0,1\}^n : f(x) = 1\}$ and $F(f) = \{x \in \{0,1\}^n : f(x) = 0\}$ then formally, consistency problem is simply defined as whether there exists a consistent extension f for $pdBf(T,F)$ such that $T \subseteq T(f)$ and $F \subseteq F(f)$. The other problem that is investigated by the authors is the best-fit extension problem. If we assume we are given, in addition to a $pdBf(T,F)$, a set of weights $w(x)$ for all examples $x \in T \cup F$ then the best-fit extension problem is to find a Boolean function f that minimizes the error which is given as,

$$\varepsilon(f) = w(T \cap F(f)) + w(F \cap T(f)) \tag{2.8}$$

where the weight of a set is defined as the sum of the weights of individual elements of the set.

For a network of n-nodes, the algorithms for solving the consistency problem for n-variable and k-variable functions are given where $0 \leq k \leq n$. The algorithms rely on the fact that to solve the consistency problem, T and F must be disjoint. The algorithm for n-variable and k-variable functions simply fills an initially empty truth-table according to the given data. An inconsistency can be detected while filling, which means that there does not exist a solution for the consistency problem. The same algorithm can be applied also for the k-variable functions by executing the algorithm for all k-element subsets of variables. Then each undetermined entry in the table is filled arbitrarily with 0 or 1.

The solution for the best-fit extension problem is also similar. The same problem was initially shown to be polynomial time solvable using another method by Shmulevich et. al. [80].

The idea here is to define two 2^n dimensional vectors $c^{(0)}$ and $c^{(1)}$, where each element in the vectors indexes a possible variable assignment to n variables for negative and positive examples, respectively. So the i^{th} element corresponds to the weight of the i^{th} variable assignment (for example for n=3, the variable assignment $\{1,1,1\}$ corresponds to the 7^{th} index). Then the solution is shown to be simply the function f that has the truth-table $\mathbf{f}_i = argmax_j c_i^{(j)}$, where $\mathbf{f}_i$ denotes the output value for the i^{th} indexed variable assignment for input variables of f. Then the authors give an algorithm to find all functions that have error less than a threshold. The results are given on the cell-cycle data of Spellman et al. [81] where the functions for a number of genes are identified having error less than 5 for a unit weight assigned to each variable assignment.

In this work, we use publicly available Matlab implementation of Best-Fit Extension in PBN-Toolbox[1].

2.3 Markov Decision Problems

A MDP is formally defined as a quadruple (S, A, T, R), where S is the set of states, A is the set of actions, T is the transition probability function such that $T(s, a, s')$ denotes the probability of the next state being s' given the current state s and action a, and R is the reward function that represents the objective of the control process. Any MDP is associated with a performance criterion. The performance criterion we adapt is the infinite horizon total discounted reward criterion. So the objective is to maximize the total discounted reward: $\sum_t \beta^t R_t(s, a)$, where $R_t(s, a)$ is the immediate reward of performing action a in state s at time t and $\beta \in (0, 1)$ is the discount factor. In this work, we assume that R_t and β are independent of t; so we omit subscript t after this point.

Solution to an MDP is called a policy, π; it is a mapping from states in S to actions in A. Every π defines a value function V^π from S to real numbers. $V^\pi(s)$ is the total discounted future reward of choosing an action a according to π in state s, and following π thereafter. V^π can be found iteratively using the following equation:

$$V_{k+1}^{\pi}(s) = R(s, \pi(s)) + \beta \sum_{s'} T(s, \pi(s), s') V_k^{\pi}(s') \qquad (2.9)$$

where iteratively applying Eqn 2.9 is called policy evaluation.

Optimal policy π^* is the best policy in terms of the given performance criterion. In our case, it is the policy that achieves maximum possible infinite horizon discounted future reward. Value function corresponding to π^* is the optimal value function, V^*, which can also be found

[1]available at: "Probabilistic Boolean Networks, `http://personal.systemsbiology.net/ilya/PBN/PBN.htm`, accessed 1-June-2009"

iteratively using the following Bellman update:

$$\forall s \in S \qquad V^*_{k+1}(s) = max_a[R(s,a) + \beta \sum_{s'} T(s,a,s')V^*_k(s')] \tag{2.10}$$

Given all components of an MDP, Eqn 2.10 converges to the unique V^* as $k \to \infty$. From V^*, π^* can be found as:

$$\pi^*(s) = argmax_a[R(s,a) + \beta \sum_{s'} T(s,a,s')V^*(s')] \tag{2.11}$$

With arbitrary initialization of V_0, the algorithm that uses Eqn 2.10 to find V^* is called *value iteration* [5]. One simple stopping criterion for value iteration is:

$$||V_{k+1} - V_k|| \leq \frac{\epsilon(1-\beta)}{2\beta} \tag{2.12}$$

where $||X|| = max\{|x| : x \in X\}$ denotes maximum norm. Eqn 2.12 ensures V_{k+1} is within $\epsilon/2$ of V^* for any state [68].

Another well-known algorithm for solving an MDP is the policy iteration algorithm [68]. Instead of starting with arbitrary V, policy iteration starts with an arbitrary policy π, and finds V^π using Eqn 2.9. Then for all states s, it searches for an action a that satisfies the following equation:

$$V^\pi(s) < R(s,a) + \beta \sum_{s'} T(s,\pi(s),s')V^\pi(s') \tag{2.13}$$

If found, it updates $\pi(s)$=a, and repeats the policy evaluation and update steps until convergence criterion is met.

There are several other proposed methods for solving MDPs. We refer the reader to the books [5, 68] for further details.

2.3.1 Factored MDPs

A FMDP is a representation language for MDPs to exploit the structure of the control problem. The FMDP framework was first proposed by Boutilier *et al.* [8]. In most problems, T can be represented in terms of a set of state variables, where in our case these variables correspond to genes.

As representing T for a MDP requires exponential space in the number of variables, FMDP proposes to represent T for each specific action in the form of a dynamic Bayesian network (DBN) [21]. A DBN is composed of variables $G = (g_1, g_2, ..., g_n, g'_1, g'_2, ...g'_n)$, where the variables with a prime denote the random variables at the next time step. So, a DBN represents the relationships between random variables in the current and next time steps. We denote the set of primed variables by X' and non-primed by X, where $G = X \cup X'$. Each variable g_i' has a set of parents P_i, where the value of g_i' depends only on P_i. In this work, we assume that $P_i \subset X$, and the variables in X do not have any parents, i.e., there are no synchronous dependencies between variables, all dependencies are between the variables at time step t and the variables at time step $t + 1$. This is a common assumption for modeling GRNs using a DBN.

A DBN associates to each g_i' and its parents P_i a conditional probability distribution (CPD). A discrete CPD is usually represented as a table. But some space can be gained if CPDs are represented by decision trees in case they have the same values for different instantiations of the parents [9].

In addition to CPDs, the structure in V and π can also be exploited to represent them by decision trees. The idea here is that V and/or π may depend only on some of the variables instead of all of them. So, they may be represented by a decision tree as well. Both value trees and policy trees have internal nodes labeled with the variables themselves and edges labeled with the values (instantiations) of the variables. Leaf nodes of a value tree have values of the states corresponding to all states that have the same instantiations of the variables in the path from the root to the leaf. The same way, leaf nodes of a policy tree have the actions corresponding to the states that have the same instantiations of the variables in the path from the root to the leaf. The reader is referred to [9] for details.

Solving FMDP requires modifying these value and policy trees at each iteration. Decision-Theoretic Regression [9] is one of the methods to modify decision tree representations of value and policy trees; each iteration results in a new value or policy tree that is closer to the decision tree representation for V^*. Structured value and policy iteration are two algorithms that use decision-theoretic regression to solve FMDPs [9]. Efficient methods to solve FMDPs by linear programming are described in [32]. Finally, we use the publicly available FMDP solver, SPUDD[2] ("Stochastic Planning using Decision Diagrams") [37]. Instead of using decision trees, SPUDD uses algebraic decision diagrams (ADD) [72]. SPUDD package also includes an approximate FMDP solver, APRICODD ("Approximate Policy Construction using Decision Diagrams") [83].

There are two approximation methods in APRICODD that depend on pruning the value tree. The first one is on keeping the value tree below a fixed size, which is good for solving FMDPs with limited computational resources. The second one uses ADDs in which the similarly valued leaves of an ADD are merged and such leaves are labeled with a range of values. This results in a smaller sized ADD called ranged value ADDs. Merging these values depend on a given error bound such that only the values that are within that error bound are merged.

[2]available at: "Welcome to SPUDD, `http://www.computing.dundee.ac.uk/staff/jessehoey/spudd/`, accessed 1-June-2009"

At the end, the midpoints of the resulted ranged value ADDs are returned as the value of the corresponding states. APRICODD also includes variable reordering schemes that can have significant effects on the resulting ADD size. Further details about SPUDD and APRICODD can be found in [38].

In terms of GRNs, given PBN model derived from some kind of biological data, actions, and the objective defined in terms of the reward genes, the PBN control problem can be solved by the following steps:

1. Convert PBN to DBN
2. For each action $a \in A$, construct DBN_a that represents probability distribution $T(s, a, s')$ for all s, s'
3. Given reward function R and discount factor β, define FMDP M
4. Solve M using SPUDD

2.4 Summary

Having defined the concepts, linking the subjects covered in this chapter with the work done in this thesis would be helpful to better understand the rest of this document.

One of the two problems considered in this work is the scalability in modeling GRNs. Two modifications for PC algorithm are proposed for scalability and quality of the derived networks. To evaluate these modifications, two of the discussed data types in Section 2.1.3 were used; gene expression and TF binding location data. Using the last one, protein-protein interaction data, is left as a future work. The methods in Section 2.1.4 are used to pre-process expression data when necessary.

The second problem investigated is the scalability in control of GRNs. We used MDP and FMDPs as the framework for solving the control problem where PBNs are exploited as the model for control as a discrete model is necessary to evaluate the control algorithms.

CHAPTER 3

CONSTRAINT-BASED MODELING OF GENE REGULATORY NETWORKS

The regulatory mechanisms for gene expression in a cell is very important as they control the activities related to protein synthesis which is one of the most essential functions for living organisms. Depending on the evolutionary level of the organism, this regulatory mechanism can get highly complex. The interaction of the large number of genes (promoters) and proteins constitute this whole mechanism. An example of such mechanisms which is a component of the whole GRN of *Caenorhabditis elegans* is given in Figure 3.1. The network in Figure 3.1 is composed of the genes involved in the development and function of *C. elegans* digestive tract; the reader is referred to [22] for details on the construction of this network.

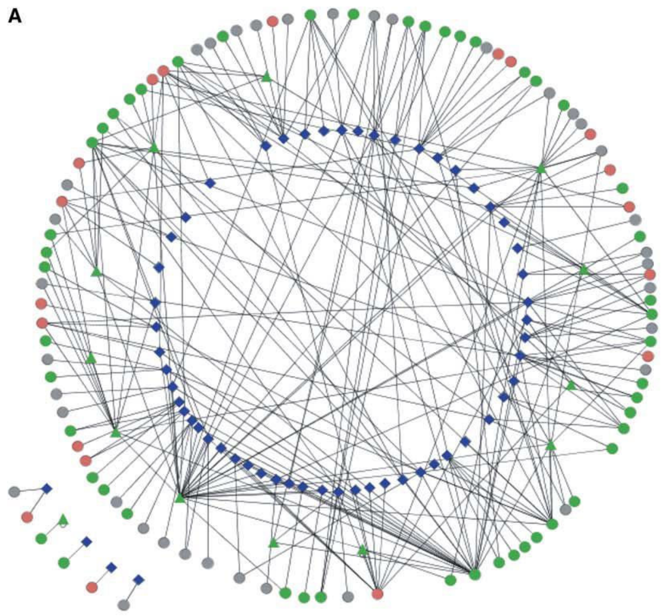


Figure 3.1: Protein-DNA interaction network of *C. elegans* (Taken from [22]). Blue diamonds, circles and triangles represent promoters, interactors and interactors whose promoters are also used as DNA baits in the experiment, respectively.

The issue that we want to emphasize here is the complexity of the network in Figure 3.1; the number of genes and the number of connections. Learning a network of this size and complexity is hard for current structure learning algorithms. There are methods that can be used to learn sparse graphs with a large number of nodes (see [43] for example), but some serious difficulties, both in terms of the quality and computational requirements, arise in case of nodes with large number of connections. For instance, there are interactors in Figure 3.1 binding 27 promoters which is a large number for structure learning algorithms for graphs of this size. Therefore, new scalable methods for learning GRNs from biological data are needed.

The amount of biological data available for research is exponentially increasing. However, GRN modeling still suffers from the problem of small sample size compared to large number of genes. This problem is sometimes referred to as the "p larger than n" problem, where p refers to number of genes and n refers to number of samples [11]. The solution investigated for this problem in this thesis is to incorporate multiple types of biological data. We will name the information inferred from one type of data as the prior knowledge; there are no constraints on which one will be the prior knowledge though. This prior knowledge will be used to "adapt" the modeling algorithm to infer better networks.

In the context of GRN modeling, the prior knowledge can be formulated as a matrix B of probabilities such that each entry B_{ij} gives the probability of existence of edge E_{ij} [6, 93]. This information might be obtained from various types of data and can be used with expression data to obtain a better GRN.

Two methods for incorporating B in the PC algorithm by adapting the conditional independence test to the given prior knowledge B are described in the following sections. The first method is a simple but effective procedure to use the prior information on the conditional independence tests in the PC algorithm. This way, the test "adapts" itself to the given prior knowledge. This method is described in Section 3.2.

Although the adaptation procedure is successful in incorporating prior knowledge into the PC algorithm, it may also lead to some problems regarding maximum order in the PC algorithm. It is well-known that with increasing order, the power of a conditional independence test decreases given small sample size. To be able to solve this problem, the second method is proposed in Section 3.1 for the type of graphs that we call Partially Dense (PD) graphs. A PD graph is a graph where some nodes have significantly larger number of connections than others. The method interprets the same type of information as above with a different perspective. The nodes that have a large number of connections are identified from prior knowledge and these nodes are treated differently. Details of this method are presented in Section 3.1. The related work is covered in Section 3.3. Finally, Section 3.4 highlights the shortcomings of the approaches described in the literature and summarizes how they are covered by the proposed approaches.

3.1 PC algorithm for PD graphs

The number of conditional independence tests required by the PC algorithm, in the worst-case, is bounded by $p^2(p-1)^{q-1}/(q-1)!$ [82]. So, the algorithm can easily become non-applicable for high values of q and p. Our experiments depicted that if we know the dense nodes in the graph in advance, then a variant of the PC algorithm can show a good performance, even for PD graphs, both in terms of computational requirements and the quality of the resulting graph.

Algorithm 2 PCPD algorithm (first part)

```
Input: Data $D$, Set of nodes $V$, Conditional independence test $Ind$, Set of dense nodes $DN$
Output: Skeleton of the graph $G$, Separator information $Sep$
 1:                                                              ▷ Stage 1
 2: Set $G$ to the fully connected undirected graph of $V$
 3: $ord = 0$
 4: repeat
 5:     repeat
 6:         Choose new adjacent ordered pair of nodes $i, j$ with $i$ having at least $ord$ neighbors and $i \notin DN$
 7:         repeat
 8:             Choose new set $S$ of nodes adjacent to $i$ where $|S| = ord$
 9:             if $Ind(i, j|S)$ then
10:                 Delete edge $i, j$ from $G$
11:                 $Sep(i, j) = S$
12:             end if
13:         until (edge $i, j$ is deleted) or (all different sets $S$ of length $ord$ have been tested for edge $i, j$)
14:     until all pairs of adjacent nodes have been tested
15:     $ord = ord + 1$
16: until number of neighbors for each node in $G$ is less than $ord$
17:                                                              ▷ Stage 2
18: for all i in DN do
19:     Choose new $j$ such that $i$ is adjacent to $j$ in $G$
20:     $[S, vanished] = gss(D, i, j, G)$
21:     if vanished then
22:         Delete edge $i, j$ from $G$
23:         $Sep(i, j) = S$
24:     end if
25: end for
26: return $G, Sep$
```

To be able to correctly infer d-separations, for an edge E_{ij}, conditioning set S in the algorithm should be chosen as a subset from neighbors of either i or j. The PC algorithm does this by choosing *ordered* pairs (i, j) during execution. If the number of neighbors for a node is high (i.e., q is high) then the algorithm tries exponentially increasing number of conditioning sets S with increasing order. This terribly slows down the algorithm response in the existence of a dense node. Also, with increasing order, (with limited sample size) the probability of error in statistical tests for conditional independence increases [82]. So, a possibly incorrect graph (with limited sample size) is derived in a long time.

One method that has been proposed in various studies to avoid the above problem is to limit the maximum value of order [20, 58, 95]. Since the nodes in a GRN are generally sparse, this method may give good results. But, if q is large even for a small number of nodes, we may still face the same type of problems described above. Another possible approximate solution method is to use unordered pairs in the algorithm and choose S from the neighbors of the node that has less number of neighbors. But this also suffers from the same problems if the underlying graph has more than one node with large number of neighbors.

As a result, we propose a new method for finding the skeleton of the underlying model with some dense nodes. This method depends on the prior knowledge about these dense nodes and a greedy search procedure for the separators of the edges of dense nodes. The proposed method, which we call *PC algorithm for Partially-Dense graphs* (PCPD), is given in Algorithm 2.

As shown in Algorithm 2, PCPD has two stages. The first stage is similar to the PC algorithm except that we do not choose edge E_{ij} to test in Line 6 if we know that i is a dense node. This avoids enumerating exponential number of subsets of neighbors of i.

By skipping some of the tests performed, at the end of Stage 1, some false positive edges may exist in G; these are the edges that have at least one dense node in one end. But the number of such edges are not expected to be large because we check edge E_{ji} even though we do not check edge E_{ij}. For such edges for which the algorithm could not find a separator from the subsets of neighbors of j, the greedy separator search procedure is executed. This procedure is described in the next section.

3.1.1 Greedy separator search

Greedy separator search (*gss*) is a search procedure to find a conditioning set S that makes the two nodes i and j conditionally independent given S. To describe the algorithm we will call the left-hand side of Equation 2.3 as $L(i, j|S)$. In each iteration, *gss* adds variable v to the current conditioning set S where $v = argmin_{v \in V} L(i, j|S \cup \{v\})$. This way it searches for S that satisfies Equation 2.3 for the given i, j pair. It is therefore a greedy procedure, it is not guaranteed to find the separator, but our experiments show that the procedure is successful in finding most of the conditional independence relationships. This procedure is very similar to a procedure recently proposed by Brown *et al* [10], where a greedy polynomial version of the MMPC algorithm [89] is discussed. The procedure is given in Algorithm 3.

Algorithm 3 Greedy Separator Search (gss)

Input: Data D, Node i , Node j, Adjacency Matrix G,

Output: Separator set S for i, j (if any), A boolean value indicating conditional independence *vanished*

```
1:  Let $N$ be the set of neighbors of $i$ except $j$ in $G$ and $L(i,j|S) = \sqrt{n - |S| - 3}|Z(i,j|S)|$
2:  $S = \{\}$
3:  $zCur = L(i,j|S)$
4:  $vanished = FALSE$
5:  repeat
6:      Choose $k$ such that $k = argmin_{k \in N} L(i,j|S \cup \{k\})$ and $L(i,j|S \cup \{k\}) < zCur$
7:      if There is such a $k$ then
8:          $zCur = L(i,j|S \cup \{k\})$
9:          $N = N \setminus \{k\}$
10:         $S = S \cup \{k\}$
11:         if $zCur \leq cutoff$ for a given significance level $\alpha$ then
12:             $vanished = TRUE$
13:         end if
14:     end if
15: until ($vanished = TRUE$) or ($N = \{\}$) or (there is no such $k$)
16: return $S, vanished$
```

3.1.2 Estimating dense nodes

The prior knowledge on dense nodes can be obtained from various sources such as the regulatory network databases, like [59]. But, if we have the prior knowledge matrix B described above, dense nodes can also be estimated by using B.

Given prior knowledge in the form of a matrix B, dense nodes can be estimated from B by assuming that i and j are connected if B_{ij} is greater than a threshold T. Then, the following simple procedure can be used to estimate dense nodes; if the number of connections of a node i is greater than a fixed value F (which is specified in the experiments based on some initial tests), then consider node i as dense. For instance assume $T = 0.8$ and the i^{th} row of B has k entries that are larger than 0.8. If $k > F$ then node i is considered as a dense node, otherwise it is a regular node. Given B, the procedure described here is executed to estimate the set of nodes that are dense, and then this set is used as the set of dense nodes in the algorithms that require this information.

3.2 Prior knowledge in conditional independence tests

In real life, when we have a prior knowledge that we think is true with high probability, our decisions are affected by that prior knowledge. For example, if we believe that a proposition P is true, then when we observe $not(P)$, we first think that it happened by chance, *i.e.*, we need to have more evidence to be convinced that P is actually false. This is called *bias* in statistical terms.

We can map the above argument to conditional independence tests (assuming our bias about P has actually a high probability of being correct). There can be several such mappings, which we use here to modify the value of the significance level α in Eq (2.3) depending on our prior knowledge. So, we increase the value of α (*i.e.*, it is more probable to reject the null-hypothesis) if $B_{ij} > 0.5$, where $B_{ij} > 0.5$ implies that we have a prior belief that the edge E_{ij} exists in the graph, and the degree of this belief depends on the value of B_{ij}. Similarly, we decrease the value of α if $B_{ij} < 0.5$, which means we have a prior belief to some degree regarding the absence of the edge E_{ij}, depending on the value of B_{ij}.

Given the prior knowledge B, α is basically updated as $\alpha = \alpha_0 * (1 + \beta(B_{ij} - 0.5))$, where $\beta \geq 0$ is a factor that denotes our "trust" on prior knowledge, and α_0 is the initial value of α. When $\beta = 0$, the prior knowledge has no effect on the decision, and when $\beta \to \infty$, the output completely represents the prior knowledge. But, it is obviously meaningless for α to be greater than 1 or less than 0, so the actual update of α is performed as in Equation 3.1.

$$\alpha = \begin{cases} 0 & \text{if } \alpha_0 * (1 + \beta(B_{ij} - 0.5)) < 0 \\ 1 & \text{if } \alpha_0 * (1 + \beta(B_{ij} - 0.5)) > 1 \\ \alpha_0 * (1 + \beta(B_{ij} - 0.5)) & \text{otherwise} \end{cases} \tag{3.1}$$

Dynamically updating the significance level for each test provides a method to take two different information sources into account while constructing the skeleton of the underlying graph by using the PC algorithm. As can be seen in the update equation (Eq. 3.1), $B_{ij} = 0.5$ means no prior knowledge about the existence of the edge E_{ij}. That is surely an advantage for GRN modeling, since prior knowledge might not be available for all edges.

We name the proposed algorithm as PCPr, where the extension indicates that the algorithm uses the given prior knowledge in the conditional independence tests as described above. The significance of this extension is evident by the supporting test results reported in Chapter 5.

3.3 Related Work

Various methods have been proposed in the literature for modeling GRNs. Generally the methods apply well-known concepts in machine learning including Bayesian networks [6, 29, 61, 86], (probabilistic) Boolean networks [3, 51, 78, 67], neural networks [92, 98], Markov chains [48], differential equations [13, 19], s-systems [87] and hybrid systems [47].

Bayesian networks are used in various studies for gene regulatory network induction. Since the expression data has a high rate of missing values, BNs are good candidates for modeling because of the BNs' ability to handle missing data. But scalability is an issue that deriving BNs without some restrictions is NP-hard [15]. But the gene regulation is suitable for this restriction since biological studies show that a gene is regulated by a number of genes that is generally not larger than 5. In addition to this, the probabilistic nature of gene regulation makes BNs a remarkable framework for modeling. Besides the studies that are just applications of the BN learning to gene expression data, there are also some studies where other biological information are also used in deriving the model [6, 35, 86].

