Learning Gene Network Using Bayesian Network Framework

Liu Tiefei

(*Bachelor of Medicine*)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Department of Computer Science

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2005

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisors, Dr. Sung Wing-Kin, Dr. Mao Pei-Lin and Dr. Liu Bing, for providing me with the wonderful opportunity to pursue my PhD degree. I am grateful to them for their continuous encouragement, support and guidance throughout of years of my study.

I am thankful to the graduate supervisory committee overseeing my work, Dr. Tung Kum Hoe and Dr. Lee Wee Sun for their constructive suggestions and critical comments.

Special thanks go to Dr. Wu Ping and Dr. Ankush Mittal for their guidance as well as helpful suggestions. Madam Leong Yoke Yee is also highly appreciated for helping me refine the thesis.

I thank all past and present members of the computational biology lab for their idea sharing. The wonderful time we have spent together in NUS will be in my mind forever. My heartfelt appreciation goes to my beloved parents for their constant support and encouragement, without whom this would have remained but a dream. Finally, my deepest gratitude goes to my wife for her unconditional love, understanding and warm support through the years.

Liu Tiefei

National University of Singapore

October 2005

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Summary

Learning gene networks is one of the central problems in molecular biology. In recent years, with enormous microarray data becoming available, learning gene network has received increasing attention, becoming one of the hottest topics in computational biology. Many models and learning methods have been proposed to solve the problem. However, the data problem and the complexity of gene regulatory systems make learning difficult. Moreover, some important biological factors which are critical to gene regulatory systems are not considered in most published works. There factors include: various time delays among gene regulatory systems, the effects of complexes and the effect of proteins as hidden variables when learning a gene network from microarray data. In this thesis, three models and learning methods are proposed to take into account the important biological factors: 1) The time delayed model is proposed to capture the various time delays among gene regulatory systems. A corresponding learning algorithm, the Time Delayed Network Learning (TDNL) algorithm, is proposed to learn the structure of a network. 2) Conditional dependance is used to find the collaborations among regulators and the effect of a complex is considered by the learning algorithm, the Conditional Dependance (CD) learning algorithm. 3) Proteins are modeled as hidden variables in the network by Semi-Fixed Network. The Semi-fixed Structure Expectation Maximization (SSEM)

algorithm, is proposed to learn the structure of a network. The effectness of the proposed methods are verified by experiments on both artificial and real-life gene expression data. The performance comparison of these methods against some published methods prove the advantages of the proposed methods.

CHAPTER 1

Introduction

1.1 Genes and Gene Networks

1.1.1 Genes and Biological Sequences

Although a cell is the fundamental unit of all living organisms, it is complicated in terms of both structure and function. Such complexities are mainly embodied in and regulated by three biological sequences: DNA, RNA and Protein.

DNA (DeoxyriboNucleic Acid) is a linear, double stranded unbranched polymer in which the monomeric subunits are four chemically distinct nucleotides (Adenine (A), Cytosine (C), Guanine (G), Thymine (T)). DNA is the carrier of genes and other regulatory information. A gene is a piece of DNA fragment which contains genetic information. The whole set of genes in a cell, called the genome, defines the structure and function of the cell.

The functions of genes are implemented via proteins, which are linear polymers composed of 20 different types of amino acids. Proteins play a central role in virtually all aspects of cell structure and functions. The sequence and function of

a protein is defined by the sequence of a corresponding gene in nature, while the expression strength, the expression place and the expression time of the protein is regulated by a set of other genes.

The genetic information between genes and proteins are linked by mRNA(messenger Ribonucleic Acid). RNA is a linear, single stranded polymer of 4 different types of nucleotides (Adenine (A), Cytosine (C), Guanine (G), Uracil (U)). A RNA copies the genetic information of a gene by transcription. After that, some RNA translates the information into proteins. This RNA is a mRNA. Therefore, genes contain coding information for encoding proteins and RNA molecules information.

1.1.2 Gene expression, gene regulation and regulatory pathways

The flow of genetic information from a gene to an RNA and a protein is called the gene expression process. In this process, DNA serves as the template to make RNA. This process is known as transcription where information determined by the nucleotide sequence is transferred from a double stranded DNA molecule to a single stranded RNA molecule. RNA then serves as the source of information to make proteins in a process called translation. Here, the nucleotide sequence information in RNA is converted to an amino acid sequence of proteins. The language of RNA is translated into the language of proteins.