A method is suggested in [6] to combine the gene expression data and transcription factor binding location data to derive a dynamic Bayesian network that better represents the regulation. Location data is used as the structure prior and expression data is used as the likelihood. Location data is reported as a p-value which is inversely related to an edge being present in the structure. They define p-value corresponding to edge E_i in terms of a random variable $P_i \in [0, 1]$ that is distributed as $P_\lambda(P_i = p|E_i \in S) = \lambda e^{-\lambda p}/(1 - e^{-\lambda p})$, where λ is the parameter of exponential distribution. So, if $P(E_i \in S) = \beta$, then after marginalizing over λ and assuming $\lambda \in [\lambda_L, \lambda_H]$, the value of P can be computed as:

$$\begin{array}{c} P(E_i \in S|P_i = p) = \\ \frac{1}{\lambda_H - \lambda_L} \int_{\lambda_L}^{\lambda_H} \frac{\lambda e^{-\lambda p} \beta}{\lambda e^{-\lambda p} \beta + (1 - e^{-\lambda})(1 - \beta)} \end{array} \tag{3.2}$$

Since the integral above can not be solved analytically, they solved it for fixed values of p and stored the result for later usage. Then the prior for a structure S is given as:

$$\begin{aligned} logP(S) = \quad & \textstyle\sum_{E_j \in S} logP(E_j \in S|P_j = p) + \\ & \textstyle\sum_{E_k \notin S} logP(E_k \notin S|P_k = p) \end{aligned} \tag{3.3}$$

When this prior is used, the error is significantly reduced. Also the results on data of Spellman et al. [81] is given and shows some interesting relationships between genes that can not be derived without priors.

Tamada et al. [86], describe a way to combine gene expression data and evolutionary information to derive continuous Bayesian networks from data. The evolutionary information is given as gene pairs set H_{AB} for two organisms A and B, which are derived using BLAST [41]. BLAST gives gene pairs that seem to be related in two different organisms, where such genes are called orthologous genes. So the idea is to use this information so that if genes a, b and

c, d are in H_{AB}, where a, c is from organism A and b, d is from organism B, and if there is a regulatory relationship between a and c in A, then it is highly probable that there is a regulatory relationship between b and d in B. They give a score based on this idea which is, (Gene Network Score)

$$\begin{aligned} GNS(G_A, G_B) = \quad & logP(X_A|G_A)P(X_B|G_B) \\ & P(H_{AB}|G_A, G_B)P(G_A)P(G_B) \end{aligned} \tag{3.4}$$

where $X_A(X_B)$ is the microarray expression data of organism $A(B)$. They start with networks G_A and G_B that is found using traditional Bayesian network search algorithms and use these as their initial networks. Then they continue with a greedy hill-climbing algorithm that in each step adds or removes an edge from one of the networks so as to increase $GNS(G_A, G_B)$.

By using only the expression data, in [61], the authors derive a BN from time series expression data of Spellman *et al.* [81]. They express the transcription rate of a gene as;

$$g(H : \beta, \gamma) = \beta \frac{\gamma H}{1 + \gamma H} \tag{3.5}$$

where H is concentration of the active regulatory protein, β is the maximum transcription rate that the gene can achieve and γ is the ratio of association and disassociation constants of the regulators to the promoter regions.

From expression data, the transcription rates are derived based on a known regulation diagram G (which is a Bayesian network) and the parameters h, θ that maximize the likelihood:

$$l(h, \theta : G, E) = logP(E, h|\theta, G) \tag{3.6}$$

where h are the values of the unobserved regulator activity levels at different times and θ is the vector of other parameters $(\gamma, \beta, etc.)$ are calculated. Then the authors describe a structural EM algorithm [28] that iteratively derives a regulation diagram which uses a Bayesian network scoring function as;

$$score(G : E) = max_{h,\theta}(h, \theta : G, E) - \frac{N_{param}}{2} log(T) \tag{3.7}$$

where N_{param} is the number of parameters in the model and T is the number of time points. The algorithm can add regulators to genes and also add new regulators that are not in the regulator set H.

In [29], discovering the interactions between genes from expression data by using BNs are investigated. Multinomial (discrete) and linear Gaussian (continuous) Bayesian networks are derived using sparse candidate algorithm. To understand whether the algorithm derives reasonable networks, two kinds of features which are the Markov property and order property (the partial order of nodes in the network) are defined. A confidence value is defined based on an algorithm that checks in what percentage the above features are observed in m new networks derived from the perturbed data.

Boolean networks are one of the widely studied methods for modeling gene regulatory networks. Since BoNs are simple to understand and polynomial time deducible for a bounded indegree from data, they received much attention in this context. Also it was proved that if indegree of each node is bounded by a constant, then only $O(logn)$ input output pairs are necessary and sufficient to derive the correct BoN [3]. But since they only work with binary data, while modeling, information loss seems unavoidable. In gene network modeling, the value of a node being 1 means that the gene is expressed and being 0 means the gene is not expressed.

In a recent study [67], the authors describe a way to construct PBNs based on the fact that if we are sampling the data from steady state, to check validity of a designed network we have to check whether the steady state mass lies in the observed sample states. They give an algorithm that first selects k attractor sets randomly, then they pick a predictor set for each gene again randomly and then check for compatibility with the attractor set by using the fact that an attractor introduces a cycle. Then other entries of the truth table of genes and predictors are filled randomly and checked for a cycle; if a cycle is found random, filling is performed again. This algorithm generates a BoN. A PBN is generated from multiple runs of this algorithm and assignment of a probability of switch of BoNs and a probability of perturbation. An application of this algorithm to gene expression data is given at the end. They generated 10000 PBNs and chose the one that minimizes the mean-squared error between data frequency of attractors and estimated steady-state distribution of each attractor based on the size of the tree corresponding to an attractor. The results are given as a histogram that shows how close the distribution of attractor states in data and the time spent in attractor states after running the designed PBN for a long time.

The paper by Weaver *et al.* [92] is one of the first attempts to derive the regulatory networks from expression data. The authors use an approach that they call weight matrices which is in fact a neural network. They assume the regulatory behavior of genes can be modeled by a number of linear functions whose input values are expression levels of other genes in the previous time step. First they describe the details of the model and how to produce expression data from the model. Given a weight matrix Z and the input $u(t)$ of expression levels, the next state of the system is given as:

$$u(t+1) = mg(Zu(t)) \tag{3.8}$$

where g is a normalization function that maps the expression levels to (0-1) interval and m is the maximum expression levels of genes. A reverse engineering approach is then given based on the data produced artificially using the above equations and randomly producing Z and m. The reverse engineering algorithm includes solving algebraic equations that comes from solving $u(t)$ from $u(t+1)$ based on the assumption that the matrix Z will be mostly composed of zeros. The results are successful in the sense that the reverse engineering approach derives the original network even in the presence of noise. But the authors give no results on real expression data.

Recurrent Neural Networks are used to model gene regulatory networks in [98]. Each gene is represented as a node and each edge gives the weight of influence of a gene on another. Instead of backpropagation the authors use Particle Swarm Optimization (PSO) for parameter learning. PSO is a search method similar in some sense to genetic algorithms (GAs). It is based on a number of particles (solutions) that are walking on the search space. The particles are accelerated towards the direction which is a combination of the direction of the previous best solution of the particle itself and the global best solution of all particles. The quality of the solution like in GAs is calculated by using a fitness function. Results are given on both an artificial data set and a real data set. Because of the amount of data the results are better on the artificial data set. Although some meaningful relationships are captured on the real one, it is not a suitable method for larger networks.

The work in [48] proposes a method to construct Markov Chains (MCs) to simulate the behavior of biological gene regulation. They select 10 genes from the set of 587 gene data set. For each of the 10 genes they select 3 genes from the data by using highest coefficient of determination value for a target gene. The MC transition probabilities are derived empirically from the data in the form of:

$$P(g_l^{t+1}) = P(a|g_i^t, g_j^t, g_k^t) \tag{3.9}$$

where $a = \{0, -1, 1\}$ and i, j, k are the 3 genes mentioned above, and l is the target gene. They perform a simulation of this MC which also includes a probability of perturbation that changes the expression value of a gene randomly. After the simulations, the authors show that the states of the MC in the simulation very much resemble the biological data.

In addition to the above mentioned methods, differential equations [13, 19], s-systems [87] and hybrids of the above methods [47] are also used in modeling. Differential equations typically model regulation as a set of rate equations of the form $dx_i/dt = f(\mathbf{x})$ where x_i is the gene we are trying to model and f is the function that will be searched and $\mathbf{x}$ is a set of variables that is thought to be effecting the expression level of x_i. S-systems are a type of power-law formalism that can be described by a set of non-linear differential equations. But it has a problem that it requires the estimation of a large number of parameters.

Hybrid methods are the ones that use a mixture of the above methods. Inferring a regulatory network model by using GAs and neural networks is described in [47]. The IDs of the genes are considered as the chromosomes of the GA and a single layer neural network is trained for the fitness function of the GA. The root mean square error of the trained neural network is used as the fitness function for GA. The best chromosome is chosen as the regulators of gene j. Then j is increased and the algorithm finds the regulators for all output genes. Finally, the neural network of the best chromosome for all output genes is used as the predictor for the next time step. The results are given for 3 different settings. The first one is a randomly generated data from a known network; the second is the rate of spinal cord data; and the third is cell cycle data of Spellman *et al.* [81].

Besides these, a graphical modeling paradigm, which is generally named as Gaussian Graphical Modeling (GGM) [52], has received considerable attention [11, 20, 42, 58, 71, 75, 95, 97]. In this paradigm, it is assumed that the data constitute a random sample from a multivariate normal distribution. Generally, the focus here is to compute (or approximate) the covariance matrix in the existence of small number of data samples compared to the large number of genes. The solutions proposed include approximately computing the covariance matrix by shrinkage estimators [75] or decomposition [42] and low-order conditional independence graphs [11, 20, 58, 95]. Graph decomposition techniques were also previously integrated with GGM and PC algorithm [97]. Recently, an information theoretic approach was proposed for using low order partial correlations as a measure of conditional independence [71]. Also importance of conditional correlation has recently been studied in reverse engineering regulatory networks [103].

Due to the small sample sizes of biological data, methods of combining multiple types of biological data have recently been developed [12, 54, 55, 105]. These studies generally propose to combine gene expression, TF binding location and TF binding motif data. All the results show that combining multiple data types lead for better identification of better gene associations/clusters/networks compared to using a single data type. These studies constitute supporting evidence for the motivation of the algorithms proposed in this chapter as well.

3.4 The Gap Covered by the Proposed Methods

The algorithms that are discussed above to derive a GRN (and structure learning algorithms in general) suffer from one or more of the following; scalability, small sample sizes and densely connected nodes. Constraint-based learning algorithms scale well in general, which makes them a strong candidate to apply for GRNs. Although there are some previous work to deal with small sample sizes, to the best of our knowledge, this is the first study that tries to handle nodes with large number of connections in a structure learning context. The PC algorithm is known to work well for sparse graphs. But the algorithm may fail to learn a good graph when either the sample size is small or the underlying graph has some dense nodes. Unfortunately, biological data and GRNs have both of these properties.

To overcome the above problems and derive GRNs from biological data by using the PC algorithm, we propose two modifications to the PC algorithm. For the first modification, we argue that integrating multiple available biological data types in learning a GRN should be helpful. We integrated TF binding location (ChIP-chip) and microarray gene expression data through statistical independence tests of the PC algorithm. As depicted in the results, this greatly improves the performance in learning. In addition to the PC algorithm, this method can also be used in other constraint-based structure learning algorithms as well. Also, other than TF binding location data, any biological data that can be converted into probabilities of edges being present, can be used as the prior knowledge. For instance, by using TF binding motif data in the form of position weight matrices one can compute the probability of existance of a binding region for a given TF on a gene's promoter regions [50].

The second modification is related to the nodes that have a large number of connections. As these constitute a problem for most of the learning algorithms, we propose a method to process them differently from the normal nodes in a graph. This method improves the performance by preventing the order in the PC algorithm to increase to large numbers that can cause errors, as a large order decreases the power of statistical independence tests. By identifying dense nodes from prior information obtained from another type of biological data, the new algorithm named PCPDPr, outputs a better network than the PC algorithm and has the potential to be improved further.

CHAPTER 4

LARGE-SCALE GENE REGULATORY NETWORK CONTROL

Controlling GRNs is an important and hard problem. As it is the case in all control problems, curse of dimensionality is the main issue in real applications. It is possible that hundreds of genes may regulate one biological activity in an organism; for instance a set of approximately 800 genes were previously estimated to be cell-cycle related in budding yeast, *Saccharomyces cerevisiae* [81]. As the state space for a GRN will be exponentially large in the number of genes, number of states for the cell-cycle model of yeast including these genes will be enormous. Although it may be possible to maintain (learn, keep, etc.) such a model, it is not possible for most of the current control algorithms to solve a control problem of this size. This is also evident in the literature that only models of small portions of the genome of a living organism could be used in control applications. Following the discussion in the previous chapter, as larger models become available, scalable control algorithms will be necessary for the analysis of these networks for interventions.

This chapter includes the description of two methods that are aimed at the scalability for control in GRNs. Given a PBN model, the methods in this chapter try to reduce the model to a "simpler" PBN model so that the control problem is easier to solve. However, this simplification, obviously, makes sense if the solution of the reduced problem is a good approximate solution to the original problem. To achieve this, first, in Section 4.1, we describe the method that can be named as a feature reduction method which tries to identify and eliminate the genes that are irrelevant for the control problem. This way, the model (and so the state space) gets smaller and becomes easier to solve by MDP solvers. But instead of eliminating a gene completely from the model, some connections of this gene in the model can be removed as well to reach a simpler model, and the resulting model may be solved more efficiently by a FMDP solver. The second method which is described in Section 4.2 is built on this idea; simplifying the model by eliminating edges, instead of genes, in the model. In addition to the proposed methods, this chapter also includes in Section 4.3 the previous work done in the domain of GRN control. We close this chapter by explicitly stating the contributions of this chapter and the gap covered by these algorithms.

4.1 Scalable Control by Feature Reduction

Feature reduction is the process of finding and excluding from further consideration, features that are expected to have reasonably negligible or minimal effect on the output quality. In general, feature reduction or feature selection is performed to improve the performance of some predictors [33]. The features in the case of gene expression data are the genes, the samples, or both. Here, we consider feature reduction as decreasing the number of genes. What we consider as output is the value function found by solving the MDP. This is reasonable as the value function represents the reward (or cost) associated with a given state by applying the control policy. In real life, this can give an indication of the cost of treatment (policy). The speed-up gained by reducing the state space of the MDP is considered as performance improvement.

$$\begin{tikzcd}
\boxed{\mathit{Data}} \arrow[r, "a"] \arrow[d, swap, "b"] \arrow[d, "\text{Feature Reduction}"] & \boxed{\mathit{Model}} \arrow[r, "a"] & \boxed{\mathit{MDP}(M)} \arrow[r, "a"] & \boxed{\mathit{Policy}(\pi)} \\
\boxed{\mathit{Data}'} \arrow[r, swap, "b"] & \boxed{\mathit{Model}'} \arrow[r, swap, "b"] & \boxed{\mathit{MDP}(M')} \arrow[r, swap, "b"] & \boxed{\mathit{Policy}(\pi')} \arrow[u, "F", swap] \arrow[u, "b"]
\end{tikzcd}$$

Figure 4.1: Finding control policy for the given data

In the GRN control domain, we observed that some of the genes can be ignored in the process of finding a control policy for a given data. So, it is essential to estimate the redundancy in the data before starting the modeling and MDP solving parts. This way, we can effectively deal with GRNs that have larger number of genes by applying reduction to obtain a smaller set of genes instead. This is depicted in Figure 4.1, where path (a) is the ordinary path of solving the control problem and path (b) is the reduction based method proposed to solve the problem. Function F that labels the last link along path (b) in Figure 4.1 maps policy π' found for MDP M' to policy π of the larger MDP M. Assume the *Model* on path (a) has two states s_i and s_j which only differ in the value of one gene, say the third gene. For example, for the binary case let these states be $s_i = 1001010$ and $s_j = 1001110$, for a network of seven genes. If, in the feature reduction step of path (b), we decide that the third gene is irrelevant, then s_i and s_j will be aggregated in *Model*$'$, forming a state $s_n = 100110$. To get policy π, after solving M', we have to remap the action defined for s_n in π' to s_i and s_j. Since we know that the third gene is redundant, s_i and s_j are in fact equivalent states. Therefore, F simply gets π' and produces π by setting $\pi(s_i) = \pi(s_j) = \pi'(s_n)$, for all s_i, s_j and s_n.

The proposed solution is based on the assumption that the objective is defined in terms of the expression values of some genes, namely the genes that we want to control. We require that the reward function is defined in the form $R(s, a, s')$ and depends only on the action and next state. This does not need to be the case in real life, but it also does not overly restrict the problem because the objective is generally defined in terms of desirable or undesirable states. Finally, model minimization can also be performed after building the model. However, if we think of the process as finding a policy for some given data, then feature (or gene) reduction before modeling saves time in the modeling stage because model building is a time-consuming task as well.

4.1.1 Selecting the genes to remove

The selection of the gene to remove is based on the following observation: Since the objective is defined in terms of reward and control genes, all other genes are candidates for removal. From the set of candidate genes, a subset will be selected based on their estimated relevance for deriving a control policy.

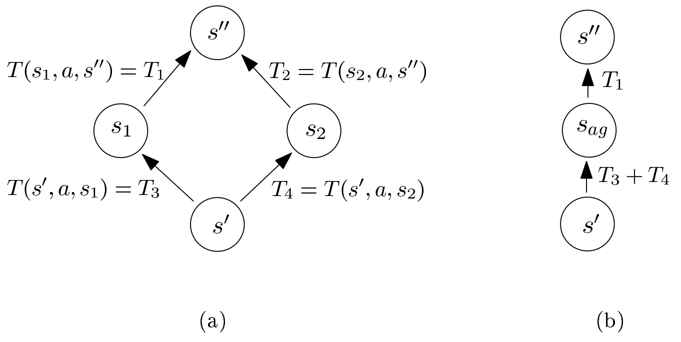


Figure 4.2: Aggregation of stochastically bisimilar states s_1 and s_2

Recall that genes to be removed should have the lowest effect on the value function. As explained above, removing a gene, say g from consideration is equivalent to aggregating the states that differ only in the value of g. Assume that s', s_1, s_2 and s'' are related in the MDP as shown in Figure 4.2(a). Both s_1 and s_2 have to be stochastically bisimilar [30] in order to be aggregated so that the resulting MDP has the same solution as the original MDP.

Definition 1. [30] Any two states s_i and s_j in an MDP are said to be stochastically bisimilar if the following two conditions hold:

I. $\forall a \;\; R(s_i, a) = R(s_j, a)$

II. $\forall a, s' \;\; T(s_i, a, s') = T(s_j, a, s')$ •

Stochastic bisimilarity for the states of an MDP is an equivalence relation (see Theorem 4 in [30] for more details). Two stochastically bisimilar states have the same value in the solution for an MDP. Under this equivalence relation, stochastically bisimilar states are said to be equivalent in an MDP; by using this information, the MDP can be reduced to another MDP with a smaller state space.

Theorem 1. [30] Two stochastically bisimilar states in an MDP are equivalent and can be aggregated. •

Proof. Follows from Theorem 7 in [30]. □

The aggregation procedure of the two stochastically bisimilar states s_1 and s_2 from Figure 4.2(a) leads to the new state s_{ag} shown in Figure 4.2(b); it has the same reward function as s_1 and s_2, i.e., $\forall a, s' \;\; R(s_{ag}, a, s') = R(s_1, a, s') = R(s_2, a, s')$, and the transition probabilities are as shown in the figure[1].

Consider states s_i and s_j that only differ in the value of g in M (see Figure 4.1). According to Theorem 1, if the states s_i and s_j are stochastically bisimilar and there is in M' a state s_{ag} as the aggregation of s_i and s_j, then M' is the minimized version of M, and hence has the same solution as M. This may be interpreted as follows, we can find the control policy faster by locating and removing from the data every gene g for which cases I and II in Definition 1 hold for the states that only differ in the value of g.

For case *I*, we will use the assumption in the definition of $R(s, a, s')$ that it does not depend on the current state s. Assume we have one reward gene g_r that we want to control. The reward function $R(s, a)$ by definition satisfies:

$$R(s,a) = \sum_{s'} T(s,a,s')R(s,a,s') \tag{4.1}$$

and by using our assumption on the reward function, it can be rewritten as:

$$R(s,a) = \sum_{i \in Val(g_r)} \sum_{s' \in S_{g_r=i}} T(s,a,s')R(s,a,s') \tag{4.2}$$

where $Val(g_r)$ denote the discrete values that g_r can take, and $S_{g_r=i}$ denote the set of all states that satisfy $g_r = i$. Notice that $R(s, a, s')$ is constant for all s' (where $g_r = i$) and a given action a (recall the assumption about $R(s, a, s')$). This means that case I in Definition 1 holds for two different states s_i and s_j if,

$$\forall i \sum_{s' \in S_{g_r=i}} T(s_i,a,s') = \sum_{s' \in S_{g_r=i}} T(s_j,a,s') \tag{4.3}$$

Eq (4.3) may be interpreted as follows: being in state s_i or state s_j makes no difference about the value of g_r in s'. If s_i and s_j differ only in the value of a gene, say g, then Eq (4.3) holds if the probability of g_r taking value i in s' is independent of the value of g,

[1]Note that Givan *et al.* [30] call s_{ag} a block (set of states) rather than a new state, but there is conceptually no difference for our case.

i.e., $Pr(g_r(t+1) = k|g(t)) = Pr(g_r(t+1) = k)$, where $g(t)$ denotes the value of g at time step t. In other words, case I holds if g has no influence on the next state value of g_r. Using similar argument, for case II to hold for states s_i and s_j that differ only in the value of g, gene g should have a low effect on determining the next state of any gene. So, we have to check similar conditions for cases I and II. If we approximate the influence of a gene on a set of genes as the average influence, both the influence of g on g_r and the average influence of g on all other genes are important.

Influence Score

Given two genes g_i and g_j, the influence of g_i on g_j can be estimated by checking to what degree the equation $Pr(g_j(t+1) = k|g_i(t)) = Pr(g_j(t+1) = k)$ is satisfied. We define the following function to estimate the influence of g_i on g_j:

$$Inf(g_i, g_j) = \sum_{k \in Val(g_j)} |Pr(g_j(t+1) = k|g_i(t)) - Pr(g_j(t+1) = k)| \tag{4.4}$$

and we define the average influence of g on a set of genes G as:

$$AvgInf(g, G) = \frac{1}{|G|} \sum_{g_G \in G} Inf(g, g_G) \tag{4.5}$$

where $|G|$ is the number of genes in G. The counts for the different values of pairs (g_i, g_j) in the data constitute sufficient statistics for $Inf(g_i, g_j)$.

Note that the function $Inf(g_i, g_j)$ that gives the influence of gene g_i on g_j is similar in nature to the influence concept introduced by Shmulevich *et al.* [78]. But, Shmulevich *et al.* [78] compute this value based on the model (Probabilistic Boolean Network), while we compute the value of $Inf(g_i, g_j)$ directly from the data without building a model.

To select a subset from the genes in the data, we assign to each gene what we call *Influence Score (IS)*, which is based on two sub-scores inspired for the cases in Definition 1. The sub-score for case I is:

$$S_I(g) = Inf(g, g_r) \tag{4.6}$$

where g_r is the reward gene. The sub-score for case II is:

$$S_{II}(g) = AvgInf(g, G) \tag{4.7}$$

where G includes all genes in the data except g. As a result,

$$IS(g) = S_I(g) + S_{II}(g) \tag{4.8}$$

Combining all the already introduced concepts, the final reduction method that we call *FRGC* (Feature Reduction for GRN Control) is given in Algorithm 4.

Algorithm 4 FRGC

```
Input: $m \times n$ discrete gene expression data ($D$) and threshold $Th$
Output: $(m-k) \times n$ reduced gene expression data ($D'$)
  $number\_of\_genes \times number\_of\_samples = size(D)$
  $Genes = \{1, \ldots, number\_of\_genes\}$
  $IrrelevantGenes = \{\}$
  for all $g \in Genes$ do
      Compute $IS(g)$
      if $IS(g) < Th$ then
          $IrrelevantGenes = g \cup IrrelevantGenes$
      end if
  end for
  $D'$ = Remove $IrrelevantGenes$ from $D$
  return $D'$
```

FRGC identifies and removes some of the lowest scored gene(s). One point to consider in the process is the number of lowest scored genes to remove. We use a threshold score Th as the stopping criteria of the removal; Th obviously depends on the analyzed expression data. In this thesis, we rely on domain expert to specify the value of Th.

Deciding on a value for the threshold is a subjective process which depends on several issues, like the usage of the results, the degree of accuracy, simplicity of the policy, computational resources and the objective. A large (small) threshold implies less (more) accurate results and requires less (more) computational resources. The expected complexity of the resulting policy can also be important because eliminating some of the genes would generally produce a simpler policy which is more attractive as the applicability is concerned. The need to take immediate action for time critical cases may tolerate lower accuracy for simpler policy. All these factors are good indicators to guide the choice of a threshold value. The process is subjective; it is like a multi-objective optimization issue because most of the factors and objectives described above do conflict. So, it is the duty of the domain expert to decide on which factors or objectives should be considered more important to the specific problem being investigated and hence set the threshold value accordingly. For instance, a lower threshold value is preferred if sensitivity is the issue, while a higher threshold value is expected if simplicity is the major concern; most of the cases it is somewhere in between. Finally, while deciding on the threshold value it is possible to employ simple procedures such as investigating the sum of IS scores for all genes and eliminating the genes with smallest scores up to a certain percentage of the sum; however, a decision on the percentage is necessary and this is again subjective. The problem can be also considered as determining the number of minimum scored genes to eliminate; this depends on the set of genes being investigated by the experimenter. Keeping all these issues in mind and knowing how subjective the process is, finding a method that minimizes as much as possible the involvement of the domain expert in a way to automatically determine the threshold value is a challenging task; once put in action such a method will add much to the value of the reduction approach proposed in this work, turning it into a more adaptable process.

As described in the experiments section, errors are computed as the percentage difference between the resulting value function and the approximate value function. How much error is tolerable highly depends on the problem specification. For instance, if the reward function is defined as the financial cost of a treatment, then the tolerable error bound can be large compared to the case when the reward function is defined in terms of life expectancy or probability of survival. For instance, a 10% error would be acceptable as financial cost is concerned, i.e., it may be tolerable for some cases. But if the survival of a patient is the concern then 10% error should not be acceptable unless it is the best available alternative. Therefore, deciding whether an error bound is tolerable or not is problem dependent.

4.2 Scalable Control by Edge Elimination from Factored Representations

Control problems can also be solved by using factored representations (see Section 2.3.1). In GRN domain, this factorization naturally occurs as each gene corresponds to a factor. In this section, we describe the application of FMDP framework to GRN control problem and propose a method to reduce the GRN model to a simpler one so that the solution can be found easier. The genes/nodes in a factored model that have no effect on the reward gene(s) do not exist in the solution of the control problem [9]. This situation gives rise to the following questions; What about the genes that have small or negligible effect on the reward genes? Can the connections of these genes be eliminated from consideration in solving the control problem? This section discusses the methods proposed based on this idea.

Although factored representations help in solving some of the problems, they still suffer from the curse of dimensionality in the worst case [9]. Fortunately, in most of these cases we can still reach a reasonable approximate solution by pruning and/or approximating the value tree.

Most of the approximate methods prune the constructed trees during the process of solving FMDP. Another possibility in finding an approximate result is to prune the transition model before solving the problem. In this section, we elaborate on such a method, but before that we introduce the concept of *edge influence*.

4.2.1 Edge Influence

We start by introducing the basic concepts required to understand edge influence as in Definition 2. For a PBN, given gene g_i and its parent genes P_i, Shmulevich *et al.* [78], formalized *Influence* to measure the effect of a parent on g_i. Influence of g_i on g_j is the probability that the next state value of g_j will change when we change the value of g_i at the current time step. To formally define the influence of g_i on g_j, denoted $I_i(g_j)$, first we have to define the influence of a gene with respect to a boolean function f, which is the probability that the output of f will change if we change g_i. Assume that f is defined on the set of input genes $P = (g_1, ..., g_k)$. The influence $I_j(f)$ of g_j on f is defined as;

$$I_j(f) = Pr\{f(g_1, ..g_{j-1}, 0, g_{j+1}, .., g_k) \oplus f(g_1, ..g_{j-1}, 1, g_{j+1}, .., g_k)\} \tag{4.9}$$

where $\oplus$ stands for exclusive *OR*. Equation 4.9 depicts the probability that f will output a different value if g_j is toggled while the other input variables are kept unchanged. Also note that $I_j(f) = 0$ if $g_j \notin P$.

Given V, F and $c_j^{(i)}$ of a PBN, $I_i(g_j)$ is defined as follows [78]:

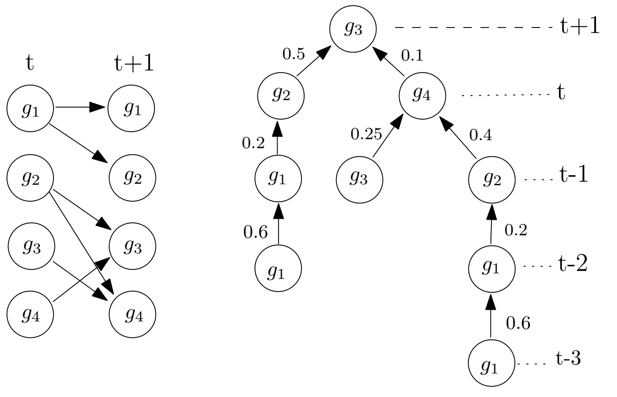
$$I_i(g_j) = \sum_{k=1}^{l_j} I_i(f_k^{(j)})c_k^{(j)} \tag{4.10}$$

which is the weighted sum of all influences of g_i on the set of functions F_j. Refer to [79] for further details of influence concept.

Given a PBN, influence of a gene g_i on gene g_j, $I_i(g_j)$, can be interpreted as a measure of the strength of the link between the two genes. But, $I_i(g_j)$ will be zero if g_i is not among the parents of g_j. However, this does not mean g_i has no influence on g_j. This becomes more clear if we consider the PBN in Figure 4.3(a) and its "unrolled" version for g_3 in Figure 4.3(b). As depicted in the "unrolled" PBN, considering the future effects of each gene, it is obvious that each gene has more to influence than only its children. If there is a path from g_i to g_j in the unrolled PBN, then g_i at time step t has influence on the value of g_j at time step $t + k$, where k is the length of the shortest path between g_i and g_j in the unrolled PBN.

Given a node g_i as the root, an unrolled PBN is constructed (as a tree) by expanding each node g at level t with the parents of g at level $t - 1$ in the given PBN. Nodes are expanded unless the unique path from the leaf node to g_i includes a cycle. For instance, leaf node g_3 is not expanded in Figure 4.3(b) because the path from leaf g_3 to the root includes g_3 twice. Also notice that there can be links like $g_1 \to g_1$ as g_1 has itself as the parent in the PBN, and g_1's at the leaves are not expanded because the two paths $g_1 \to g_2 \to g_3$ and $g_1 \to g_2 \to g_4 \to g_3$ include g_1.

Recall that each $I_i(g_j)$ corresponds to the effect of the value of g_i at time step t on the value of g_j at time step $t + 1$, and assume g_k is one of the parents of g_i. When we unroll the PBN one time step, we will observe the path $g_k \to g_i \to g_j$. We know that, based on Markovian



(a) PBN

(b)Unrolled PBN for g_3

$I_1^{(*)}(g_3)$	0.1080
$I_2^{(*)}(g_3)$	0.54
$I_3^{(*)}(g_3)$	1
$I_4^{(*)}(g_3)$	0.1
$EI_{g_1,g_1}(g_3)$	0.0648
$EI_{g_1,g_2}(g_3)$	0.1080
$EI_{g_2,g_3}(g_3)$	0.5
$EI_{g_3,g_4}(g_3)$	0.025

(c) Influences corresponding to all paths

Figure 4.3: Unrolling a PBN

property, $I_k(g_i)$ is independent of $I_i(g_j)$. Therefore, to compute the influence of g_k on g_j after two time steps, we simply multiply $I_k(g_i)$ and $I_i(g_j)$.

Formally, consider a simple path (a path that does not have any cycles) $p = g_{i_1}, g_{i_2}, ..., g_{i_k}$ between g_{i_1} and g_{i_k} in an unrolled PBN; if we label each edge $E_{g_{i_l}g_{i_{l+1}}}$ on the path with $I_{i_l}(g_{i_{l+1}})$, we can define the influence of g_{i_1} on g_{i_k} corresponding to path p, denoted $I_{i_1}^{(p)}(g_{i_k})$, as: $I_{i_1}^{(p)}(g_{i_k}) = \prod_{l=1}^{k-1} I_{i_l}(g_{i_{l+1}})$.

Note that there can be more than one simple path from one gene to another in an unrolled PBN. Let P be the set of all simple paths from g_i to g_j in an unrolled PBN. We define *all-path influence* of g_i on g_j, denoted $I_i^{(*)}(g_j)$, as:

$$I_i^{(*)}(g_j) = \begin{cases} \sum_{p\in P} I_i^{(p)}(g_j) & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \tag{4.11}$$

Figure 4.3(b) shows $I_i(g_j)$ values as labels on the edges; $I_2(g_3) = 0.5$, $I_3(g_4) = 0.25$, etc. From those, we can easily compute $I_i^{(*)}(g_j)$ values. To compute $I_1^{(*)}(g_3)$ for instance, we consider the two simple paths from g_1 to g_3 in the unrolled PBN, which are $p_1 = g_1 \to g_2 \to g_3$ and $p_2 = g_1 \to g_2 \to g_4 \to g_3$. For the first path $I_1^{(p_1)}(g_3) = 0.2 * 0.5 = 0.1$, and for the second path, $I_1^{(p_2)}(g_3) = 0.2 * 0.4 * 0.1 = 0.008$. So, $I_1^{(*)}(g_3) = I_1^{(p_1)}(g_3) + I_1^{(p_2)}(g_3) = 0.1080$. The other $I^{(*)}(g_3)$ values can be computed similarly based on Equation 4.11 and they are given in Figure 4.3(c).

Definition 2 (Edge Influence)**.** *Given three genes g_i, g_j and g_k, Edge Influence (EI) of the edge between g_i and g_j on g_k is defined as: $EI_{i,j}(g_k) = I_i(g_j)I_j^{(*)}(g_k)$.*

EI can also be computed on a set of genes, denoted EIS: $EIS_{i,j}(S) = \sum_{g_k\in S} EI_{i,j}(g_k)$, which is simply the sum of influences of an edge on all genes in the given set.

Considering nodes with same label as different nodes, the unrolled PBN corresponds to a tree. By considering them as the same node and aggregating them, we find a graph; then, path search in a tree turns into path search in a graph. Computing all paths between two nodes in a graph is a hard problem. It is NP-complete as it includes the solution of the longest path problem which is known to be NP-complete [45]. Also, the size of the unrolled PBN tree can grow exponentially large depending on the structure of PBN and the number of genes. So, it is better to compute approximate values for EI. One possible method is to prune the unrolled PBN tree. Also notice that the unrolled PBN tree for a given gene only includes relevant genes and edges. All nodes in a tree have an influence on the given gene, so the parts of the PBN that are not related to the solution of the control problem are not expanded and the EI values for those edges are not computed.

Approximate computation of EI

Limiting the size of the unrolled PBN up to a certain level can give good results. But a better method is to prune the unrolled PBN if $I_i^{(*)}(g_j)$ is less than a certain threshold T. As $I_i(g_j)$ is actually a probability value, $I_i^{(p)}(g_j)$ for any i, j and p monotonically decreases with each new level in the unrolled PBN. So, when we consider that the edge influence values below T are not significant then we may stop expanding a node i further down in case $EI_{i,j}(g_k) \leq T$, where g_k is the root.

After this approximation, we are ready to construct an approximate algorithm for computing EI values for a given gene g_j which will be the root of the unrolled PBN tree. The complete process is given in Algorithm 5. It is a recursive algorithm that actually does a limited depth first traversal of the unrolled PBN tree (in the reverse direction of the arcs shown in Figure 4.3(b)), and does not expand node i for sufficiently small values of $I_i^{(*)}(g_j)$.

4.2.2 Edge elimination for approximate solutions of FMDPs

According to Definition 2, the EI value is a measure of how a certain gene is effected by the changes in values of other genes. In FMDP, the solution includes genes that have some effect on the reward genes. So, genes that have no effect on the reward genes at any time in the future can be eliminated from FMDP. However, based on the study described in this section, we realized that instead of eliminating a gene completely (as done in Section 4.1), removing some of the unimportant edges from a DBN in FMDP may produce better results.

Given the set of reward genes Γ of FMDP, $EIS_{i,j}(\Gamma)$ denotes how each relevant edge in the FMDP influences the set of reward genes. This influence can be very low such that some of these edges can be neglected from the model. This means that edges with low EIS values can be eliminated from consideration. So, given a threshold δ, the edges with the smallest EIS values whose total EIS do not exceed δ are removed. As a final step, *reduceFMDP* performs

Algorithm 5 *computeEI*$(g, p, I_g^{(p)}(g_t), T, g_t, pbn, EI)$

```
Input: gene $g$, path $p = g_t, p_{i_1}, .., p_{i_k}$, path influence $I_g^{(p)}(g_t)$, target gene $g_t$, PBN $pbn$, initial values
    of $EI$
Output: $\forall g_i, g_j$ reachable from $g_t$, $EI_{(g_i,g_j)}(g_t)$
    if $g \in p$ then
        $EI_{g,p_{i_k}}(g_t) = EI_{g,p_{i_k}}(g_t) + I_g(p_{i_k}) * I_{p_{i_k}}^{(p)}(g_t)$
    else
        if $EI_{g,p_{i_k}}(g_t) > T$ then
            for every $p_g \in parents(g)$ in $pbn$ do
                $EI = computeEI(p_g, \{p, g\}, I_g(p_{i_k}) * I_{p_{i_k}}^{(p)}(g_t), T, g_t, pbn, EI)$
            end for
        end if
    end if
    return $EI$
```

a maximum-likelihood learning of the parameters of the DBN using data sampled from the original model.

Algorithm 6 *reduceFMDP*$(EIS(\Gamma), \delta, M, D)$

```
Input: $EIS(\Gamma)$,$\delta$,FMDP $M$,$D$
Output: FMDP $\hat{M}$
    $\hat{M} = M$
    Let $S$ be the sorted set of edges $E_{i,j}$ where $EIS_{i,j}(\Gamma) \neq 0$
    Take the first $k$ edges,$S_k$, from $S$ such that $\sum_{E_{i,j} \in S_k} EIS_{i,j}(\Gamma) < \delta$
    for all $E_{i,j} \in S_k$ do
        Remove edge $E_{i,j}$ from $DBN$s for all actions in $\hat{M}$
        Learn maximum likelihood parameters of new $DBN$ from data $D$
    end for
    return $\hat{M}$
```

The process in Algorithm 6 reduces a given FMDP M, to another possibly *sparser* FMDP $\hat{M}$ by applying the procedure described above. Let π^* and $\hat{\pi}^*$ denote the optimal policies for M and $\hat{M}$, respectively; $\hat{\pi}^*$ will depend on fewer number of variables than π^* because of the absent edges. This means that value trees or policy trees may require less computational resources to store and modify.

4.2.3 Estimating δ

One of the drawbacks of *reduceFMDP* is that it assumes δ as given by an expert. Moreover, it requires that we sample from the model and use it for maximum likelihood learning of model parameters of the reduced model. In this section we propose a method to estimate a reasonable value for δ, denoted δ_{est}, and use δ_{est} to reduce an FMDP without sampling from it.

To estimate δ we assume that the objective for the FMDP is defined as a propositional logic clause in terms of some of the genes in the model. Given the set of genes V and a set Γ of reward genes where $\Gamma \subset V$, let us denote the objective function as the logical formula, $\Phi(\Gamma)$. So, we say that a gene activity profile s satisfies our objective if $\Phi(\Gamma) = 1$ for the values of Γ in s. Usually, there can be more than one ways of achieving an objective. For instance if $\Phi(\{A, B\}) = A \vee B$ for genes A and B then we can try to achieve either $A = 1$ or $B = 1$ or both.

Disjunctive normal form (DNF) is a standardization of logical formulae. It represents the formula as a disjunctions of conjunctions of literals. Every logical formula can be converted into DNF. This means that we can represent any $\Phi(\Gamma)$ as;

$$\Phi(\Gamma) = C_1 \vee C_2 \vee C_3 \vee ...C_k \tag{4.12}$$

where each C_i is a conjunction of literals that we will call the *components* of the objective. Hereafter, we will denote the DNF of $\Phi(\Gamma)$ as $\Phi^N(\Gamma)$, where N stands for normalization. Each of these C_i's actually correspond to different ways of achieving our objective. Finally, let us denote the set of genes that exist in the formula C_i as γ^i.

Assume we are given a control problem composed of the PBN model P of a GRN, action set A, and the objective $\Phi(\Gamma)$. By using the actions in A we will try to achieve the objective, $\Phi^N(\Gamma) = C_1 \vee C_2 \vee ...C_k$. Also remember that each action corresponds to the intervention of one of the genes in the model. We say that C_i is *achievable* if a given action a for gene g *may* satisfy C_i. In other words, if there is a path between g and all genes in γ^i in the unrolled P then we have a *chance* to achieve C_i. Otherwise, if there is at least one gene in γ^i that is not reachable from g in the unrolled P then action a is not useful to satisfy C_i. This defines a mapping that we call the *achievable set of objectives*, C, from A to a subset of C_i's, where $C_i \in C$ iff C_i is *achievable* by some action $a \in A$. We will denote the projection of the objective function to achievable set of objectives as $\Phi^N_C(\Gamma)$, where $\Phi^N_C(\Gamma) = \bigvee_{C_i \in C} C_i$.

Since there can be more than one ways of satisfying an objective, simplifying the problem by decomposing the objective into components may help an FMDP solver to find a reasonable approximate solution easily. For this, we say that two components C_i and C_j are *separated* if $\forall g_i \in \gamma^i$ and $\forall g_j \in \gamma^j$, there is no path between g_i and g_j in the GRN model represented as an *undirected* graph obtained by converting each edge in the original model (P) to an undirected edge. If all $C_i \in C$ are separated from each other we say that C is maximally separated.

Based on the mentioned ideas, we constructed an algorithm to determine a reasonable value for δ. The algorithm CT (for choose threshold) is given in Algorithm 7. CT tries to find the minimum δ that achieves maximum separation of components (decompose the objective) while preserving achievable objectives.

Algorithm 7 $CT(EIS(\Gamma), M, \Phi(\Gamma))$

```
Input: $EIS(\Gamma)$,FMDP $M$, Objective $\Phi(\Gamma)$
Output: $\delta_{est}$
    Compute $\Phi^N(\Gamma) = C_1 \vee C_2 \vee ...C_k$
    $C$ = achievable set of objectives of $M$
    Compute $\delta_{max}$ = maximum $\delta$ that preserves $C$
    Compute $\delta_{sep}$ = minimum $\delta$ that maximally separates $C$
    $\delta_{est} = min(\delta_{sep}, \delta_{max})$
    return $\delta_{est}$
```

Computing δ_{sep} and δ_{max} in CT should be straight-forward; sort $EIS(\Gamma)$ values and eliminate edges in ascending order starting from the minimum, until the constraints are violated. δ_{max} is computed by eliminating edges as long as achievable set of objectives is preserved and δ_{sep} is computed until C is maximally separated. To be able to make as few modifications as possible to the original model (so that the solution is a good approximate policy), in the final step, minimum of δ_{sep} and δ_{max} is chosen as δ_{est}.

Given δ_{est} by CT, the reduction procedure is given in Algorithm 8. Note that Algorithm 8 does not require sampling from the original model and also all edges less than δ_{est} are eliminated; different from *reduceFMDP*[2]. The last step of *reduceFMDP2* updates the reward function R in $\hat{M}$. Defining a reward function for a given objective is a relatively subjective procedure. To the best of our knowledge, there is no well-defined procedure to map a given objective to a reward function and finding such a mapping is out of the scope of this thesis. In this thesis, we assume the objective $\Phi(\Gamma)$ is given, and it is mapped to a reward function R that represents the objective and the cost of actions as "good" as possible. So, if we represent this mapping as F where $F : \Phi(\Gamma) \rightarrow R$, then by using the same procedure F we can also map $\Phi_C^N(\Gamma)$ to a new reward function; this corresponds to the last step of *reduceFMDP2*. For instance, given $\Phi(\{A, B\}) = A \vee B$, a possible reward function can be constructed by assigning a reward of 10 to the states where $\Phi(\{A, B\})$ is satisfied and 0 to all other states. So the reward function returns 0 for $A = 0, B = 0$ and 10 otherwise. Now assume $\Phi_C^N(\{A, B\}) = A$ which

[2]This difference is not important as δ in *reduceFMDP* can be mapped to the usage in *reduceFMDP2* by choosing δ as the maximum EIS of the eliminated edges

means that the achievable set of objectives is $\{A\}$. The reward function for Φ_C^N constructed by the same procedure above assigns 0 to states where $A = 0$ and 10 for $A = 1$.

Algorithm 8 *reduceFMDP2*$(EIS(\Gamma), \delta_{est}, M)$

```
Input: $EIS(\Gamma)$,$\delta_{est}$ by $CT$,FMDP $M$
Output: FMDP $\hat{M}$
  $\hat{M} = M$
  for all $E_{i,j} \in S$ do
    if $EIS_{i,j}(\Gamma) < \delta_{est}$ then
      Remove edge $E_{i,j}$ from $DBN$s for all actions in $\hat{M}$
      Marginalize out $g_i$ from the CPTs of $g_j$ for all actions
    end if
  end for
  Update reward function $R$ of $\hat{M}$ based on $C$
  return $\hat{M}$
```

4.3 Related Work

Generally, control in GRNs is studied on Markovian models [79, 17, 18, 64, 66]. In [79], PBNs are investigated in terms of perturbations and interventions. Random gene perturbations in PBNs are introduced. The transition probabilities in the existence of perturbations are derived. Then intervention that is forcibly changing value of a gene is introduced to PBNs. According to a goal (for example reaching the state 111) they try to select the best gene to intervene in terms of the influence concept that is introduced in [78] and first passage times in Markov Chain theory. Finally they investigate the sensitivity of stationary distributions to gene perturbations.

One of the first studies of formulating the problem of control in GRNs in an MDP framework is by Datta *et al.* [17]. PBNs are used as the model and an MDP is formulated and solved by dynamic programming in a general setting. A real world example is given at the end based on gene expression data that constitute a 10-gene network whose objective is to down-regulate one of the genes. Although the derivations are given for a PBN, the network used in the example is ternary (so is not a PBN) and derived using the methods in [48]. But since transition probabilities are important for dynamic programming, ternary valued variables make no difference other than increasing the search space from 2^n to 3^n.