The expression of a gene is controlled by some other genes. This process is called gene regulation. Gene expression is regulated both temporally and spatially [60]. The temporal expression of a gene refers to the process that a gene expresses (or is regulated) at the appropriate time and keeps itself silent otherwise [60]. It also indicates a gene has different expression patterns at different times [51]. For example, the expression pattern of human globin genes are different at different stage of the development [65, 98]. There is also spatial control of gene expression [60]. Although cells from the same organism have identical genomes, cells in the

different parts of an organism may have different gene expression patterns due to the various functions they fulfill. Therefore, the regulation of gene expression is an essential part of life [38]. There are two types of regulations: positive and negative. Given two genes A and B, if an expression level of B is affected by the expression level of A, we say A regulates B. If an increase in the expression level of A leads to the increase of expression level of B, it is a positive regulation; otherwise, it is a negative expression.

Regulatory events in a set may depend on each other and form some regulatory chains, named regulatory pathways. Moreover, a chain may form a loop. There are two types of loops: negative and positive. If the sum of the negative regulations of the loop is odd, it is a negative loop; otherwise, it is positive. Regulatory loops are important to an organism as they maintain the stability or development of cells [96]. It is necessary to understand the gene regulation system.

1.1.3 Gene networks

It is estimated that each gene on average interacts with four to eight other genes, and is involved in 10 biological functions [24]. The complexity of a living cell is achieved by the concerted activity of many genes and their products. This activity is often coordinated by the organization of the genome into regulatory modules, or sets of co-regulated genes that share a common function[83]. The global gene expression pattern is therefore the result of the collective behavior of individual regulatory pathways. In such highly interconnected cellular signaling networks, gene functions depend on the cellular context. Genes (proteins) work together as a team to accomplish certain processes that no single protein can do alone, such metabolism, detoxification, and various responses to the environment. This partially explains why many novel "gene targeted" drugs have failed during clinical trials–because of side effects and poor specificity[1]. Thus, understanding a gene network as a whole is

[1]What is a gene network: http://www.gene-networks.com/english/technology/page1.html

essential, and learning gene networks is an important central theme in post genomic research [24, 30, 42]. This thesis aims to contribute to the study of gene networks.

There are several applications and advantages to studying gene networks:

- Gene networks provide a large-scale, coarse-grained view of the physiological state of an organism at the mRNA level [13]. Gene networks describe a large number of interactions in a concise way. They also present the dynamic properties of the gene regulatory system. They are capable of being the annotation of genomics and functional genomics data.

- It is an important step to uncover the complete biochemical networks of cells [13].

- Knowledge about gene networks might provide valuable clues for the therapeutics of complex diseases [13, 30].

- As most phenotypes are the result of the collective response of a group of genes, gene networks help to explain how complex traits arise and which groups of genes are responsible for them [13, 30].

- Gene networks are well suited for comparative genomics [13]. Comparing gene networks from different genomes helps with the understanding of evolution.

1.2 Motivation behind the thesis

Early work on learning gene regulation takes the biological experimental approach. This traditional approach is an inherently local one: examining and collecting data on a single gene, a single protein or a single reaction at a time, then analyzing the binding sites and reactions one by one. It normally takes one or more years to discover the regulation of a gene. Over the years, this manual approach has made remarkable achievements, allowing us to make highly accurate biochemical

models of some individual genes for some small sized genome, such as the bacteria phage lambda [23]. However, taking into account the huge information stored in a genome (there are thousands or even tens of thousands of genes in a genome [25]), it is far from possible to construct a network by the conventional way. The number of experiments that are necessary for constructing a network is simply too many. In addition, s network learnt by the experimental approach can only describe the regulation relationship. It would not have the ability to predict the properties not observed. To obtain the full picture of even a medium size genome using the experimental approach is not only time consuming but also expensive [23]. Therefore, it is unrealistic to obtain an understanding of a global regulation profile by analyzing regulation pathways one by one [23, 25].