In an extension of the study described above, the authors give the results of dynamic programming solution of the case in which the state of the PBN is not known, but a “clue” about it can be observed in the form of a number of measurable outputs [18]. Results are given on a

7-gene network in which the control objective is to ensure that a gene (namely WNT5A) is not up-regulated. Directly controlling the gene (by an inhibitory protein) gives better results than controlling through another gene that influences WNT5A. Expected costs are decreased and the probability of being in the desired state is increased with control compared to uncontrolled case.

The work described in [64] concentrates on finding an optimal policy using dynamic programming for a PBN that is constructed using the method of [104].They first derive transition probabilities $T(s, s')$ and $T(s, a, s')$ and use this to solve a finite-horizon control problem which minimizes a cost function. The action is set as toggling a gene's value. Selecting the control gene is performed using the influence metric of [78]. They finally give the results of an application to melanoma data where the objective is again to have WNT5A gene not up-regulated and show that the cost is decreased with control.

The infinite-horizon of the problem whose finite-horizon solution was given in [64], is studied in [66]. The authors use PBNs derived by the method of [67] and transition probabilities for PBNs derived in [64]. They give the solutions for both discounted costs and average cost per stage. Results are given on a melanoma application which includes a 7-gene network that has 128 states. Value iteration and policy iteration results are given according to total cost with control and without control, and according to time spent in desirable and undesirable states during the application of the current policy.

The problem is also investigated by dividing the finite-horizon into episodes of control and monitoring that is generally done in treatment of diseases [1]. Again the model is assumed to be given, and dynamic programming solutions to four different types of problems are studied, which are finite-control, finite-control finite-monitoring, finite control infinite-monitoring and infinite control. For a GRN, three kinds of models can be available; M as GRN model, L as state cost model, and K as state-action cost model. All or some of these models can be available to us in solving problems of optimal action sequences based on our biological knowledge of the domain. In this study, solutions depending on the availability of these models and ways of combination of these models are also discussed including a multi-objective solution.

As opposed to what has been suggested in [17], state costs and state-action costs are in fact non-additive because they denote different kinds of values. Based on this fact, a multi-objective solution is suggested in [2]. The solution is general for any number of objectives, but specifically the solution for state and state-action costs is given.

An approximate solution by reinforcement learning (Q-learning) based on the assumption of a model simulator is given in [25]. The results are promising for scalability but the authors report the results only for a 10-gene network to be able to compare to the optimal solutions. As all biological data incurs some type of noise, the models derived from such data may be erroneous. Pal *et al.* [65] investigate the effect of the application of a control policy on a gene network whose transition probabilities are different from the one for which the policy is found. Finally, in real life some constraints may exist in the application for a treatment to a patient. For instance, a given action may be applied only up to a certain number of times. Deriving a control policy in the existence of such constraints has been investigated by Faryabi *et. al* [26].

4.4 The Gap Covered by the Proposed Methods

As already mentioned earlier in this chapter, the main issue in control problems is the curse of dimensionality. In real problems, such as the GRN control domain, it is hard to reach a solution by applying the available techniques, especially with MDP representations. All the proposed methods for finding an intervention strategy in GRN control is based on MDP representations that are hardly scalable. Because of this, all the applications in the field are restricted to small networks that have no more than 7 or 8 genes at best. Although these methods are promising and important for handling the control problem in general, algorithms that can work for larger networks are needed as a GRN can include genes in the order of thousands.

My expectation is that the reader has absorbed the different novel aspects of the two types of algorithms proposed in this chapter to fill this gap. The first algorithm concentrates on MDP representations, in which a feature (or gene) reduction method is devised for reducing the given model to another, where the latter have a significantly smaller state space than the former. This type of reduction provides near-optimal solutions to the otherwise unsolvable GRN control problems. This reduction is also different from the existing MDP reduction (or minimization) techniques already described in the literature (see [30] for instance) in the sense that it is applied before modeling to focus on the components of the network that is essential for control.

The second proposed method works well for FMDP representations. There are two contributions here; first, it is the first application of FMDP representations to GRN control domain, second, a new reduction algorithm for a given FMDP representation is proposed for near-optimal solutions to control problems. This algorithm is very promising to reach the objective of solving genome-wide control problems.

Although FMDP formalism is more appropriate for domains that are easily factorized like GRN control, MDP representations are still investigated due to their ease of implementation and interpretation. This makes both of the methods proposed in this chapter applicable for the studies in this field. This becomes more convincing by the experimental results reported in the next chapter.

CHAPTER 5

EXPERIMENTAL RESULTS

This chapter includes the results of the experiments performed to demonstrate the applicability of the proposed algorithms. Both synthetic and real data sets are used in the experiments. Naturally following from the presentation of the algorithms, the results are also given in two parts; the first part of the results covers the modeling and the second part of the results is dedicated to the control of GRNs. Each section includes the description of the data sets, results and discussion.

5.1 Implementation and Execution Environments

The algorithms are mostly implemented in Matlab[1] (unless otherwise stated). No specific toolbox dominates the implementation but some functions from the statistics toolbox are used. Other than that, for tasks like graph construction/drawing and data pre-processing, some features of R[2] and Python[3] are used.

Most of the experiments are performed on a computer with Intel Core2 2.4 GHz CPU and 3GB of RAM running Linux. For some of the time consuming empirical experiments, an HPC cluster[4] is also used. But none of the algorithms here require more than an ordinary desktop computer.

5.2 Constraint-based Modeling of Gene Regulatory Networks

5.2.1 Data sets

In addition to the data constructed from synthetic networks, there are four real data sets used for the experiments in this section. These are widely investigated data sets in the litetature.

[1]The Mathworks - MATLAB and Simulink for Technical Computing, `http://www.mathworks.com`, accessed 1-June-2009

[2]The R Project for Statistical Computing, `http://www.r-project.org`, accessed 1-June-2009

[3]Python Programming Language, `http://www.python.org`, accessed 1-June-2009

[4]High Performance Computing, `http://hpc.ceng.metu.edu.tr`, accessed 1-June-2009

The list of these data sets are as follows:

1. The expression data of Spellman *et. al.* [81]: It is time-series gene expression data composed of 77 time steps for different phases of the cell cycle and includes 6178 genes.

2. The TF binding location data of Lee *et al.* [53]: It has binding data for 106 TFs and 6270 genes.

3. The TF binding location data produced by Harbison *et al.* [34]: It is composed of binding data for 203 TFs and 6229 genes.

4. Protein concentration data from cytometry experiments by Sachs *et al.* [73] for Raf signalling pathway. It is static data composed of 500 samples for each of the 11 genes.

5.2.2 The PCPr algorithm

To evaluate the PCPr algorithm as described in Chapter 3, we performed two different types of experiments. The experiments differ in the source of data used; synthetic networks or real biological data. In the experiments with synthetic networks, a sparse synthetic network which resembles biological networks is constructed. Then we sample from this network and check how well we build the network from the data. We repeated the experiment for synthetic networks with different number of nodes and different sample sizes. Prior information matrix is built from given network by adding some amount of error (noise).

The second type of experiment is the one that involves real biological data. This time, prior knowledge is constructed from one type of data (TF binding data) and the other type of data (microarray data) is used in statistical tests by adapting the significance level according to prior knowledge. Results are verified by constructing a gold standard network from the literature whenever possible. Methods based on Gene Ontology annotations are used when a gold standard network is not available or is hard to construct.

There are three evaluation measures that we use in the experiments; precision, recall and structural hamming distance (SHD). Precision and recall are defined in terms of the number of true positive (TP), false positive (FP) and false negative (FN) edges. Precision is given as; $\frac{TP}{(TP+FP)}$, and recall is $\frac{TP}{(TP+FN)}$. High precision along with high recall should be the objective for an algorithm. SHD is a measure to find a distance between two directed graphs where each operation of edge removal, edge orientation and edge addition is defined to be of distance 1. SHD is the total number of operations applied on one of the graphs to obtain the second graph. We will use SHD to evaluate directed graphs, precision and recall for undirected graphs. Since the skeletons of the graphs (sometimes referred as undirected dependency graphs [20]) are considered important in the bioinformatics community, we report both undirected and directed graph evaluations.

Synthetic networks:

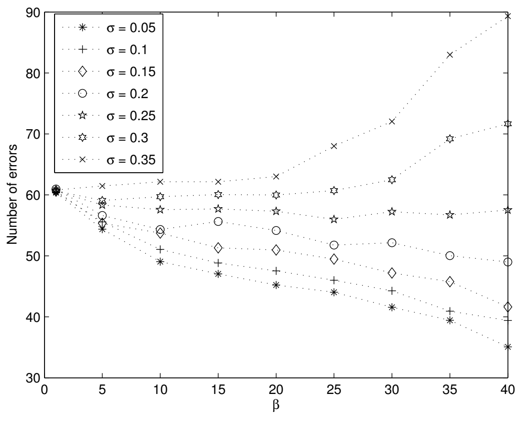
To construct synthetic networks, we used the method described in [43], which is publicly available in the R-package *pcalg*. The algorithm in *pcalg* constructs a DAG whose sparseness can be controlled by a parameter. The synthetic graphs used in these experiments are constructed based on the following parameters:

- Number of nodes(p): $p \in \{20, 40, 60, 80, 100\}$.
- Expected number of connections for each gene($E[N]$): 3

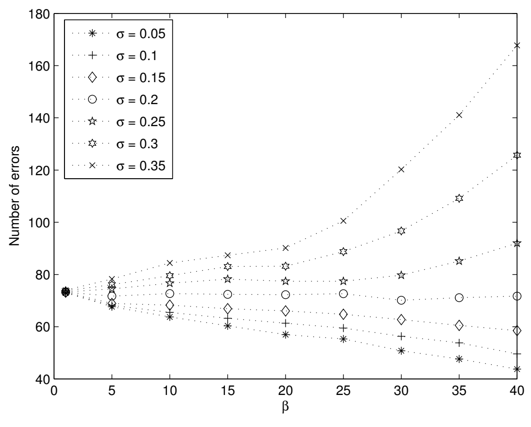
A total of 10 graphs are constructed for each p with $E[N]$ set as specified above. As GRNs are thought to be sparse, we chose $E(N)$ as 3. From each of these 10 graphs, 5 datasets with size n are generated, where $n \in \{100, 1000, 5000\}$. The reported results are the averages of these.

The prior information matrix B for the synthetic networks is constructed from the DAG G constructed by the *pcalg*. If $G_{ij} = 1$ (the edge exists in G) then B_{ij} is set to a random real number in the range $[0.5, 1]$. When $G_{ij} = 0$, we have $B_{ij} \in [0, 0.5]$. After that, a noise term ε_{ij} is added, where ε_{ij} is a random variable distributed as $N(0, \sigma)$. In the given results, σ is set to 0.1, which means that the error is approximately in the range $[-0.25, 0.25]$ with 0.99 probability.

Choosing the value of β depends on the noise level ε_{ij} of prior knowledge. Given a low noise level (a low standard deviation σ of ε_{ij} with mean 0), a high value of β can increase the quality of the output, but a very high value can also bias the output in such a way that it only represents the prior knowledge. So, the value of β must be chosen carefully.



(a)



(b)

Figure 5.1: Change in number of errors with β for a) $p = 40$ b) $p = 60$

Figure 5.1 shows the change in the number of errors with the value of β for different values of σ. The curves plotted in Figure 5.1 are constructed as follows; given a DAG G with p nodes and $E[N] = 3$, we constructed matrix B for a given value of σ and applied the PCPr algorithm to a dataset of size 500. We performed 50 such runs, producing a new B each time; the number of errors are averaged over these 50 runs. This procedure is repeated for a set of σ values. Figure 5.1 shows the results of this procedure for $p = 40$ and $p = 60$. If we don't know the value of σ (which is generally the case), choosing a high value of β is risky. For instance, if β is chosen as 35 for $p = 60$, the number of errors approximately doubles for $\sigma = 0.35$ compared to the case when β is 0 (see Figure 5.1). So, choosing a value between 15 and 25 decreases the number of errors in case of small σ (low error) and does not increase the number of errors too much if σ is large. In the experiments conducted using the synthetic datasets, since we don't assume any prior knowledge of σ, we experimentally chose the value of β as 20.

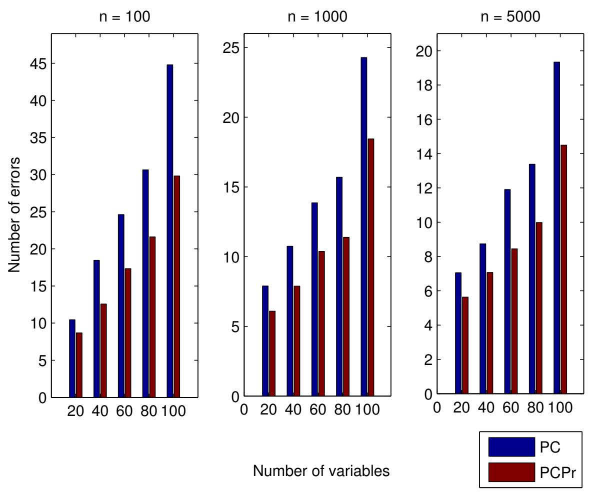


Figure 5.2: Number of errors in finding the skeleton

Figure 5.2 depicts the total number of errors in the experiments with the synthetic data. The error is measured as the sum of false positive and false negative edges, the x-axis shows the number of variables and α_0 is set to 0.05. It is obvious from Figure 5.2 that prior knowledge always changes the result in a positive way, though effects differ. Here, it is important to emphasize one more time that we could do better by choosing a larger value for β if we assumed a knowledge of σ as 0.1.

Gene Regulatory Networks:

In this experiment, we used TF binding data in addition to the microarray data in order to be able to cope with the problem of small samples in microarray data. Each TF binding data cell has a p-value which indicates the confidence of the binding of a certain TF to a certain gene. The smaller the p-value, the more confident we are about that binding. In this study, microarray data is the data we used for conditional independence tests; from the TF binding data we constructed the prior information matrix B to set the significance level. In other words, binding data will constitute our prior information on the existence of edges between genes and TFs.

In the experiments, we used the gene expression data of Spellman *et al.* [81] and the TF binding data of Lee *et al.* [53]. Each time sample in the expression data of Spellman *et al.* is used as a feature for a gene; it is used in deriving the conditional independence relationships along with binding data as the prior knowledge in the PC algorithm. Evaluating the results is performed as searching the literature for evidence of the derived interactions, so we had to choose a subset of genes to demonstrate the performance of the method. The gene set chosen to model is the 25-gene set that was previously used in [6]; this choice will also allow for comparison of the results. The expression data corresponding to this set is extracted from Spellman *et al.* and missing values are imputed in microarray data using the k-nearest neighbors algorithm (KNNImpute [88]) with $k = 10$. No further processing is performed for expression data. But, the binding data has to be processed to derive probability values corresponding to the entries in B. Next, we summarize this process by using the notation described in [6].

As mentioned before, the TF binding location data is in the form of p-values, which can be interpreted as indicators of edges being present in the graph. To convert a p-value to a probability of an edge being present, we follow the method described in [6]. The p-value P_{ij} of edge E_{ij} is assumed previously to be exponentially distributed given that E_{ij} exists in the model structure G [76], and uniformly distributed given that E_{ij} does not exist (follows from the definition of p-value). This means that $P(P_i = p|E_i \in G) = \lambda e^{-\lambda p}/(1 - e^{-\lambda})$, where λ is a parameter of exponential distribution, and $P(P_i = p|E_i \notin G) = 1$. After this step, applying Bayes rule and integrating over predefined minimum and maximum values of λ lead to the following:

$$B_{ij} = P(E_{ij} \in G|P_{ij} = p) = \frac{1}{\lambda_H - \lambda_L} \int_{\lambda_L}^{\lambda_H} \frac{\lambda e^{-\lambda p} \vartheta_{ij}}{\lambda e^{-\lambda p} \vartheta_{ij} + (1 - e^{-\lambda})(1 - \vartheta_{ij})} d\lambda \tag{5.1}$$

In Eq (5.1), G is the structure of the model, P_{ij} is the p-value of E_{ij}, which indicates the confidence of the binding of TF i to gene j, λ_L and λ_H are the chosen lowest and highest values of λ, respectively, and $\vartheta_{ij} = P(E_{ij} \in G)$. In the computations, $\vartheta_{ij} = 0.5, \lambda_L = 0.1$ and $\lambda_H = 10,000$ were used as suggested in [6]. The integral is then solved numerically for each fixed value of p_{ij} [6].

In the chosen set of 25 genes, 10 exist as TFs in the binding data of Lee *et al.* So, the p-values are available only for edges connecting these 10 genes to the other genes in the set. Thus, only for these edges B_{ij} values can be computed from the binding data. After we compute each

B_{ij} for these edges, the other entries in B are filled with 0.5, indicating no prior knowledge for those edges. Each entry in B then, indicates the probability of a TF binding to a certain gene.

The connections of genes in a GRN may be affected by the current phase of the cell cycle. To be able to explain these dependencies, we added a phase variable in the same way as described in [6]. The phase variable is assumed to be connected to all the other genes and these connections are assumed to be permanent, *i.e.*, they are not tested like the other edges in conditional independence tests in the algorithm.

To be able to evaluate the results, we built what is called a "gold standard" network, which includes the edges that are experimentally verified in the literature. In order to construct the gold standard network, we used the pathwaystudio tool (available at: "Ariadne Genomics: Pathway Studio, `http://www.ariadnegenomics.com/products/pathway-studio`, accessed 1-June-2009"). This tool takes a set of genes and builds all the direct interactions among the given genes based on the ResNet database. To the network derived in this way, we also added the interactions extracted from the BioGrid database [84]. We used this network as our "gold standard" network. The results are given in Table 5.1, where TP, FP and FN, respectively, stand for true positives, false positives and false negatives.

Table 5.1: Quality of the derived networks for 25-gene experiment

Algorithm	**TP**	**FP+FN**
PC	5	$19 + 79 = 98$
PCPr ($\beta = 20$)	16	$11 + 68 = 79$
PCPr ($\beta = 30$)	20	$11 + 64 = 75$
PCPr ($\beta = 40$)	37	$14 + 47 = 61$
DBN_{BA}	44	$34 + 40 = 74$

Instead of choosing a fixed β, this time we give the results corresponding to 3 different values of β, namely 20, 30 and 40; these values have been selected based on some initial tests where we realized that the number of false positives increases after 40. The network derived by PCPr with $\beta = 40$ is also given in Figure 5.3. It is obvious from the results reported in Table 5.1 that PCPr always outputs a better network than PC. This shows the effectiveness of our procedure in combining multiple types of biological data in this study, *i.e.*, microarray data and TF binding data. Table 5.1 also includes the results of Bernard *et al.* [6] for comparison; We name their algorithm as DBN_{BA}. Although they derive a different type of network, namely a dynamic bayesian network, we can compare the results by converting the graph in their results to an undirected graph. Using the latter undirected graph we compared their output to our "gold standard" network[5]. Precision value of PCPr for $\beta = 40$ is 0.72 and the precision for the output of DBN_{BA} is 0.56. This shows that the existence of each edge in the output of PCPr is more "reliable". Recall values for the outputs are very close, 0.44 and 0.52 for PCPr and DBN_{BA}, respectively. As a result, PCPr outputs a network with a much better precision and comparable recall values compared to DBN_{BA}.

Our results also verify the results of Bernard *et al.* in the sense that the binding data of Lee *et al.* is more informative in deriving the GRN than the microarray data of Spellman *et al.*, at least for the chosen set of genes.

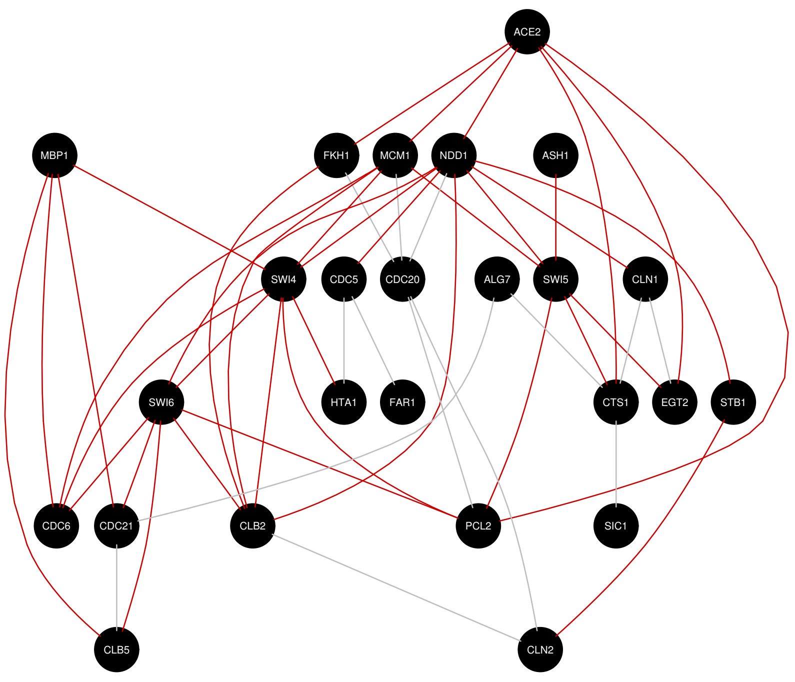


Figure 5.3: Output of PCPr with $\beta = 40$ and $\alpha_0 = 0.05$. Red edges are the ones that have been verified in the literature, green ones are the novel relationships proposed by the algorithm.

5.2.3 PCPDPr algorithm

The experimental results for the PCPDPr algorithm as described in Section 3.1 are reported and discussed in this section. We follow the same evaluation strategy applied to the PCPr

[5]Note that the numbers reported in Table 5.1 are different from the ones reported in [6]; we mapped them into the scale used by our model because of the different evaluation criteria and different "gold standard" networks.

algorithm; we also compare PCPr to PCPDPr.

Synthetic Networks:

Note that the PCPDPr algorithm is proposed for graphs that have some densely connected nodes; we call such graphs *partially-dense* (PD) graphs. So synthetic PD graphs have to be constructed for testing PCPDPr.

To construct synthetic networks that are PD, we used a modified version of the algorithm in *pcalg*. We modified the algorithm to be able to produce networks that have dense nodes, i.e., whose expected number of neighbors is larger from the other nodes. We constructed PD graphs this way and used them for testing the proposed algorithms.

The synthetic graphs used in the experiments are constructed based on the following parameters:

- Number of nodes (p): $p \in \{100, 300, 500\}$.
- Number of dense nodes (dn) : To the best of our knowledge, determination of the number of dense nodes for a given genome has not been studied before. Number of genes in a functional category in a genome scales by following a power-law [90]. This power-law relationship is given as $n_c = \kappa * g^\gamma$, where n_c is the number of genes in category c, κ and γ are the parameters of the relationship. The key components in the GRN are usually the transcription factors, each of which regulates several genes and other TFs. As the TFs can regulate other components (genes), we set the TFs to be the dense nodes in the GRN. In order to find the number of TFs in the GRN, we found the number of genes that are related to "Transcriptional regulation" category in both eukaryotes and bacteria [90]. The average is taken because not all the TFs have the same degree of density. Also, since there are common TFs between eukaryotes and bacteria which are essential for the basic cellular processes like DNA synthesis and signal transduction, it is more reasonable to include both eukaryotes and bacteria. So, we set the number of dense nodes by using the above equation where $\kappa = 0.002$ and $\gamma = 1.5$. γ is chosen as close to the mean parameter value of "Transcriptional regulation" category for eukaryotes and bacteria. $\kappa = 0.002$ is derived again by using the values in [90]. So for instance the number of dense nodes is 22 for a network with 500 nodes.
- Expected number of edges for each node ($E[N]$): 3 for sparse nodes and 30 for dense nodes. 30 is chosen by calculating the average number of connections for some of the dense genes in different organisms (see Table 5.2).