With the development of high-throughput genomics and functional genomics, massive data on thousands of cellular species are being gathered. This is a significant shift from the traditional molecular biology approach of focusing on single molecules and reactions. The need is now data-driven, and there is great urgency to find methods that can handle the massive data in a global manner and that can analyze large systems at some intermediate level [23].

At the time, computer science shows that inferring a logical regulatory network is possible with genome-wide gene expression data solely [23]. A gene network can be modeled by various mathematical methods. A model is a representation of reality used to simulate a process, understand a situation, predict an outcome, or analyze a problem. The success of the computational approach in learning gene networks has been proved biologically, with many exciting results reported in recently published literature [30, 53, 87].

Early computational approaches operate based on learning the relationships among genes either by studying mutual information or the correlation among their expression values. The representatives of such approaches are pair-wise interaction [6] and clustering [100], which seek to directly find correlations among genes. Since

then, Boolean networks [2, 67] have been used in several works, where gene expression levels are represented by Boolean values and the gene regulatory relationship is represented by a set of Boolean functions. Linear and nonlinear models followed, and they represent regulatory relationships by linear functions and non-linear functions. In a well-known paper by Friedman et al. [34], an algorithm for learning gene networks using the Bayesian network is presented. Since then, several extensions of the Bayesian network have been proposed, such as the Bayesian network integrated with nonparametric regression [48], Dynamic Bayesian network (DBN) [75], etc. A detailed literature review will be presented in Chapter 2.

However, when learning gene networks using time series gene expression data, biologically significant results have so far been obtained from only some small datasets. Scientists have failed to learn gene networks from medium or large datasets because they seem to have overlooked some important biological factors including: variable time delays in the gene regulatory system, the effect of proteins in the regulatory system, the special collaboration style of genes/proteins, and so on. A detailed description of these issues will be presented in Chapter 3. New methods which take into consideration these biological factors are clearly necessary.

1.3 Contribution of the thesis

To resolve the problems outlined above, I propose three models and corresponding learning algorithms:

- **A learning framework based on the Bayesian network enhanced with a various time delayed model.** Most research in learning gene networks either assumes that there is no time delay in gene expression or that there is a constant time delay. I present how the Bayesian network can be applied to represent variable time delay relationships by a various time delayed model. A set of improvements are made to increase the efficiency and accuracy of the

learning process: (1) an improved mutual information calculation method for measuring the dependence between two genes; (2) a new structure learning algorithm which is suited to learning gene networks; (3) an approximation method for joint probability inference to speed up the learning process. In addition, unlike the traditional Bayesian network, the proposed framework can represent and detect directed loops that commonly occur in cell cycles.

- **Bayesian network enhanced by conditional dependence.** Traditional Bayesian network learning methods often use pairwise correlations to help find regulatory relationships [34]. However, it is reported that less than 20% regulation relationships can be found by pairwise correlation in some widely used microarray datasets [29]. Barrier exist, preventing the use of pairwise correlations to infer large regulatory pathway[29]. I propose using the descendant-based conditional dependence together with the mutual information to extract more regulation relationships. Basically, if two genes g_1 and g_2 regulate the same target gene g, due to the combined effect of g_1 and g_2, it is expected that g_1 and g_2 are dependent given the gene g. This idea enables us to find regulation pairs without strong correlations.

  Proteins often form complexes before they regulate other genes. A complex is a set of physically interconnected proteins[2]. Such physical interaction is necessary for the functions of proteins. Therefore, the complex form is an important factor in learning gene network. It demands the parent selection method should be based on multiple gene combinations instead of single ones. However, few existing methods consider this important factor. I identify complexes through the hidden information provided in datasets, then treat a complex as a unit of a candidate parent. A learning algorithm called Conditional Depen-

[2]It is represented by a set of strongly interacting genes in a simplified gene network; a simplified gene network indicates that proteins are omitted in the regulatory system.

dence Learning algorithm is proposed to extract the underlying structure of a gene network based on the idea. Compared to the time delayed network, this method is more appropriate for learning a big size gene network as it is not iterative and there is no random sampling.