A total of 10 graphs are constructed for each p with $E[N]$ and dn set as specified above. From each of these 10 graphs, 5 datasets with sample size n are generated, where $n \in \{100, 250, 500\}$. The reported results are the average of these. Notice that the chosen values of n are typical sample sizes for a microarray experiment.

The prior information matrix is constructed in the same way as in the experiments for PCPr described in the previous section. β is set as 20 and α_0 is again set to 0.05.

One important difference from the previous experiments is that this time, not only the skeleton but the PDAG derived from the second part of the PC algorithm is evaluated as well.

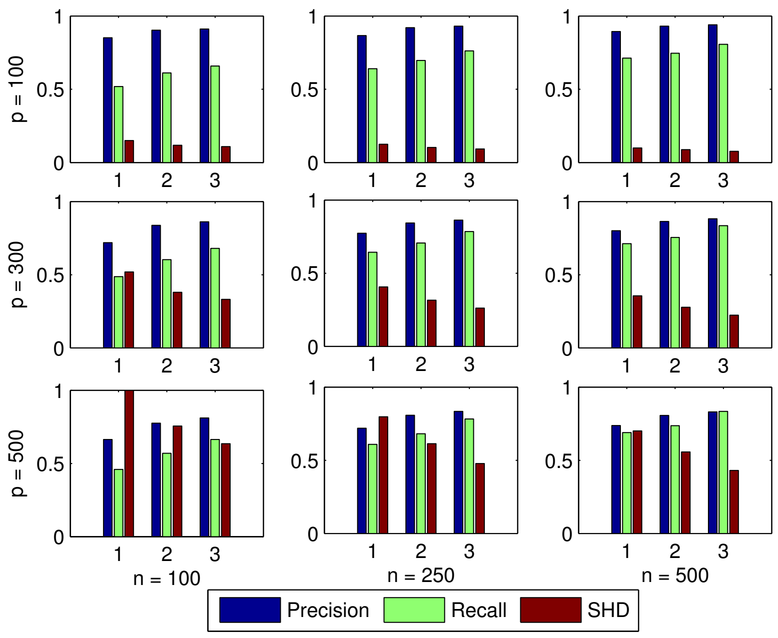


Figure 5.4: Evaluation results for synthetic experiments. 1: PC, 2: PCPr, 3: PCPDPr. Rows correspond to a fixed number of nodes (p) and columns correspond to a fixed sample size (n).

Figure 5.4 gives the evaluation results (note that SHD values are normalized to $[0, 1]$). It can be easily seen in Figure 5.4 that prior knowledge always improves the result, though effects differ. Also, the PCPDPr algorithm is always better than PCPr. Notice that as the number of dense nodes follows a power-law, effect is more apparent for larger sized networks. Rate of improvement in precision and recall for $p = 100$ is larger than rate of improvement in SHD. But for $p = 500$ for instance, SHD also improves more rapidly. This shows that the improvement in skeleton is reflected to the edge orientation part as well. So PCPDPr can discover causal interactions much better than PC and PCPr. Also another important aspect is the consistency of PCPDPr with increasing sample size. It outputs a better network consistently when the sample size increases (see the columns in Figure 5.4).

Figure 5.5 gives the execution time in seconds for the algorithms, where it is obvious that PCPDPr outperforms others for all settings, though it is difficult to reach a conclusion for

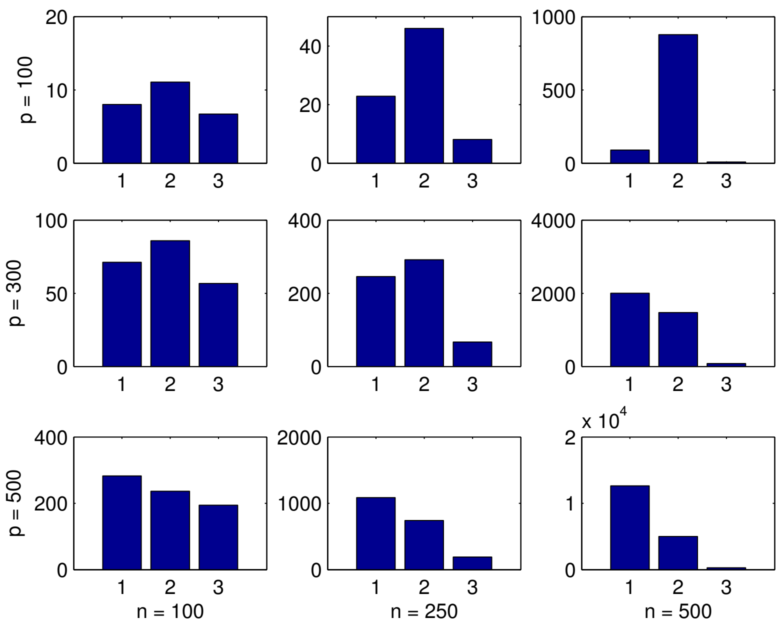


Figure 5.5: Execution time (seconds) for the algorithms. 1: PC, 2: PCPr, 3: PCPDPr.

PCPr and PC. Since the mistakes done by conditional independence tests can both increase or decrease the computational time required by the algorithms, the execution time depends greatly on the used network and data. But the gain by PCPDPr is clear; it always requires less time to complete and sometimes the difference is huge. For instance, for $p = 500$ and $n = 500$, PCPDPr works 41 times faster than PC.

Table 5.2: Some TFs and number of bindings for a) E. Coli, b) B. Subtilis, c) S. Cerevisiae

Gene	Bindings
arcA	20
crp	72
purR	16
fnr	22
rpoE_rseABC	24
ycfC_purB	26
himA	21

(a)

Gene	Bindings
AbrB	33
GerE	20
codY	15
CcpA	44
TnrA	26
Fur	23
PhoP	19

(b)

Gene	Bindings
Mig1	27
Msn4	32
Skn7	21
Ste12	72
Tec1	44
Ume6	38
Gln3	29

(c)

Gene Regulatory Networks:

GRNs are generally thought to be sparse. For example, the average number of connections in *Escherichia Coli* transcriptional network is given as 5 in [77]. But, there are also some TFs that are found to bind on a large number of genes (either as an activator or a repressor). Some of the TFs that are known to bind to more than 15 genes (by using TF binding data) in *Escherichia coli*, *Bacillus subtilis* and *Saccharomyces cerevisiae* are given in Table 5.2 [59, 60, 77]. (The data used for Table 5.3(a) and Table 5.3(c) are available at "Uri Alon's Molecular Cell Biology Lab, `http://www.weizmann.ac.il/mcb/UriAlon` (accessed 1-June-2009)", and the data used for Table 5.3(b) is available at "DBTBS, `http://dbtbs.hgc.jp` (accessed 1-June-2009)".

Table 5.2 demonstrates and emphasizes that our proposed approach can be applied to GRNs as well. In this section, the results of applying the proposed algorithms to real gene expression data are presented. We used the gene expression data of Spellman *et al.* [81] and the TF binding data of Lee *et al.* [53].

The same data set used to test PCPr in 5.2.2 is also used to test PCPDPr. The prior information matrix, B, is again derived from binding data by the method of Bernard *et al.* [6].

Dense nodes are found in the same way described in Section 3.1.2. If a gene is estimated to have more than 5 connections with probability greater than 0.8 in B, then that node is considered as dense. Based on the data we used, genes that have been found as dense from B are ACE2, MCM1, NDD1, SWI4, SWI5, SWI6 and CLB2. These dense genes are important for transition between phases of the cell cycle. Both SWI5 and ACE2 are TFs that activate the transcription of genes expressed early in G1 phase in order to promote the transition from M to G1. The three genes SWI4, SWI6, and MBP1 are DNA binding components of MBF and SBF, which regulate the late G1 specific transcription, including cyclins and DNA synthesis genes. The two genes NDD1 and CLB2 play a role in G2/M transition. At this stage of the cell division, the cell undergoes a huge change in the transcription of genes in order to proceed to the following phase.

The same "gold standard" network mentioned in Section 5.2.2 is used to evaluate the algorithms. The results are given in Table 5.3. The network derived by PCPDPr with $\beta = 40$ is also given in Figure 5.6. It is clear from the results reported in Table 5.3 that PCPDPr outputs a better network in a shorter time. The results for DBN_{BA} and PCPr are included in this table again for the ease of comparison.

As reported in Table 5.3, PCPDPr and PCPr always have higher precisions than both PC and DBN_{BA}. And PCPDPr with $\beta = 40$ has a much better precision than DBN_{BA} with a comparable recall. Also, we should emphasize here that since we could not obtain the algorithm but only the results for DBN_{BA}, we can not report the time for that algorithm. But, the time elapsed for PCPr and PCPDPr for $\beta = 40$ shows that PCPDPr outputs a better network in much shorter time than PCPr.

Table 5.3: Quality of the derived networks for 25-gene experiment

Algorithm	**TP**	**FP+FN**	**Time**	**Precision**	**Recall**
PC	5	$19 + 79 = 98$	0.37	0.20	0.05
PCPr ($\beta = 20$)	16	$11 + 68 = 79$	0.48	0.59	0.19
PCPr ($\beta = 30$)	20	$11 + 64 = 75$	0.67	0.64	0.23
PCPr ($\beta = 40$)	37	$14 + 47 = 61$	4.17	0.72	0.44
PCPDPr ($\beta = 20$)	18	$11 + 65 = 76$	0.49	0.62	0.21
PCPDPr ($\beta = 30$)	25	$11 + 59 = 70$	0.46	0.69	0.29
PCPDPr ($\beta = 40$)	38	$14 + 46 = 60$	0.69	0.73	0.45
DBN_{BA}	44	$34 + 40 = 74$	-	0.56	0.52

Again, by considering the result when $\beta = 40$, we can see that PCPDPr is missing important interactions like: SWI6-STB1, SWI6-CLB2, SWI6-MBP1, SWI5-CLB2, ACE2-CLN2, and ACE2-CDC28. Most of these interactions are also missing in the study of Bernard *et al.* [6]. It is also worth mentioning here that all these are not actually protein-protein interactions [84], where protein-protein interactions should not be expected to be extracted by using the types of data used here.

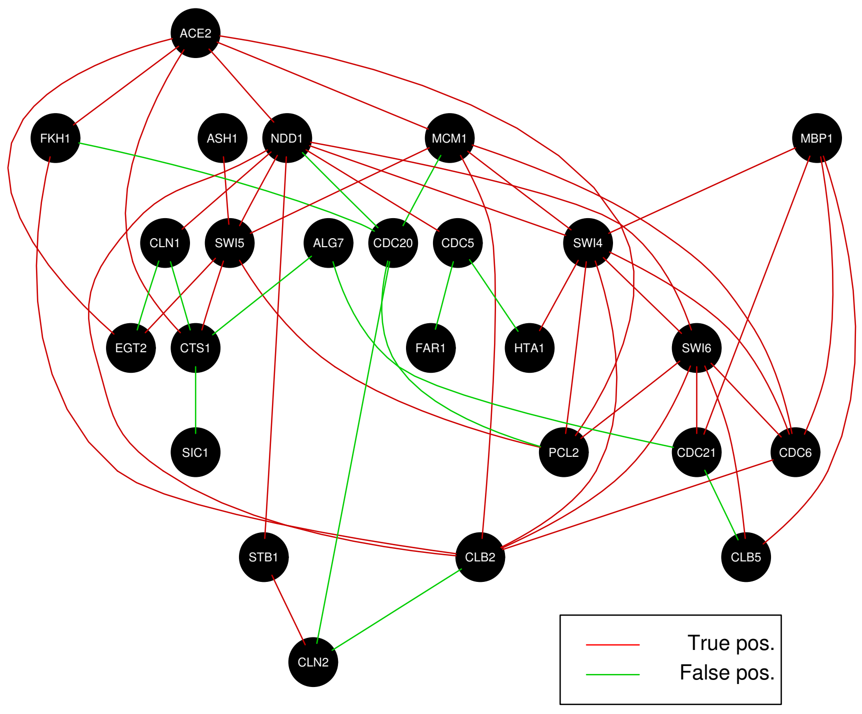


Figure 5.6: Output of PCPDPr with $\beta = 40$ and $\alpha_0 = 0.05$

In the resulted network shown in Figure 5.6, we observe that we have 14 FP interactions; in other words, our algorithm predicted 14 interactions which are not discovered yet; 11 of those interactions are gene-gene interactions and only 3 are between TFs and genes; this indicates that using TF binding data is useful to decrease FPs. Our algorithm predicted the false positive relationship between cdc21 and CLB5. The reason for this FP type of relationship is that both cdc21 and CLB5 are regulated by the same TF, namely MBP1; this indicates that both cdc21 and CLB5 are correlated. Similarly, the edge between CLB2 and CLN2 resulted because both are regulated by two different TFs which are regulated by NDD1. We also predicted a false positive relationship between CLN1-CTS1 and CLN1-EGT2. A possible interpretation for this could be that both CTS1 and EGT2 are regulated indirectly by NDD1 through SWI5; CLN1 is also regulated by NDD1. Therefore, our algorithm predicted interactions between CLN1 and both CTS1 and EGT2. These two interactions were also predicted by Bernard *et al.*. Two more interactions, namely FKH1-cdc20 and NDD1-cdc20, are worth considering for further investigation. Both FKH1 and NDD1 are TFs required for G2/M specific transcription; also cdc20 is important in metaphase/anaphase transition in the M-phase of the cell cycle. This result should be interesting because FKH1 and NDD1 may control the expression of cdc20 through other genes, which we did not consider in this experiment. PCPDPr has an additional advantage that supports the need for its development; PCPDPr discovered all the edges discovered by PCPr, and in addition it discovered a new TP edge, namely CDC6-CLB2, which is the only difference between the outputs of the two algorithms.

Comparing our TP interactions with those reported by Bernard *et al.*, it can be seen that there are 30 common interactions. In addition to these, PCPDPr did discover 7 novel interactions, with respect to Bernard *et al.*, like: ACE2-PCL2, MBP1-CLB5, FKH1-CLB2, SWI5-PCL2, CDC6-CLB2, SWI6-CDC6 and SWI6-CLB5.

Key TFs based validation

Gene Ontology (GO) is one of the most important ontologies built within the functional bioinformatics field [16]. The goal of GO is to provide a structured and controlled vocabulary to describe gene functions and the process in which the genes are involved.

We validated our results using GO annotations based on the sub-networks derived from the resulting network shown in Figure 5.6. We grouped the 10 TFs into four groups: G1/S transition of mitotic cell cycle (SWI4, SWI6 and STB1), G2/M specific transcription in mitotic cell cycle (FKH1 and NDD1), Interphase of mitotic cell cycle (SWI4, SWI5, SWI6, FKH1, ACE2, STB1 and NDD1) and DNA replication (MBP1 and MCM1); these TFs will be called key TFs. We considered all the genes and (non-key) TFs which interact with each group of key TFs as a sub-network, and we validated the GO annotations for each subnetwork using the GO Term Finder available at "Gene Ontology Term Finder, `http://db.yeastgenome.org/`

`cgi-bin/GO/goTermFinder` (accessed 1-June-2009)". This system takes a set of genes and returns p-values corresponding to GO terms. Each p-value indicates the confidence that the set of genes share the corresponding GO term. The smaller the p-value is, the more specific is the GO term shared by the genes.

Here, we propose a new technique which utilizes key TFs to measure and validate the significance of the interactions between the genes and the TFs of each group. The proposed approach works as follows. We first find the p-value of the TFs within each group of genes and the genes/TFs they interact with; we denote this set of genes by S. Then, we find the p-value of the genes/TFs with whom the key TFs interact, after excluding (from the former TFs) key TFs that do not interact with each other, *i.e.*, we leave in the former set of TFs all key TFs that have internal interactions among each other; we denote this set of genes by S'.

To illustrate the proposed validation process, consider the following example set S of G_1/S group which contains the following TFs/genes (SWI4, SWI6, STB1, CLB2, PCL2, HTA1, cdc6, MBP1, MCM1, NDD1, CLN2, cdc21, CLB5). The genes/TFs other than the key TFs are the genes that the key TFs interact with. Set S' has the following TFs/genes (SWI4, SWI6, CLB2, PCL2, HTA1, cdc6, MBP1, MCM1, NDD1, CLN2, cdc21, CLB5). We see that SWI4 and SWI6 are included in the set S' because SWI4 interacts with SWI6, and SWI6 interacts with SWI4. We found the p-value for each set of genes S and S' for the four groups enumerated above. The p-value in S' indicates how significant are the interactions among TFs in S. If the p-value of S' is very close to the p-value of the corresponding S, then we say that we have gained most of the information that was in S, and this infers that the interactions within S are significant. We applied this method to the four groups enumerated above and the results are summarized in Table 5.4.

Table 5.4: P-values of S and S' sets for both PCPr and PCPDPr when $\beta = 40$

Group	**Set S**	**Set S'**
G_1/S	$9.39 \times e^{-7}$	$5.6 \times e^{-5}$
G_2/M	0.00025	>0.01
Interphase	$4.23 \times e^{-14}$	$4.23 \times e^{-14}$
DNA Replication	$1.34 \times e^{-5}$	0.00996

From the results reported in Table 5.4, it can be easily seen that our sub-networks for the first and fourth groups are strong as they gained all of the interactions in the S set. The other two groups did not gain enough information from set S as they contain small number of TFs. Since the sets S and S' are the same for both PCPr and PCPDPr (because the interactions are almost the same), we had the same p-values for both algorithms.

Finally, we want to elaborate more on the applicability of the proposed validation approach. There are three cases to be considered as the key TFs are concerned. In the first case, there does not exist any interaction between the key TFs. Consequently, no key TF is expected to be in set S'. As a result, the p-value of set S' will be much larger than the p-value of set S; this indicates that the information in S was not completely gained. This case is what we see in the DNA Replication group. The second case covers the situation where each key TF is connected to at least one other key TF. In this case, the p-values of the two sets S and S' are the same because S and S' contain the same set of genes. This case is prominent in group Interphase. The last case is somehow in between the other two cases, *i.e.*, only some of the key TFs are connected to each other while each of the remaining key TFs are not connected to any key TFs. In this case, not all the key TFs in S will be present in S', and that will cause the p-value to drop down. Depending on the significance of the eliminated key TFs, the p-value of S' will increase. This case is shown in group G_1/S in Table 5.4. To sum up, the proposed approach considers three cases of key TFs connectivity and the information gain depends on the degree of connectivity.

Empirical analysis of gss

As *gss* is a greedy procedure, it may not always find the best graph. To be able to see how well the *gss* procedure effects the output, we defined two algorithms called gPC and gPCPr, where gPC (greedy PC) is the same as the PC algorithm except that the separators are searched only by applying the *gss* procedure. Like the PC algorithm, no prior knowledge is used. On the other hand, gPCPr, is the same as gPC except that it uses prior knowledge to update the value of α_0 (see Section 3.2). We performed some experiments with the same parameters given in Section 5.2.3. The results are shown in Figure 5.7. gPC and gPCPr demonstrate an acceptable performance, but never performs better than PCPDPr, as a structure learning algorithm is said to perform better than another, if it increases both precision and recall. But neither gPC nor gPCPr shows such a performance, though they are sometimes better than PCPDPr in only one aspect. Similar results for (normalized) SHD are also available in Figure 5.7.

One important point in these results is the high recall values in gPC and gPCPr. This shows that as the algorithms do not search for all candidate separators, but only the ones chosen greedily, sometimes they can not find the separators despite their existence, therefore the algorithms keep such edges in the graph. This increases TPs and decreases FNs, but also increases FPs, therefore a larger recall and smaller precision is obtained. There are two types of errors in gPC and gPCPr. The first type of error is the error resulting from small sample size and the other type is the error due to greedy steps. As the sample size increases, greedy step errors become more apparent; precision decreases and recall increases.

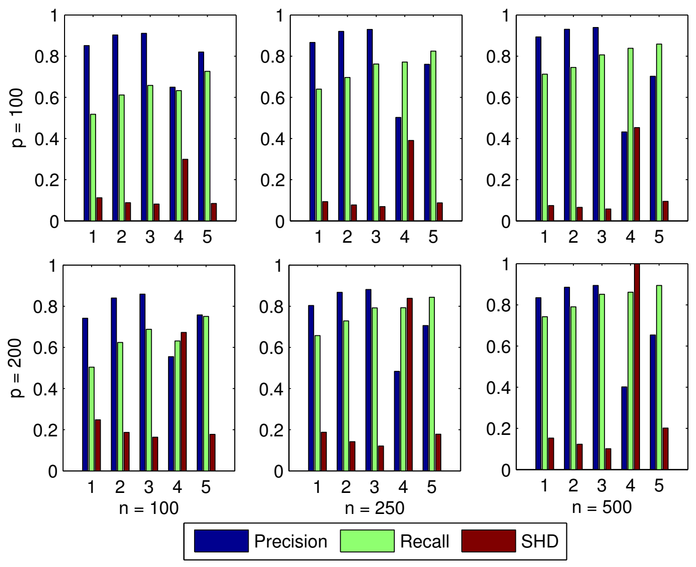


Figure 5.7: *gss* emprical analysis. 1 : PC, 2 : PCPr, 3 : PCPDPr, 4 : gPC, 5 : gPCPr

5.2.4 Yeast cell-cycle model

To be able to demonstrate the scalability of PCPDPr, an experiment is devised to derive a large network by using the PCPDPr algorithm. For this experiment, we chose all genes that have been previously identified as related to the cell cycle [81]. We used the same microarray data as in the previous experiment and used the location data of Harbison *et al.* [34]. There are almost 800 genes that have been previously determined to be cell cycle related [81]. From these genes, we extracted a set of 763 genes for which both microarray data and location data are available. Among these 763 genes, only 27 of them are available as TFs in location data, so only these are used to compute prior information matrix B. A phase variable is then added as in the previous experiment. It took approximately 84.5 minutes for PCPDPr to output a network with 1830 edges for $\beta = 30$ and 87.9 minutes to output a network with 2112 edges for $\beta = 40$; we used the same other parameters as in the previous experiment.

As it is hard to construct a "gold standard" network for such a large gene set, we use only the key TFs based validation for this experiment. This is another evidence in support of the importance of the developed validation approach.

For the analysis, first we have classified the 27 TFs based on their GO annotation. Seven genes are related to Interphase of mitotic cell cycle, 10 genes are related to the cell cycle process, 4 genes are related to G1/S transition of mitotic cell cycle, and 4 genes are related to G2/M specific transcription in mitotic cell cycle. After this, we found the S and S' sets from the output graph.

The results of PCPDPr when $\beta = 30$ show that the 7 TFs related to the interphase term have connections to 40 genes, which have p-value of 0.00022 with respect to interphase of mitotic cell cycle term. Among those 40 genes, YHP1, SWI4, FKH1, ACE2, CIK1, NDD1, and KIP2 are the genes most related to the interphase term. The 10 TFs related to cell cycle terms have 60 genes connected to them; those genes are related to the mitotic cell cycle term with p-value as 4.77×10^{-6}. The genes which are connected to the 4 TFs related to G1/S transition of the mitotic cell cycle were related to the same term with a p-value of 0.00568. Also, we have seen that CIK1 and KIP2 are among the genes connected to G2/M transition TFs. These two genes are related to the microtubule motor activity, which is essential for assembly of the mitotic spindle at the beginning of M phase, with p-value of 0.00463.

Similarly, we have analyzed the genes connected to TF classes when $\beta = 40$. We have found that the list of genes which are related to the interphase TFs have a significant p-value of 2.63×10^{-5} with respect to the same term. Also, we have found that the same gene list is highly related to cycline –dependent protein kinase regulator activity function with p-value of 1.47×10^{-8}. This functional term was not discovered using $\beta = 30$. Besides, the genes related to G1/S term showed to be related to the term with p-value of 0.00549.

Moreover, we have analyzed the histone cluster by Spellman [81]. This cluster has 9 genes, all of them are histone genes; they are related to chromatin assembly and disassembly GO term with p-value of 1.7×10^{-12}. We got all the genes with which the 9 histone genes interact based on our algorithm (17 genes when $\beta = 40$ and 16 genes when $\beta = 30$), and study the GO annotation related to them. For $\beta = 40$, we found that 8 out of 9 of the histone genes are among the 17 genes they bind to. This means that, most of the histone genes are found to be dependent on each other. The 17 genes are related to chromatin assembly and disassembly GO term with p-value of 2.35×10^{-9}. When $\beta = 30$, we found that 8 out of 9 of the histone genes were among the 16; these genes have p-value of 1.1×10^{-9} with respect to the chromatin assembly and disassembly GO term. For both $\beta = 30$ and $\beta = 40$, HHO1 is the gene that was not included in the 16 or 17 genes, respectively.

5.2.5 Raf Signalling Pathway

To evaluate PCPDPr with a well-known structured network, in this section we report the results for the Raf signalling pathway. Raf is an important protein for human immune system. Raf is involved in signalling proliferation of immune system cells. Raf signalling pathway has widely been studied in the literature, e.g., [24, 73, 93]. So this network has a relatively well-known structure; currently accepted Raf signalling pathway is shown in Figure 5.8.

From the study by Sachs *et al.* [73], protein concentration data from cytometry experiments is available about Raf signalling pathway. Werhli *et al.* [93] split this data into 5 data sets of 100 samples each in order to better test their inference algorithm. We also follow the same strategy

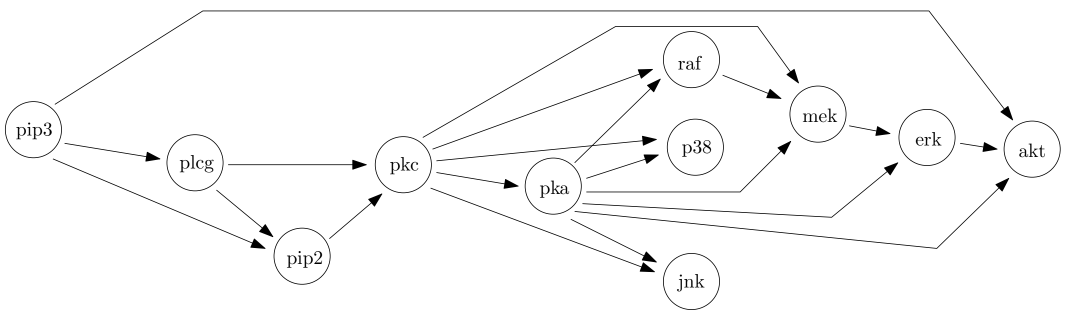
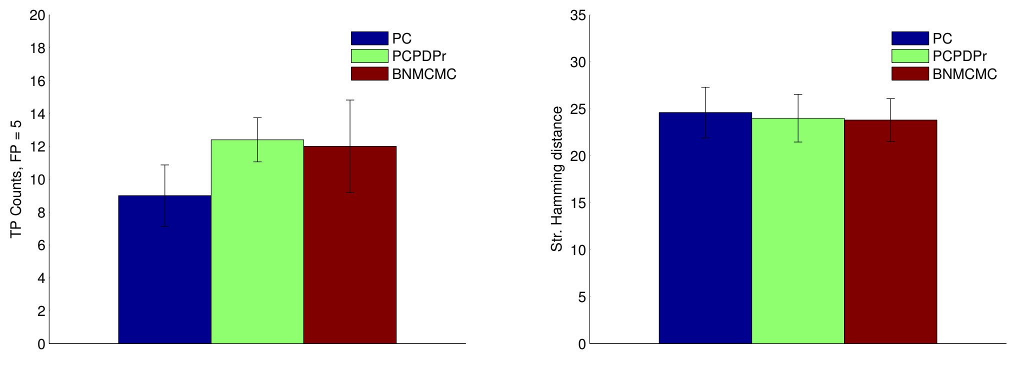


Figure 5.8: Raf signalling pathway (taken from [73, 93])

and use these 5 data sets separately and give mean results of these experiments. Here, we are using the prior data as well, which was used by Werhli *et al.* [93]. This prior data have been derived from the Kyoto Encyclopedia of Genes and Genomes (KEGG) pathway database [44]. For each pair (i, j) of genes, the number of pathways where there is an edge between the genes is divided by the total number of pathways where two genes exist together and B_{ij} is set to this ratio. If there are no pathways with an edge between the genes, then we set that entry of matrix B to 0.5, indicating no prior knowledge. The value of β is again set to 30 for this experiment.

We compared the results of our algorithm to the results of the algorithm by Werhli *et al.* [93]. Their algorithm is a bayesian network structure learning algorithm by using prior knowledge and Markov Chain Monte Carlo (MCMC) simulations. The results are shown in Figure 5.9. It is worth noting that the algorithm by Werhli *et al.* [93] is named as BNMCMC in the figure.



(a) (b)

Figure 5.9: Raf Signalling pathway results. a) gives TP counts corresponding to 5 FPs for the skeleton b) shows SHD between directed graphs.

In [93], the results are given in terms of the TP counts corresponding to 5 FPs for the undirected graphs. We also report the results here in the same way for comparability. To be able to fix the number of FPs, we found the value of α for PC and α_0 for PCPDPr that outputs a network with 5 FPs according to the currently accepted Raf pathway in Figure 5.8, and used that α values in the experiments. Again for BNMCMC we found a threshold that outputs a network with 5 FPs based on the set of sampled networks by MCMC. The results in Figure 5.9(a) demonstrates that PCPDPr outputs a more reliable network as the error bars show the standard deviation of the results for 5 different data sets. The result for directed graph evaluation is shown in Figure 5.9(b). As can be seen, the performances of PCPDPr and BNMCMC is almost the same, but we must mention here that, MCMC is a computationally very expensive procedure and BNMCMC is executed for approximately 2.6 hours on the average with the parameters suggested by the authors of [93] for each of the 5 data sets, while PCPDPr outputs these networks in only 0.32 seconds on the average.

5.3 Large-scale Gene Regulatory Network Control

The experimental results for the algorithms proposed for scalable intervention in GRNs are given in this section. Again, following the presentation of the methods in Chapter 4 they are given in two separate sections.

5.3.1 Scalable Control by Feature Reduction

This section reports the experimental results for the feature reduction method described in Section 4.1. For these experiments, we used PBNs [78, 79] as the modeling technique. The idea in PBNs is to use more than one boolean function for each target gene instead one, used in Boolean networks (see Section 2.2.2 for details)

The PBN derivation algorithm uses three parameters:

1. The number of regulators that will be chosen for each gene. Biologically, genes are thought to be regulated by few number of genes [17, 48]. So, among one, two and three gene-regulator sets, the genes with highest COD values are selected, where the error measure is the best-fit extension error [51].

2. The number of functions that will be used to model each gene. It is set to 3 based on some initial test runs that check the model's ability to predict the next state of the network given its current state.

3. The probabilities assigned to the functions chosen to model a gene. Probability $c_j^{(i)}$ which is the probability of choosing the j^{th} function for gene i, is calculated based on COD values as discussed in Section 2.2.2.

Perturbation probability [79], say p, is the probability of randomly changing the expression level of genes in the model. This way, all the states in the model become reachable, and the underlying Markov Chain corresponding to the PBN becomes ergodic [79]. Ergodicity means that the steady-state probability of a Markov Chain can be estimated empirically. More details about this parameter of the PBN can be found in [79]. In our settings for deriving the PBN, the perturbation probability p is set to 0.01.

There are two types of error measures used in this section. The first error measure is the percentage difference between the optimal value function of the original model and the value function of the policy found after feature reduction. In other words, error is the percentage difference between value functions of the policies found by following paths (a) and (b) in Figure 4.1. The second measure is the type named as simulation error. This type of error is found by simulating the model applying the interventions implied by the policy to see how well the policy does in keeping the model out of the undesirable states.

As the models used are discrete in this section, data are discretized before usage. Interval discretization with 2 bins is applied as the discretization method when necessary.

Synthetic data

We first evaluated our algorithm on some synthetic data sets generated using the algorithm proposed in [102], which is based on a regulation matrix A. Matrix A is set such that each entry a_{ij} of A gives the degree of regulation of gene j on gene i, and the diagonal of A is 1, i.e., for all i, $a_{ii} = 1$. If Y_t denotes the system state at time t, the next state is generated as follows:

$$Y_{t+1} = A(Y_t - N) + \varepsilon \tag{5.2}$$

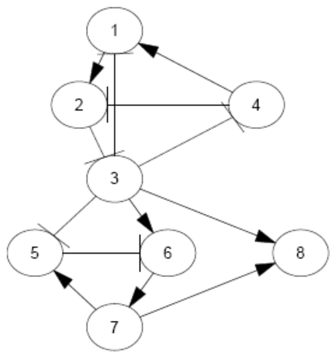
where N is the threshold that a gene has to be above (or below) in order to affect other genes, and ε is the noise uniformly distributed in a specified range. In the experiments, N is set as 50 and ε is randomly set uniformly in the range $[-10, 10]$.

To generate the data sets, we used the same parameters that are used in [102]. Setting Y_0 to random values, we generated 500 samples from each network, where one sample is taken every 5 steps of the simulation. Each a_{ij} is set to be 0.1 or -0.1 representing positive or negative regulation[6], respectively. For example, for the network shown in Figure 5.10(a), $a_{53} = -0.1$ and $a_{14} = 0.1$. In the figures, arrows denote positive regulation and the lines with a bar denote negative regulation. Expression levels are assumed to be in the range $[-100, 100]$, so in data generation, if a value goes above (below) these limits, it is set to 100 (-100). Finally, the data generated by the above simulation is discretized into binary levels (ON and OFF).

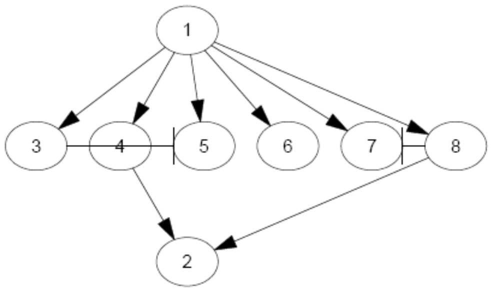
In all of the synthetic data experiments, the objective is to down regulate the second gene

[6]Notice that positive or negative regulation does not mean to always increase or decrease the expression level of the gene. The net effect depends on the value of the regulator's expression level and N.

and the first gene is intervened. For this objective, we assigned a negative reward of -5 to the states if the expression level of the second gene is 1 (ON), and a reward of 0 otherwise. There are two actions where one is the intervention of the first gene, whose cost is 1, and the other is the costless monitoring action.



(a) Network 1



(b) Network 2

Figure 5.10: Synthetic networks

The first set of data is generated from the network shown in Figure 5.10(a) that represents matrix A in Eq (5.2). As can be seen in Figure 5.10(a), there are two components in the network connected via gene 3. So, we can expect that a subset of genes, namely $\{4, 5, 6, 7\}$, can be ignored in finding a control policy because they seem to be less related to the control and reward genes, namely 1 and 2.

Table 5.5: Influence Scores of genes in network 1

Gene	3	4	5	6	7	8
Score	1.004	1.181	0.626	0.923	1.162	1.282

The *IS* values for all genes are shown in Table 5.5; these values demonstrate that the least scored genes are 5 and 6. In case Th is given as 1, the set that will be chosen is $\{5, 6\}$. Table 5.6 shows the errors associated with the different gene subsets; only subsets that have error less than 10% are listed. From the results reported in Table 5.6, it can be easily seen that $\{5, 6\}$ is one of the three best subsets.

The second set of synthetic data is generated from the network shown in Figure 5.10(b). In this network, the expression level of the second gene has to be controlled indirectly. The

Table 5.6: Gene subsets with error less than 10% for network 1

Subset	Error	Subset	Error	Subset	Error
3	2.582	5 8	2.073	3 5 8	2.245
5	2.073	6 7	0.203	3 5 7	2.245
6	0.134	6 8	0.000	3 5 6	0.173
7	0.184	7 8	0.758	5 6 7 8	0.173
8	0.026	6 7 8	0.173	3 6 7 8	0.173
3 5	2.245	5 7 8	2.245	3 5 7 8	2.245
3 6	0.173	5 6 8	0.000	3 5 6 8	0.173
3 7	2.245	5 6 7	0.173	3 5 6 7	0.173
3 8	3.012	3 7 8	2.245	3 5 6 7 8	0.173
5 6	0.000	3 6 8	0.173		
5 7	2.245	3 6 7	0.173		

Table 5.7: Influence Scores of genes in network 2

Gene	3	4	5	6	7	8
Score	2.837	2.502	1.944	2.623	1.534	2.875

Table 5.8: Gene subsets with error less than 10% for network 2

Subset	Error	Subset	Error	Subset	Error
3	3.177	3 7	1.246	3 6 7	2.004
5	0.085	5 6	4.812	3 5 7	0.858
6	3.990	5 7	0.429	3 5 6	4.235
7	0.287	6 7	2.004	3 5 6 7	2.004
3 5	2.295	5 6 7	2.004		
3 6	4.985	4 5 7	9.517		

first gene is connected to the reward gene through genes 4 and 8. This time, a subset of the genes, namely $\{3,5,6,7\}$, is expected to be the candidates for removal. The results are shown in Table 5.7 and Table 5.8. Genes 5 and 7 have the smallest scores, so they can be the candidates for removal. And if we check Table 5.8, we see that the best subset is $\{5\}$ followed by $\{7\}$ and $\{5,7\}$. This means that if Th is set as 2 for the data derived from this network, the subset $\{5,7\}$ will be chosen for removal; it is the third best subset out of 64 subsets.

Gene expression data

Metastatic Melanoma:

In this section, we report the results of the application of the gene selection algorithm to the gene expression data produced in a study of metastatic melanoma [7]. The data was also used in [67] for deriving a PBN model; 7 genes are chosen from the whole data set based on their ability to predict the states of each other; these genes are pirin, WNT5A, S100P, RET1, MART1, HADHB and STC2. The objective here is specified as down-regulating WNT5A; and pirin is used as the control gene, as in [64]. The reward function is set in the same way as in the synthetic data based experiments. The data is relatively small compared to the synthetic data sets, it has 31 samples. This can be a disadvantage for the gene selection algorithm because the information that the data contains is small compared to the synthetic data sets. Since the authors of [67] were also working on binary data, the samples were discretized to binary levels.

Table 5.9: Influence Scores of genes for melanoma data

Gene	3	4	5	6	7
Score	0.775	0.911	0.333	0.526	1.333

The results are given in Table 5.9 and Table 5.10. Although the error rates are large in this case, there is still one subset, $\{5\}$ with minimum IS, that we can remove with error less than 2% and with appropriate Th. The high error rates can be due to high degree of connectivity among the selected genes. This is consistent with the information stated above that the genes are selected based on their ability to predict each other's state. Another reason can be the possible high effects of most of the genes on the reward gene, WNT5A.

Yeast Cell Cycle:

In this section, we report the results of application of our method to a set of well-known transcription factors of budding yeast (*Saccharomyces cerevisiae*). These 11 transcription factors were previously identified to be the important regulators for the yeast cell cycle [99] : ACE2, FKH1, FKH2, MBP1, MCM1, NDD1, SKN7, STB1, SWI4, SWI5 and SWI6. The microarray data with 77 time steps that we have used in this experiment was produced by

Table 5.10: Gene subset errors of genes for melanoma data

Subset	Error	Subset	Error	Subset	Error
3	20.766	4 5	12.907	3 5 7	30.179
4	14.477	3 7	27.822	3 5 6	31.739
5	1.856	3 6	34.248	3 4 7	16.549
6	19.223	3 5	21.643	3 4 6	38.641
7	39.257	3 4	29.564	3 4 5	34.974
6 7	42.276	5 6 7	43.796	3 4 5 6	27.966
5 7	38.417	4 6 7	32.570	3 4 5 7	16.549
5 6	19.980	4 5 7	31.461	3 4 6 7	16.549
4 7	31.238	4 5 6	27.966	3 5 6 7	35.619
4 6	19.935	3 6 7	44.455	4 5 6 7	31.274

Spellman *et al.* [81]. Missing values in the data were imputed by using the *KNNImpute* software [88]. Again, before applying our method, we discretized the data set first into binary levels by interval discretization.

The reward gene is set as SWI4, which is one of the important transcription factors (part of the SBF complex) that play a role in G1 phase. Control gene is set as ACE2; it is chosen since the PBN model derived from the data has ACE2 as one of the regulators of SWI4[7]. The objective is set as down-regulating SWI4 and the reward function is set in the same way as previously described.

Table 5.11: Influence scores of genes for yeast data

Gene	FKH2	MBP1	MCM1	NDD1	SKN7	STB1	FKH1	SWI5	SWI6
Score	1.605	0.836	0.610	1.128	1.312	0.642	1.489	0.471	1.226

The *IS* scores of genes are given in Table 5.11. SWI5 is the lowest scored gene with a score of 0.471. If this gene is eliminated, an error of 1.5% occurs; see Table 5.12 for errors of some of the subsets. This error is very low and demonstrates the applicability of the method in case the threshold is chosen as 0.5. The case corresponding to elimination of SWI5 and MCM1 (with a threshold of 0.62 for instance) has an error of 5.4% which can be considered acceptable for some cases. The computational gain, however, corresponding to this error rate is huge; it

[7]Note that, to the best of our knowledge, ACE2 and SWI4 have not been identified as regulating each other. But verification of the model derived by the modeling algorithm we use here, is out of the scope of this experiment.

takes 3.32 minutes to solve when SWI5 and MCM1 are eliminated and 12.10 minutes when SWI5 is eliminated, while it takes 52.25 minutes to solve without any elimination. We discuss more time complexity below.

Table 5.12: Subset errors of genes for yeast data

Subset	Error	Subset	Error
FKH2	9.684	FKH1	5.049
MBP1	0.123	SWI5	1.506
MCM1	2.399	SWI6	1.821
NDD1	5.119	SWI5 MCM1	5.409
SKN7	2.062	SWI5 MCM1 FKH1	9.319
STB1	1.372		

Comparison to other methods

The GRN control problem has been previously studied as evident by the available corresponding literature (see Section 4.3), however scalability and feature reduction issues have not yet been considered for this problem. As mentioned before, feature reduction can also be performed after the modeling phase. But due to the computational gain of reduction before modeling, irrelevant genes are eliminated prior to the modeling phase (see Figure 4.1).

As mentioned before, control genes can be determined by using the influence concept [64, 78] which is the underlying notion for the Influence Score introduced in this study. The genes can also be eliminated after the modeling phase on path (a) in Figure 4.1 by using the influence concept. This feature reduction method eliminates genes with the lowest scores, where the score is computed as in Equation 4.8 except that instead of the $Inf(g_i, g_j)$ value, this time, Influence value from [78] is used. Notice that, this method is different from ours in the step where it applies; it is a method that can be applied given the model, i.e., it is applied after the modeling phase. Since the Influence value discussed in [78] is computed based on the PBN model; we performed a number of experiments to demonstrate how this type of elimination compares to ours. This will show the effect of elimination before the modeling step.

A structure learning (or modeling) algorithm *may* output a number of models for a given data set, where each of these models are equally likely. When this is the case, most of the modeling algorithms choose one of these models as their output. The number of equally likely methods gets smaller as the number of samples in the dataset gets larger. The PBN learning algorithm we use in this work outputs one of the equally likely models by breaking the ties randomly during the construction. To eliminate the effect of this for a fair comparison, we repeated the process of the following the paths 20 times for each data set and reported the average results, also we sampled the data sets of 1000 steps instead of the previously used 500 for synthetic networks.

A control policy can also be evaluated by simulations; starting from a random initial state, apply the policy and count the number of undesired state visits. Although comparison of value functions is more accurate, this type of evaluation provides a kind of weighted difference between value functions by eliminating the effect of states that are hardly visited. So if the steady state probability of a state in a model is small then the effect of the value function difference (if any) for that state will also be small. This type of evaluation also has the advantage of being faster compared to the value determination, as the value determination for a given policy requires very long time to execute. So, we chose the number of undesired state visits in the simulation as the evaluation metric for this experiment with 20 iterations. Each starting from a random initial state, we performed 5 simulations of 1000 steps and averaged the number of undesired state visits. The results will be given as average percentages of undesired state visits in 1000 steps.

For the experiment, we used the same 4 data sets. The control problems are defined in exactly the same way as before. To be able to make a fair comparison, we chose the threshold values so that two genes will be eliminated for each of the methods. We will call the method that is based on choosing the control gene in Pal *et al.* [64] and Shmulevich *et al.* [78] as PS referring to the names of the authors. Note that this time, Path (a) also has a feature reduction step applied after model generation (i.e. after $Model(M)$ is obtained in Figure 4.1).

Table 5.13: Comparison to PS

	$FRGC$		PS	
	Sim. Error	Time	Sim. Error	Time
Network-1	48.35	4.76	50.81	48.27
Network-2	32.30	2.67	36.53	40.60
Metastatic Melanoma	28.05	0.77	21.43	14.15
Yeast Cell Cycle	9.46	305.20	9.12	5772.83

The results are given in Table 5.13. The Sim. Error column gives the simulation errors defined above. Although PS has the advantage of directly using the model, the results demonstrate that $FRGC$ outputs comparable results to PS, sometimes even better. This shows that focusing on important parts of the model by eliminating irrelevant genes provides a reliable model reduction method for control. Only the metastatic melanoma results can be considered as significantly different, but notice that we force two-gene elimination for comparison here instead of the one-gene elimination in Section 5.3.1. The given table also includes the total time of following Path (a) with PS and Path (b) with $FRGC$ in Figure Figure 4.1. Execution time results demonstrate the computational gain of $FRGC$ compared to PS.

Time complexity

The complexity of deriving a Boolean network under the best-fit extension paradigm is given in [51] as $O(\binom{n}{k}.n.m.poly(k))$, where n is the number of genes, k is the number of predictors (regulators) for each gene, m is the number of samples in the data and $poly(k)$ is a polynomial function of k, which is in most cases equal to k. Deriving a PBN adds an additional cost of $O(\binom{n}{k}.nf.n)$, where nf is the number of functions for each target gene, because, for each gene, we are choosing nf functions out of $\binom{n}{k}$. The last two steps in path (a) of Figure 4.1, which are the construction of the MDP and value iteration, have an equal complexity of $O(a.4^n)$ for the binary case, where a is the number of actions. So, the dominating term in the total complexity of path (a) is $O(4^n)$ for $k < n$ (which is generally the case for GRNs). The complexity of computing $IS(g)$ is $O(n.m)$, since it depends on all genes other than g and the sufficient statistics for $Inf(g, g_i)$ are collected from the data in one pass. Since we are computing IS for all genes and removing the l selected genes in the algorithm, the total complexity of feature reduction is $O(n^2.m + l.n.m)$, assuming no clever data structures in shifting the columns of a multi-dimensional array. So, the total complexity of both paths in Figure 4.1 is dominated by the $O(4^n)$ term for the binary case. Even if the structured representations that may have lower average case complexity in terms of n are used in solving the MDP, the feature reduction algorithm does not dominate the overall complexity, provided that $k \geq 2$ in PBN modeling, which is usually the case.

Table 5.14: Elapsed time (in secs.) for the experiments

	Network 1	Network 2	Metastatic melanoma	Yeast Cell cycle
Path (a)	19.141	19.157	4.329	3135.258
Path (b)	1.125	1.110	1.015	199.362

The main purpose of performing feature reduction is to achieve speed-up in reaching the policy with tolerable error rate. In this sense, Table 5.14 contains the elapsed time for finding the policies for each of the data sets used in the experiments. From the results reported in Table 5.14, it can be easily seen that there is a significant decrease in time. The results here are also in correlation with the complexity analysis.

5.3.2 Scalable Control by Edge Elimination from Factored Representations

This section reports the results for the edge elimination technique proposed for scalability in control of GRNs. The details of the proposed method are discussed in Section 4.2. There are two experiments reported in this section. They are based on the Boolean models proposed for mammalian cell-cycle and human T-cell activation.

Control of mammalian cell-cycle

The first experiment is adapted from a recent study [26]. In this study, a mutation that can lead to a cancerous state is implemented in the Boolean logic model of the mammalian cell cycle. The model is relatively small and has 9 genes, where this property encouraged us to use this model, as this will demonstrate the effect of the proposed method to the optimal solutions of control problems. This model has been constructed by Faure *et al.* [27]. In [26] gene p27 is assumed to be mutated and it is always inactive. This leads to the situation where both CycD and Rb genes might be inactive (OFF), which in turn leads to unlimited proliferation. The logical rules of the mutated cell cycle model is given in Table 5.15. The notation is conventional; $\overline{X}$ represents logical NOT of X, $\vee$ and $\wedge$ represent logical OR and AND operators, respectively.

Table 5.15: Mutated cell cycle model

Product	Predictors
$CycD$	$Input$
Rb	$(\overline{CycD} \wedge \overline{CycE}) \wedge \overline{CycA} \wedge \overline{CycB})$
$E2F$	$(\overline{Rb} \wedge \overline{CycA} \wedge \overline{CycB})$
$CycE$	$(E2F \wedge \overline{Rb})$
$CycA$	$(E2F \wedge \overline{Rb} \wedge \overline{Cdc20} \wedge \overline{(Cdh1 \wedge Ubc)}) \vee (CycA \wedge \overline{Rb} \wedge \overline{Cdc20} \wedge \overline{(Cdh1 \wedge Ubc)})$
$Cdc20$	$CycB$
$Cdh1$	$(\overline{CycA} \wedge \overline{CycB}) \vee (Cdc20)$
Ubc	$\overline{(Cdh1)} \vee (Cdh1 \wedge Ubc \wedge (Cdc20 \vee CycA \vee CycB))$
$CycB$	$(\overline{Cdc20} \wedge \overline{Cdh1})$

The relationship given in Table 5.15 is temporal; the value of the *Product* column at time step $t+1$ is determined by the value of the logical formula given in *Predictors* column at t. So the set of formulae given constitute a number of different BoNs corresponding to different values of the input gene. In the cell cycle model, there is only one input gene so there are two different BoNs corresponding to $CycD$ taking values 0 and 1. From these two BoNs, we constructed a PBN with each BoN being equally probable at each step of the simulation.

Given the PBN model of the mammalian cell cycle, the undesirable states are defined as the states leading to cell cycle without any limitation and these are the ones where $CycD$ are Rb are both inactive. So the objective function is defined as: $\Phi(\{CycD, Rb\}) = CycD \vee Rb$. Based on this, we defined the reward function for the FMDP as follows:

$$R(s,a) = \begin{cases} 10 & \text{if } a = noop, (CycD, Rb) \neq (0,0) \text{ in s} \\ 1 & \text{if } a = noop, (CycD, Rb) = (0,0) \text{ in s} \\ 9 & \text{if } a \neq noop, (CycD, Rb) \neq (0,0) \text{ in s} \\ 0 & \text{if } a \neq noop, (CycD, Rb) = (0,0) \text{ in s} \end{cases} \tag{5.3}$$

Equation 5.3 reflects the fact that the cost of each action is 1 and the reward received for a desirable state is 10. Given the reward function, each of the genes in the model (except the input gene) is then considered to be the only control gene for a separate experiment. So for each experiment, the action set is composed of a 'noop' action and the action that immediately toggles the value of the control gene. Having defined all components of the FMDP this way, we solved it using the proposed reduction method. As this model is small enough for optimal solution, we used SPUDD to solve the FMDP.

The reward function defined in Equation 5.3 is for the FMDPs without reduction. If the reduction is applied then, as discussed in Section 4.2.3, this reward function is updated according to the achievable set of objectives and δ.

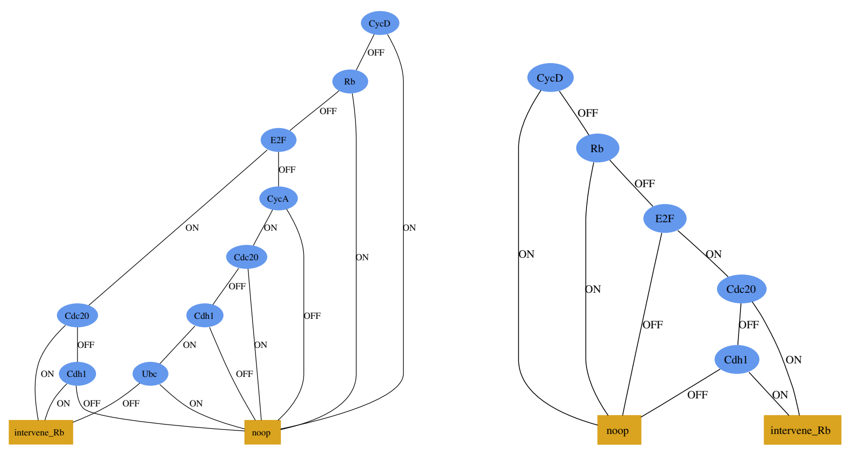
To be able to evaluate the results, we performed simulations. These simulations correspond to observing the evolution of the model under the policies computed. Starting from a random initial state, the simulation is executed for 10,000 steps, and this is repeated 10 times each starting from a new random state. The average of these 10 runs is reported in the results. A policy is evaluated according to the number of interventions performed (which gives an idea about the cost of this policy) and the number of undesirable state visits (which gives an idea on how "successful" the policy is) throughout the simulation. Without any intervention, the system stays in undesirable states in 130 out of 10,000 steps on the average. The results are given in Table 5.16; they also include optimal solutions where the reduction method is not applied (the column corresponding to $\pi_{\delta=0}$). The column with $\delta = 0.05$ gives the results of *reduceFMDP* where 0.05 is chosen as the threshold, and the column corresponding to $\pi_{\delta_{est}}$ is the one on which the reduction method is applied (*reduceFMDP2*), where δ_{est} denotes δ computed by the CT algorithm. The last value in Table 5.16 is the time elapsed to solve the problem. Running times should be interpreted keeping in mind that such a small network is hardly suitable for the analysis of the computational requirements of the reduction method. As can be seen in the results, estimating the threshold based on CT is not only is efficient in terms of time, but also outputs much better policies than *reduceFMDP*.

The results demonstrate that the policy found after edge elimination by *reduceFMDP2* is

Table 5.16: Cell cycle control, number of undesired states, number of interventions and time in secs.

Control gene	$\pi_{\delta=0}$ (no reduction)	$\pi_{\delta=0.05}$ (*reduceFMDP*)	$\pi_{\delta_{est}}$ (*reduceFMDP2*)
Rb	92.60, 14.00, 0.26	101.20, 14.60, 13.30	91.00, 12.20, 0.59
E2F	106.70, 11.40, 0.33	107.30, 36.60, 14.00	110.40, 87.10, 0.21
CycE	96.40, 12.40, 0.37	108.10, 13.10, 13.31	109.90, 14.30, 0.21
CycA	112.20, 9.20, 0.46	923.30, 855.50, 12.89	110.80, 4.60, 0.26
*Cdc*20	86.10, 27.90, 0.34	88.50, 27.40, 13.62	92.10, 28.70, 0.29
*Cdh*1	89.90, 37.90, 0.27	409.80, 366.00, 13.65	1068.70, 1035.20, 0.21
Ubc	121.80, 0.50, 0.26	433.10, 338.30, 13.15	123.90, 0.00, 0.56
CyB	83.90, 29.30, 0.44	81.40, 55.50, 14.27	105.10, 14.80, 0.24

almost as good as the optimal policy for most of the control genes. Only for one control gene (Cdh1), the policy is not as good. Rb is seen as the most effective one, and this coincides with the results reported in [26]. For a comparison, the policies corresponding to $\delta = 0$ and δ_{est} are given in Figure 5.11. Note that after the reduction procedure is applied, a simplified policy is expected as we simplify the problem. This is clearly seen in Figure 5.11; $\pi_{\delta_{est}}$ is a "generalized" version of $\pi_{\delta=0}$. This is also important as simplicity is an issue to determine the applicability of a policy in clinical practice.



(a) $\pi_{\delta=0}$ (b) $\pi_{\delta_{est}}$

Figure 5.11: Policies for control gene *Rb* for $\delta = 0$ and δ_{est}

Control of T-cell activation

A similar Boolean pathway to the mammalian cell-cycle for the activation of transcription factors (TFs) that activate T-cells is given in [49]; this model has 40 genes. So, solving this problem with MDP formalism requires very large resources as the size of the state space is 2^{40}.

T-cells form a type of white blood cells known as lymphocytes. They play an important role for immunity such that its dysfunction has severe consequences for the organism. T-cells have the ability to recognize foreign agents and subsequently eliminate them. By their T-Cell Receptor (TCR) they detect the potentially dangerous agents and then activate (and proliferate) through a signalling cascade [40].

Chronic lymphocytic leukemia (CLL) is a type of cancer caused by the uncontrolled proliferation of immunologically immature lymphocytes. ZAP-70 is an important gene in the signalling pathway of T-cell activation [40]. High ZAP-70 expression is thought to be the indicator of T-cell activation and prognosis and overall survival for CLL [36, 62]. Similarly, in T-cell activation model of Klamt *et al.* [49], if ZAP-70 is overexpressed (it is always ON), then the TFs that lead to proliferation of T-cells become always active (ON). So, in the light of these findings, we introduced a ZAP-70 overexpression mutation to the model given in [49]. ZAP-70 is, therefore, always active (ON) in our mutated model given in Table 5.17 as logical formulae. This mutation (according to our model) leads to unlimited T-cell proliferation; that is a cancerous state.

The first three genes in Table 5.17 are the input variables as given in [49], and the last four are the output TFs which activate T-cells. Having defined the PBN model of T-cell activation in the same way as the cell cycle model in the previous section, the control problem here is defined as finding an intervention strategy that avoids the activation of output TFs. So, the states that we try to avoid are those where $AP1$, CRE, $NFAT$ and $NFkB$ are all active (ON) together. The objective function, therefore, is:

$$\Phi(\{AP1, CRE, NFAT, NFkB\}) = \overline{AP1} \vee \overline{CRE} \vee \overline{NFAT} \vee \overline{NFkB} \tag{5.4}$$

We defined the reward function similarly to cell cycle model as follows:

$$R(s,a) = \begin{cases} 10 & \text{if } a = \mathit{noop}, (AP1, CRE, NFAT, NFkB) \neq (1,1,1,1) \text{ in s} \\ 1 & \text{if } a = \mathit{noop}, (AP1, CRE, NFAT, NFkB) = (1,1,1,1) \text{ in s} \\ 9 & \text{if } a \neq \mathit{noop}, (AP1, CRE, NFAT, NFkB) \neq (1,1,1,1) \text{ in s} \\ 0 & \text{if } a \neq \mathit{noop}, (AP1, CRE, NFAT, NFkB) = (1,1,1,1) \text{ in s} \end{cases} \tag{5.5}$$

Again, we should mention that the reward function in Equation 5.5 may be updated when the reduction methods are applied.

Given one of the genes as the control gene and a "noop" action, we tried to find the best

Table 5.17: Mutated T-cell activation model

Product	Predictors
$CD45$	$Input$
$CD8$	$Input$
$TCRlig$	$Input$
Ca	$IP3$
$Calcin$	Ca
$cCbl$	1
$CREB$	Rsk
DAG	$PLCg(act)$
ERK	MEK
Fos	ERK
Fyn	$(Lck \wedge CD45) \vee (TCRbind \wedge CD45)$
$Gads$	LAT
$Grb2Sos$	LAT
$lKKbeta$	$PKCth$
$IP3$	$PLCg(act)$
JNK	SEK
Jun	JNK
LAT	1
Lck	$\overline{PAGCsk} \wedge CD8 \wedge CD45$
lkB	$\overline{lKKbeta}$
ltk	$SLP76$
MEK	Raf
$PAGCsk$	$Fyn \vee \overline{TCRbind}$
$PKCth$	DAG
$PLCg(act)$	$(SLP76 \wedge PLCg(bind)) \wedge (ltk \vee Rlk)$
$PLCg(bind)$	LAT
Raf	Ras
Ras	$Grb2Sos \vee RasGRPI$
$RasGRPI$	$PKCth \wedge DAG$
Rlk	Lck
Rsk	ERK
SEK	$PKCth$
$SLP76$	$Gads$
$TCRbind$	$TCRlig \wedge \overline{cCbl}$
$TCRphos$	$Fyn \vee (TCRbind \wedge Lck)$
$AP1$	$Jun \wedge Fos$
CRE	$CREB$
$NFAT$	$Calcin$
$NFkB$	$\overline{lkB}$

control gene and intervention strategy using the proposed methods. The problem is too large to solve exactly with our current computing resources; so for an approximate solution, we used APRICODD with the size parameter set to 75. To be able to evaluate the policies, we performed simulations. We applied each policy in 10 simulations each starting from a random initial state for 10,000 steps and counted the number of undesired states (the states where all output TFs are ON) and number of interventions during the simulation. The results are reported in Table 5.18 as average of these 10 simulations; note that not all of the genes are reported in the table as control genes, rather only those that lead to good policies in terms of the simulation results. Without any intervention, the system stays in undesirable states in 9,754 steps out of 10,000. Again, we report the results with and without reduction and δ_{est} is δ computed by the CT algorithm.

Among all other control genes, ERK and MEK are the most effective ones in terms of both the number of undesired states and the number of interventions. Number of interventions here is important as it represents the cost associated with the control policy. ERK and MEK effectively stop activation of four output TFs with relatively low cost. The policies where ERK is the control gene for $\delta = 0$ and δ_{est} are given in Figure 5.12. The policy corresponding to δ_{est} is again a “simpler” version of the one for $\delta = 0$. $Raf/MEK/ERK$ pathway has been shown to be important in the development of leukemia [85]. All genes in this pathway are seen as the most effective ones in control which coincide with this finding. It is also interesting to note that the policy found where Raf is the control gene for δ_{est} is better than for $\delta = 0$. This may be due to the fact that we are using an approximate FMDP solver here (APRICODD). So simplifying the model by the reduction method proposed, leads to focusing on the parts of the model that are more important for control.

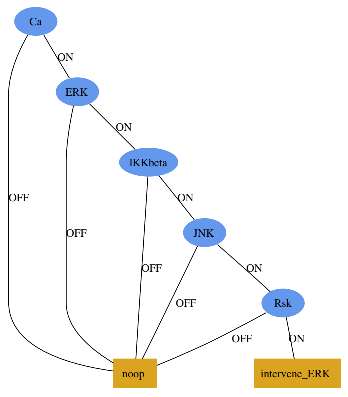
As can be recognized from the results reported in Table 5.18, reduction with the CT algorithm can provide large computational savings for most of the control genes. For instance, it takes only 1.05 seconds to find a good policy where $Calcin$ is the control gene, instead of 173.3 seconds with no reduction. For 3 genes, namely DAG, $PKCth$ and $PCLg(act)$, usage of the reduction method does not help in terms of time, but also does not effect the solution quality.

5.4 Closing Remarks

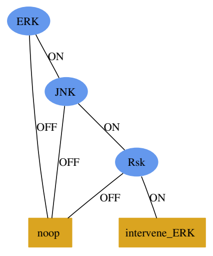
There are several data sets and models used for evaluating the proposed algorithms. The data sets used for modeling are the most widely studied data sets in the field of GRN modeling. These data sets have become the benchmark data sets for model derivation algorithms. The time-series gene expression data of Spellman *et al.* [81] covers a large number of genes of budding yeast. Most of the studies investigating yeast cell-cycle use this data set. The data

Table 5.18: T-cell activation control, number of undesired states, number of interventions and time in seconds

Control gene	$\pi_{\delta=0}$ (no reduction)	$\pi_{\delta=0.1}$ (*reduceFMDP*)	$\pi_{\delta_{est}}$ (*reduceFMDP2*)
Ca	20.00, 9648.90, 92.96	18.10, 9638.80, 41.55	20.10, 9773.50, 1.05
$Calcin$	9.70, 9608.90, 158.95	9.40, 9601.90, 41.03	10.40, 9742.60, 1.04
$CREB$	9.80, 9604.10, 157.71	8.60, 9605.00, 40.99	10.90, 9742.60, 1.08
DAG	3240.60, 3242.20, 211.24	3243.40, 3247.40, 41.12	3241.70, 3242.90, 212.06
ERK	19.60, 4865.30, 263.38	18.70, 4865.30, 41.03	21.60, 4868.60, 35.99
Fos	9.70, 9609.10, 156.64	11.40, 9588.20, 40.94	8.80, 9593.20, 1.03
$lKKbeta$	18.10, 9644.30, 87.79	18.80, 9628.40, 40.99	20.10, 9772.70, 1.05
JNK	19.80, 9643.30, 89.71	18.70, 9639.40, 41.19	19.30, 9657.10, 32.51
Jun	9.80, 9602.90, 155.98	9.30, 9580.50, 41.17	9.70, 9600.10, 1.03
lkB	9.90, 9585.70, 160.03	10.40, 9600.30, 41.22	9.90, 9743.70, 1.05
MEK	26.00, 4883.40, 121.95	27.10, 4886.20, 41.14	29.6, 4973.60, 27.15
$PKCth$	30.40, 4878.10, 126.99	30.30, 4874.70, 41.03	32.40, 4878.70, 128.49
$PLCg(act)$	2455.60, 2445.90, 211.46	2453.20, 3280.70, 41.32	2456.00, 2444.00, 222.43
Raf	3244.10, 3251.90, 217.02	3249.40, 3249.90, 41.01	39.40, 4991.20, 17.14
Rsk	20.30, 9630.60, 89.95	18.40, 9658.00, 41.05	19.90, 9768.90, 1.05



(a) $\pi_{\delta=0}$



(b) $\pi_{\delta_{est}}$

Figure 5.12: Policies for control gene ERK for $\delta = 0$ and δ_{est}

sets of Lee *et al.* [53] and Harbison *et al.* [34] are also two of the most widely used genome-wide location analysis (ChIP-chip) data sets in the literature. These data sets exhibit the common characteristics (or challenges) in biological data sets, namely having large number of genes associated with small number of samples, missing values, some amount of noise and the requirement for little or a fair number of pre-processing steps for some of the analysis. Unlike the data by Spellman *et al.*, protein concentration data by Sachs *et al.* [73] for the Raf pathway is static; it is one of the largest data sets for the Raf pathway. This diversity in the datasets utilized demonstrate the ability of the proposed algorithms to work for data sets of different types.

Compared to the modeling literature, GRN control may be considered in its infancy. Therefore, there are no data sets that can be classified as benchmarks for testing new methods. Having this in mind, for feature reduction in MDPs, again the data by Spellman *et al.* was used in addition to the small metestatic melanoma data set that was used in several papers for both discrete modeling and control. As the algorithms proposed for FMDPs do not require a data set but a model for testing, two models from two different organisms (yeast and human) were used. These models have the common property of being Boolean models. Although a Boolean model is not mandatory, for the ease of presentation and clarity, PBNs were chosen as the model in FMDP evaluations. This is one of the reasons for using these two models (cell cycle and T-cell activation models) as the testbed. Another reason is their potential involvement in the development of undesired situations. Mutations can lead to cancerous states in both models.

Some further tests can be performed to get more insights about the algorithms. It is possible to study the effect of the two thresholds for the prior knowledge to determine whether a gene is dense or not; a gene is considered to be dense if it has more connections than a given number (first threshold) with a larger probability value than a given probability (second threshold). Another test could be conducted to study limiting the maximum order for PCPDPr; there are some suggestions in the literature to determine the maximum value of the order depending on the sample size [82]. Moreover, the effect of the subjective reward function definition on the reduction algorithms for MDPs and FMDPs is an interesting aspect to investigate. All of these are on my agenda for future work.

CHAPTER 6

CONCLUSIONS

Modeling and control of GRNs is an essential problem that has received the attention of different research group as evident by the already published literature which has been reviewed in this dissertation. However, we identified certain gaps within the existing literature and successfully handled them within the scope of this dissertation. To wrap up this document, we present in this closing chapter a summary of our findings, the conclusions and the possible future research directions.

6.1 Summary and Conclusions

This dissertation investigates two important issues about GRNs which could be classified among the most essential mechanisms in order to conceive the cellular organization. These two issues are GRN modeling and control, where modeling refers to deriving a representation of the GRN and control refers to intervening the dynamics of the GRN in a way that alters the possible future states according to a certain objective.

In general, we concentrated on constraint-based structure learning algorithms for GRN modeling and especially the PC algorithm. Two modifications have been proposed for the PC algorithm to learn better networks by integrating multiple data types. This is based on the fact that most of the biological data types have some amount of noise and not abundant enough to derive all relationships between the genes. One of the data types (TF binding location - ChIP-chip) is named as the prior knowledge and is used to "direct" the search for conditional independencies through adapting the significance level in statistical tests in the PC algorithm. The data type on which these tests are performed is the microarray gene expression data. Another information derived from TF binding location data is the set of dense nodes that have a large number of connections. These nodes are handled in a special way with a greedy search algorithm to get rid of the exponential burst of the number of statistical tests.

For GRN control, we focused on the reduction algorithms that can be used to eliminate some of the irrelevant components in the data or model. This is important for scalability and for reducing the resources (cost) required to tackle the problem. We proposed a method for MDPs that removes genes at the very beginning of the process starting from the data and ending in a policy to control the GRN implied by the data. Those genes are identified to be negligible for the solution of the control problem. Other than that, for the first time, GRN control is formulated by using the FMDP framework and a method to simplify the given FMDP has been devised. The experiments showed that the solution of the simplified FMDP is a near-optimal policy for the original problem.

Synthetic and real experiments are performed to evaluate the proposed methods. The results demonstrate the applicability, effectiveness and scalability of the proposed algorithms.

6.2 Future Research Directions

First, we are working on the theoretical error bounds for the greedy separator search procedure. Although some empirical tests have been performed to evaluate the effectiveness of the method, a theoretical analysis may also be very useful. Second, we want to apply the proposed algorithms to other types of data and derive other kinds of biological networks, like protein interaction networks. Protein interaction networks are also known to include dense regions, a property that makes them a suitable candidate application area. The procedure introduced to use prior knowledge also allows for the use of prior information in an incremental way. As we have already computed prior information matrix B from some type of biological data, we argue that it should be possible that new information obtained from other sources can be added to this matrix as long as the prior knowledge can be mapped to a probability value. Of course there should be some constraints and restrictions to be taken into consideration while expanding B to cover new information sources; in other words, this is not a trivial process and should be carefully handled in order not to diverse from the main theme of having matrix B. Such an incremental extension of this work is also to be investigated. Furthermore, incorporating temporal information available in time-series microarray data into PCPDPr is also on our agenda. This should bring a new dimension into the problem and still need to be carefully investigated.

Although the gene elimination algorithm for MDPs is good at finding some less important genes, the order relationship among genes in the error rates can not be in general captured by the score function. To give exact solutions or to be able to give an error bound, the score must always be directly proportional to the error. Also, a score function that gives the score of a set of genes instead of a single gene may improve the results because summation of the scores of genes in a set may not be always proportional to the error of that set. We are also working on an automated method to determine the threshold value. Solving the constructed MDP in finite horizon is another extension that is worth further consideration; investigating the effect of the horizon on the quality of the solution can bring new insights to the problem. Finally, adapting some other biological information (pathway information for instance) while determining the genes to eliminate is also among our plans.

In the light of the findings in this thesis, we are planning to apply the developed ideas to a genome-wide control problem. This will test the scalability limits of the algorithms. But this study should wait until the existence of the genome-wide PBN model of an organism. Existing models mostly focus on certain biological components. Although the results and models presented here are not directly applicable to clinical practice yet, genome-wide solution of a control problem gives the chance to compare the policies found to real treatments, where this may lead to new insights of the applied clinical treatments and drug discovery research.

There are several types of different FMDPs in terms of the performance criterion and the length of the horizon. Also there can be constraints on the solution related to the applicability, such as the number of genes involved in the final factored policy. Finding the best simple policy depending on a set of genes whose size does not exceed a pre-defined value can be essential for the applicability of that policy as a treatment in medicine for instance. Investigating the effect of the reduction method to these other types of FMDPs is another future research direction to be investigated.

As a closing remark, it is worth emphasizing that this thesis contains the description of some novel approaches to handle the modeling and control of GRNs. The developed approaches are very promising as evident by the reported test results, the published related papers, and the identified future research directions. I do consider my efforts reflected in this dissertation as a major step in the right direction. The outcome and vision I shaped as a result of this study will definitely drive my future research for the coming years.

REFERENCES

[1] O. Abul. *Controlling discrete genetic regulatory networks.* PhD thesis, Department of Computer Engineering, Middle East Technical University, 2005.

[2] O. Abul, R. Alhajj, and F. Polat. An optimal multi-objective control method for discrete genetic regulatory networks. In *Proceedings of the IEEE Sixth International Symposium on Bioinformatics and Bioengineering*, 2006.

[3] T. Akutsu, S. Miyano, and S. Kuhara. Identification of genetic networks from a small number of gene expression patterns under the boolean network model. In *Pacific Symposium on Biocomputing*, pages 17–28, 1999.

[4] Z. Bar-Joseph. Analyzing time series gene expression data. *Bioinformatics*, 20(16):2493–2503, 2004.

[5] R.E. Bellman. *Dynamic Programming.* Princeton University Press,Princeton, New Jersey, 1957.

[6] A. Bernard and A. Hartemink. Informative structure priors: Joint learning of dynamic regulatory networks from multiple types of data. In Altman R., Dunker A.K., Hunter L., Jung T., and Klein T., editors, *Pacific Symposium on Biocomputing 2005 (PSB05).* World Scientific: New Jersey, 2005.

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

[8] C. Boutilier, R. Dearden, and M. Goldszmidt. Exploiting structure in policy construction. In *Proceedings of the Fourteenth International Joint Conference on Artifical Intelligence (IJCAI)*, pages 1104–1111, 1995.

[9] C. Boutilier, R. Dearden, and M. Goldzmidt. Stochastic Dynamic Programming with Factored Representations. *Artificial Intelligence*, 121:49–107, 2000.

[10] L.E. Brown, I. Tsamardinos, and C.F. Aliferis. A Comparison of Novel State-of-the-Art Polynomial Bayesian Network Learning Algorithms. In *Proceedings of the Twentieth National Conference on Artificial Intelligence, (AAAI)*, pages 739–745, 2005.

[11] R. Castelo and A. Roverato. A robust procedure for Gaussian graphical model search from micorarray data with p larger than n. *Journal of Machine Learning Research*, 7:2621–2650, 2006.

[12] G. Chen, S.T. Jensen, and C. Stoeckert. Clustering of Genes into Regulons using Integrated Modeling-COGRIM. *Genome Biology*, 8(1):R4, 2007.

[13] T. Chen, H.L. He, and G.M. Church. Modeling gene expression with differential equations. In *Pacific Symposium on Biocomputing*, pages 29–40, 1999.

[14] D.M. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2002.

[15] D.M. Chickering, D. Heckerman, and C. Meek. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 5:1287–1330, 2004.

[16] The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

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

[18] A. Datta, A. Choudhary, M.L. Bittner, and E.R. Dougherty. External control in markovian genetic regulatory networks : The imperfect information case. *Bioinformatics*, 20(6):924–930, 2004.

[19] M.J.L. de Hoon, S. Imoto, K. Kobayashi, N. Ogasawara, and S. Miyano. Inferring gene regulatory networks from time-ordered gene expression data of bacillus subtilis using differential equations. In *Pacific Symposium on Biocomputing*, page 17 28, 2003.

[20] A. de la Fuente, N. Bing, I. Hoeschele, and P. Mendes. Discovery of meaningful associations in genomic data using partial order correlation coefficients. *Bioinformatics*, 20:3565–3574, 2004.

[21] T. Dean and K. Kanazawa. A model for reasoning about persistance and causation. *Computational Intelligence*, 5(3):142–150, 1989.

[22] B. Deplancke, A. Mukhopadhyay, W. Ao, A.M. Elewa, C.A. Grove, N.J. Martinez, R. Sequerra, L. Doucette-Stamm, J.S. Reece-Hoyes, I.A. Hope, H.A. Tissenbaum, S.E. Mango, and A.J.M. Walhout. A Gene-Centered C. elegans Protein-DNA Interaction Network. *Cell*, 125(6):1193 – 1205, 2006.

[23] E.R. Dougherty, S. Kim, and Y. Chen. Coefficient of determination in nonlinear signal processing. *Signal Processing*, 80:2219–2235, 2000.

[24] M.K. Dougherty, J. Muller, D.A. Ritt, M. Zhou, X.Z. Zhou, T.D. Copeland, T.P. Conrads, T.D. Veenstra, K.P. Lu, and D.K. Morrison. Regulation of Raf-1 by direct feedback phosphorylation. *Molecular Cell*, 17:215–224, 2005.

[25] B. Faryabi, A. Datta, and E.R. Dougherty. On Approximate Stochastic Control in Genetic Regulatory Networks. *IET Systems Biology*, 1(6):361–368, 2007.

[26] B. Faryabi, G. Vahedi, J.F. Chamberland, A. Datta, and E. R. Dougherty. Optimal Constrained Stationary Intervention in Gene Regulatory Networks. *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2008(Article ID 620767), 2008.

[27] A. Fauré, A. Naldi, C. Chaouiya, and D. Thieffry. Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle. *Bioinformatics*, 22(14):e124–e131, 2006.

[28] N. Friedman. The bayesian structural em algorithm. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 129–138. Morgan-Kaufmann, 1998.

[29] N. Friedman, M. Linial, I. Nachman, and D. Peer. Using bayesian networks to analyze expression data. *Journal of Computational Biology*, 7:601–620, 2000.

[30] R. Givan, T. Dean, and M. Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1-2):163–223, 2003.

[31] M.R. Green. Targeting Targeted Therapy. *The New England Journal of Medicine*, 350(21):2191–2193, 2004.

[32] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient Solution Algorithms for Factored MDPs. *Journal of Artificial Intelligence Research (JAIR)*, 19:399–468, 2003.

[33] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[34] CT Harbison, DB Gordon, TI Lee, NJ Rinaldi, KD MacIsaac, TW Danford, NM Hannett, JB Tagne, DB Reynolds, J Yoo, EG Jennings, J Zeitlinger, DK Pokholok, M Kellis, PA Rolfe, KT Takusagawa, ES Lander, DK Gifford, E Fraenkel, and RA Young. Transcriptional regulatory code of a eukaryotic genome. *Nature*, 431:99–104, 2004.

[35] A.J. Hartemink, D.K. Gifford, T. Jaakkola, and R.A. Young. Combining location and expression data for principled discovery of genetic regulatory network models. In *Pacific Symposium on BioComputing*, pages 437–449, 2002.

[36] Y. Herishanu, S. Kay, O. Rogowski, M. Pick, E. Naparstek, VR. Deutsch, and A. Polliack. T-cell ZAP-70 overexpression in chronic lymphomatic leukemia (CLL) correlates with CLL ZAP-70 levels, clinical stage and disease progression. *Leukemia*, 19:1289–1291, 2005.

[37] J. Hoey, R. St-Aubin, A. Hu, and C. Boutilier. SPUDD: Stochastic Planning using Decision Diagrams. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI 99)*, 1999.

[38] J. Hoey, R. St-Aubin, A. Hu, and C. Boutilier. Optimal and Approximate Stochastic Planning using Decision Diagrams. Technical Report TR-00-05, University of British Columbia, BC, Canada, June 2000.

[39] E.L. Hong, R. Balakrishnan, Q. Dong, K.R. Christie, J. Park, G. Binkley, M.C. Costanzo, S.S. Dwight, S.R. Engel, D.G. Fisk, J.E. Hirschman, B.C. Hitz, C.J. Krieger, M.S. Livstone, S.R. Miyasato, R.S. Nash, R. Oughtred, M.S. Skrzypek, S. Weng, E.D. Wong, K.K. Zhu, K. Dolinski, D. Botstein, and J.M. Cherry. Gene Ontology annotations at SGD: new data sources and annotation methods. *Nucl. Acids Res.*, 36(Database issue):D577–581, 2008.

[40] Y. Huang and R.L. Wange. T cell receptor signaling: beyond complex complexes. *Journal of Biological Chemistry*, 279(28), 2004.

[41] M. Johnson, I. Zaretskaya, Y. Raytselis, Y. Merezhuk, S. McGinnis, and TL. Madden. NCBI BLAST: a better web interface. *Nucleic Acids Res*, 36(Web Server issue):W5–9, 2008.

[42] B. Jones and M. West. Covariance decomposition in undirected gaussian graphical models. *Biometrika*, 92(4):779–786, 2005.

[43] M. Kalisch and P. Bühlmann. Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *Journal of Machine Learning Research*, 8:613–636, 2007.

[44] M. Kanehisa and S. Goto. Kegg: Kyoto encyclopedia of genes and genomes. *Nucleic Acid Research*, 28:27–30, 2000.

[45] D.R. Karger, R. Motwani, and G.D.S. Ramkumar. On approximating the longest path in a graph. *Algorithmica*, 18:82–98, 1997.

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

[47] E. Keedwell and A. Narayanan. Discovering gene networks with a neural-genetic hybrid. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2(3):231–242, 2005.

[48] S. Kim, H. Li, E.R. Dougherty, N. Cao, Y. Chen, M. Bittner, and E.B. Sui. Can markov chain models mimic biological regulation? *Journal of Biological Systems*, 10(4):337–357, 2002.

[49] S. Klamt, J. Saez-Rodriguez, J. A. Lindquist, L. Simeoni, and Ernst D. Gilles. A methodology for the structural and functional analysis of signaling and regulatory networks. *BMC Bioinformatics*, 7(56), 2006.

[50] H. Lähdesmäki, A.G. Rust, and I. Shmulevich. Probabilistic inference of transcription factor binding from multiple data sources. *PLoS ONE*, 3(3), 2008.

[51] H. Lähdesmäki, I. Shmulevich, and O. Yli-Harja. On Learning Gene Regulatory Networks Under the Boolean Network Model. *Machine Learning*, 52(1-2):147–167, 2003.

[52] S. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[53] T.I. Lee, N.J. Rinaldi, F. Robert, D.T. Odom, Z. Bar-Joseph, G.K. Gerber, N.M. Hannett, C.T. Harbison, C.M. Thompson, I.S., J.Z., E.G. Jennings, H.L. Murray, D.B. Gordon, B.R., J.J. Wyrick, J.B. Tagne, T.L. Volkert, E. Fraenkel, D.K. Gifford, and R.A. Young. Transcriptional regulatory networks in saccharomyces cerevisiae. *Science*, 298:799–804, 2002.

[54] K. Lemmens, T. Dhollander, T. De Bie, P. Monsieurs, K. Engelen, B. Smets, J. Winderickx, B. De Moor, and K. Marchal. Inferring transcriptional modules from chip-chip, motif and microarray data. *Genome Biology*, 7(5):R37, 2006.

[55] H. Li and M. Zhan. Unraveling transcriptional regulatory programs by integrative analysis of microarray and transcription factor binding data. *Bioinformatics*, 24(17):1874–1880, 2008.

[56] A. Lindlöf and B. Olsson. Genetic network inference: the effects of preprocessing. *Biosystems*, 72(3), 2003.

[57] H. Liu, F. Hussain, C. Lim Tam, and M. Dash. Discretization : An enabling technique. *Data Mining and Knowledge Discovery*, 6:393–423, 2002.

[58] P.M. Magwene and J. Kim. Estimating genomic coexpression networks using first-order conditional independence. *Genome Biology*, 5:R100, 2004.

[59] Y. Makita, M. Nakao, N. Ogasawara, and K. Nakai. Dbtbs: database of transcriptional regulation in bacillus subtilis and its contribution to comparative genomics. *Nucleic Acid Research*, 32, 2004.

[60] S. Mangan and U. Alon. Structure and function of the feed-forward loop network motif. *PNAS*, 100:11980–11985, 2003.

[61] I. Nachman, A. Regev, and N. Friedman. Inferring quantitative models of regulatory networks from expression data. *Bioinformatics*, 20(1):i248–i256, 2004.

[62] J.A. Orchard, R.E. Ibbotson, Z. Davis, A. Wiestner, A. Rosenwald, and P.W. Thomas *et al.* ZAP-70 expression and prognosis in chronic lymphomatic leukemia. *Lancet*, 363:105–111, 2004.

[63] V. Orlando. Mapping chromosomal proteins in vivo by formaldehyde-crosslinked-chromatin immunoprecipitation. *Trends in Biochemical Sciences*, 25(3):99–104, 2000.

[64] R. Pal, A. Datta, M.L. Bittner, and E.R. Dougherty. Intervention in context-sensitive boolean networks. *Bioinformatics*, 21:1211–1218, 2005.

[65] R. Pal, A. Datta, and E. R. Dougherty. Robust Intervention in Probabilistic Boolean Networks. *IEEE Transactions on Signal Processing*, 56(3):1280–1294, 2008.

[66] R. Pal, A. Datta, and E.R. Dougherty. Optimal infinite-horizon control for probabilistic boolean networks. *IEEE Transactions on Signal Processing*, 54(6):2375–2387, 2006.

[67] R. Pal, I. Ivanov, A. Datta, M.L. Bittner, and E.R. Dougherty. Generating boolean networks with a prescribed attractor structure. *Bioinformatics*, 21(21):4021–4025, 2005.

[68] M.L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley and Sons, New York, USA, 1994.

[69] J. Quackendush. Microarray data normalization and transformation. *Nature*, 32:496–501, 2002.

[70] B. Ren, F. Robert, J.J. Wyrick, O. Aparicio, E.G. Jennings, I. Simon, J. Zeitlinger, J. Schreiber, N. Hannett, E. Kanin, T.L. Volkert, C.J. Wilson, S.P. Bell, and R.A. Young. Genome-Wide Location and Function of DNA Binding Proteins. *Science*, 290(5500):2306–2309, 2000.

[71] A. Reverter and E.K. Chan. Combining partial correlation and an information theory approach to the reverse engineering of gene co-expression networks. *Bioinformatics*, 24(21):2491 – 2497, 2008.

[72] R.I. Bahar, E.A. Frohm, C.M. Gaona, G.D. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Algebraic Decision Diagrams and Their Applications. In *IEEE /ACM International Conference on CAD*, pages 188–191, Santa Clara, California, November 1993. ACM/IEEE, IEEE Computer Society Press.

[73] K. Sachs, O. Perez, D. Pe'er, D.A. Lauffenburger, and G.P. Nolan. Causal protein-signalling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529, 2005.

[74] L. Salwinski, C.S. Miller, A.J. Smith, F.K. Pettit, J.U. Bowie, and D. Eisenberg. The Database of Interacting Proteins: 2004 update. *Nucl. Acids Res.*, 32(Database issue):D449–451, 2004.

[75] J. Schäfer and K. Strimmer. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical Applications in Genetics and Molecular Biology*, 4(1):article 32, 2005.

[76] E. Segal, Y. Barash, I. Simon, N. Friedman, and D. Koller. From Promoter Sequence to Expression: A Probabilistic Framework. In *RECOMB '02 : Proceedings of the sixth annual international conference on Computational biology*, pages 263–272, 2002.

[77] S. Shen-Orr, R. Milo, S. Mangan, and U. Alon. Network motifs in the transcriptional regulation network of *Escherichia Coli*. *Nature Genetics*, 31:64–68, 2002.

[78] I. Shmulevich, E.R. Dougherty, S. Kim, and W. Zhang. Probabilistic boolean networks: A rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, 18(2):261–274, 2002.

[79] I. Shmulevich, E.R. Dougherty, and W. Zhang. Gene perturbation and intervention in probabilistic boolean networks. *Bioinformatics*, 18(10):1319–1331, 2002.

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

[81] P.T. Spellman, G. Sherlock, M.Q. Zhang, V.R. Iyer, K. Anders, M.B. Eisen, P.O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle regulated genes of yeast saccharomyces cerevisiae by microarray hybridization. *Mol. Biol. Cell*, 9:3273–3297, 1998.

[82] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search.* MIT Press, Cambridge, MA, 2000.

[83] R. St-Aubin, J. Hoey, and C. Boutilier. APRICODD: Approximate Policy Construction using Decision Diagrams. In *Advances in Neural Information Processing 13 (NIPS 2000)*, 2000.

[84] C. Stark, BJ. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and M. Tyers. BioGRID: a general repository for interaction datasets. *Nucleic Acids Res*, 2006 Jan 1;34(Database issue):D535–9, 2006.

[85] L.S. Steelman, S.L. Abrams, and J. Whelan *et al.* Contributions of the Raf/MEK/ERK, PI3K/PTEN/akt/mTOR and Jak/STAT pathways to leukemia. *Leukemia*, 22:686–707, 2008.

[86] Y. Tamada, H. Bannai, S. Imoto, T. Katayama, M. Kanehisa, and S. Miyano. Utilizing evolutionary information and gene expression data for estimating gene regulations with bayesian network models. *Journal of Bioinformatics and Computatitonal Biology*, 3(6):1295–1313, 2005.

[87] R. Thomas, S. Mehrotra, E.T. Papoutsakis, and V. Hatzimanikatis. A model-based optimization framework for the inference on gene regulatory networks from DNA array data. *Bioinformatics*, 20(17):3221–3235, 2004.

[88] O. Troyanskaya, M. Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B. Altman. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17(6):520–525, 2001.

[89] I. Tsamardinos, L.E. Brown, and C.F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, 2006.

[90] E. van Nimwegen. Scaling laws in the functional content of genomes. *Trends in Genetics*, 19(9):479–484, 2003.

[91] T. Verma and J. Pearl. Equivalance and synthesis of causal models. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 220–227, 1990.

[92] D.C. Weaver, C.T. Workman, and G.D. Stormo. Modelling regulatory networks with weight matrices. In *Proceedings of Pacific Symposium on Biocomputing*, pages 112–123, 1999.

[93] A.V. Werhli and D. Husmeier. Reconstructing Gene Regulatory Networks with Bayesian Networks by Combining Expression Data with Multiple Sources of Prior Knowledge. *Statistical Applications in Genetics and Molecular Biology*, 6(1):article 15, 2007.

[94] Wikipedia. DNA microarray — wikipedia, the free encyclopedia, 2008. [http://en.wikipedia.org/wiki/DNA_microarray; accessed 22-September-2008].

[95] A. Willie and P. Bühlmann. Low-order conditional independence graphs for inferring genetic networks. *Statistical Applications in Genetics and Molecular Biology*, 5(1), 2006.

[96] A.S. Wilson, B.G. Hobbs, T.P. Speed, and P.E. Rakoczy. The microarray : potential applications for ophthalmic research. *Molecular Vision*, 8:259–270, 2002.

[97] X. Wu and Y. Ye. Exploring gene causal interactions using an enhanced constraint-based method. *Pattern Recognition*, 39(12):2439 – 2449, 2006.

[98] R. Xu and D.C. Wunsch II. Gene regulatory networks inference with recurrent neural network models. In *Proceedings of International Joint Conference on Neural Networks*, pages 286–291, 2005.

[99] Y. Yang, J. Suen, M. Brynildsen, S. Galbraith, and J. Liao. Inferring yeast cell cycle regulators and interactions using transcription factor activities. *BMC Genomics*, 6(1):90–104, 2005.

[100] Y.H. Yang, S. Dudoit, P. Luu, and T.P. Speed. Normalization for cDNA microarray data. *Nucleic Acid Research*, 30(4), 2002.

[101] C. Yoo, V. Thorsson, and G.F. Cooper. Discovery of causal relationships in a gene regulation pathway from a mixture of experimental and observational DNA microarray data. In *PSB'02*, pages 498–509, 2002.

[102] J. Yu, V. Smith, P. Wang, A. Hartemink, and E. Jarvis. Using bayesian network inference algorithms to recover genetic regulatory networks. In *International Conference on Systems Biology (ICSB02)*, 2002.

[103] M. Zampieri, N. Soranzo, and C. Altafini. Discerning static and causal interactions in genome-wide reverse engineering problems. *Bioinformatics*, 24:1510–1515, 2008.

[104] X. Zhou, X. Wang, R. Pal, I. Ivanov, M. Bittner, and E.R. Dougherty. A bayesian connectivity-based approach to constructing probabilistic gene regulatory networks. *Bioinformatics*, 20(17):2918–2927, 2004.

[105] J. Zhu, B. Zhang, E.N. Smith, B. Drees, R.B. Brem, L. Kruglyak, R.E. Bumgarner, and E.E. Schadt. Integrating large-scale functional genomic data to dissect the complexity of yeast regulatory networks. *Nature Genetics*, 40:854–861, 2008.

VITA

PERSONAL INFORMATION

Surname, Name: Tan, Mehmet

Nationality: Turkish (TC)

Data and Place of Birth: 3 August 1977, Erzurum

Marital Status: Married

Phone: +90 312 2105523

Fax: +90 312 2105544

EDUCATION

Degree	**Institution**	**Year of Graduation**
MS	METU Computer Engineering	2003
BS	METU Computer Engineering	2000
High School	Atatürk Anatolian High School	1995

WORK EXPERIENCE

Year	**Place**	**Enrollment**
2007-2008	University of Calgary	Visiting scholar
2005-2006	AGMLAB	Software engineer
2000-2005	METU Computer Engineering Dept.	Research assistant
1998-1999	TÜBİTAK-BİLTEN	Part-time programmer

PUBLICATIONS

International Journal Publications

M. Tan, M. Alshalalfa, R. Alhajj and F. Polat, "Influence of Prior Knowledge in Constraint-Based Learning of Gene Regulatory Networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics,* (forthcoming).

M. Tan, R. Alhajj and F. Polat, “Automated Large-Scale Control of Gene Regulatory Networks,” *IEEE Transactions on Systems, Man, and Cybernetics-Part B,* (forthcoming).

International Conference Publications

M. Tan, F. Polat and R. Alhajj, “Large-Scale Approximate Intervention Strategies for Probabilistic Boolean Networks as Models of Gene Regulation,” *Proceedings of IEEE Symposium on Bioinformatics and Bioengineering,* Oct. 2008.

M. Tan, M. Alshalalfa, F. Polat and R. Alhajj, “Combining Multiple Types of Biological Data in Constraint-Based Learning of Gene Regulatory Networks,” *Proceedings of IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology,* Sep. 2008.

M. Tan, F. Polat and R. Alhajj, “Feature Reduction for Gene Regulatory Network Control,” *Proceedings of IEEE Symposium on Bioinformatics and Bioengineering,* Oct. 2007.

National Conference Publications

F. Polat, A. Coşar, S. Girgin, M. Tan, E. Çilden, M. Balcı, E. Kapusuz, V. Koç, A. Yavaş, Ç. Ündeğer. Küçük Ölçekli Harekatın Modellenmesi ve Simülasyonu. In *USMOS’05 I. Ulusal Savunma Uygulamaları Modelleme ve Simülasyon Konferansı*, 2005.