- **A semi-fixed Bayesian network.** This models a gene network as a directed graph with hidden variables, representing the combined effects of collaborated proteins. In this model, the number of hidden variables is predefined using biological knowledge. In addition, the relationships between hidden variables and observed variables are partially fixed. A Semi-Fixed Structure Expectation Maximization (SSEM) algorithm is proposed to learn such networks. $SSEM$ employs the advantages of the semi-fixed structure of the model. It makes the Bayesian network decomposable, thus the learning is efficient. Various time delay is also integrated in the model. Compared to the other two methods, the semi-fixed model is closer to the real gene regulatory process and gives better learning performance. However, it is slow since it needs more computing resource to infer the parameters.

1.4 Organization of the thesis

Chapter 2 gives a brief literature review covering the research area. Some existing models and algorithms for learning gene networks are introduced. Some important but rarely mentioned biological knowledge is discussed in Chapter 3; this knowledge forms the basis of my current research. Chapter 4 models a gene network as a time delayed network and describes a time delayed network learning algorithm to learn the model. Chapter 5 describes the use of conditional dependence in learning gene networks. Chapter 6 describes a semi-fixed model with hidden variables as well as the Semi-Fixed Structure Expectation Maximization algorithm. Chapter 7

concludes the thesis and gives some further perspectives to the project.

CHAPTER 2

Review and Background

As early as the 1960s, some mathematical formalisms were proposed to describe gene regulatory networks [54]. Traditionally, the emphasis has been on simulation techniques [25] instead of structure reconstruction. With more experimental data available, automatic structure reconstruction techniques are gaining popularity. In recent years, the number of papers in learning gene networks has grown exponentially.

In this chapter, I briefly review: 1) the data collection methods: microarray, and 2) the mathematical techniques in reconstructing gene networks from microarray gene expression data.

2.1 Data collection

We may observe a gene network is observed by monitoring expression levels of its elements. Thus, to learn a gene network, one important prerequisite is the availability of the expression data of elements in the gene network.

The main measurable variables in the gene regulation system is the level of

protein synthesized and mRNA transcribed. A widely used method to measure protein level is 2D-PAGE which separates proteins on a two-dimensional sheet of gel, first in one direction based on their isoelectric point, and then in the other direction based on their molecular weight. The result is a two-dimensional image with a large number of protein "spots". The intensity of each spot is proportional to the amount of the specific protein present. However, the sensitivity and accuracy of this method are not high enough to identify all proteins. Besides, the high time expense makes it difficult to obtain a genome-wide scale profile using this method. Meanwhile, mRNA levels are measurable on a genome wide scale using the new DNA microarray technology. Thus, the only available large scale data for learning gene regulation is mRNA data, which represent gene expression levels. Therefore, most learning methods are based on gene transcriptional data, with few using protein levels. In the following, I briefly describe the microarray technology.

Traditional methods to measure gene expression levels in molecular biology such as Northern Blot, RNAase Protect Assay, Reverse Transcription-Polymerase Chain Reaction (RT-PCR) generally work on a "one gene in one experiment" basis, which means that the throughput is very limited and the "whole picture" of all genes in a cell is difficult to obtain. In the past several years, a new technology called DNA microarray, has attracted tremendous interest among biologists. This technology could monitor the whole genome on a single chip, giving researchers can have a better picture of the interactions among thousands of genes simultaneously.

A microarray is a glass (sometimes nylon) chip, onto which single stranded DNA molecules are attached at fixed locations (spots). There may be tens of thousands of spots on an array, each representing a single gene. The sample mRNA is labeled with fluorescent dye. They may hybridized with the DNA(known as probes) attached on the chip. If the mRNA is complementary to these probes, then the mRNA will stay at the spot. The amount of hybridized mRNA can be measured by imaging equipment according to the fluorescence intensities [14, 71]. An experiment

with a single DNA chip can provide researchers information on thousands of genes simultaneously – a dramatic increase in throughput.

There are two main variants of the DNA microarray technology, depending on the types of single stranded DNA being plotted: oligonucleotide microarray and cDNA microarray: Oligonucleotide microarray employs oligonucleotides (20∼80-mer oligos) or peptide nucleic acid (PNA) probes. These probes are synthesized either in situ (on-chip) or by conventional synthesis followed by on-chip immobilization. cDNA microarray employs cDNA probes (500∼5,000 bases long) which are immobilized to a solid surface such as glass using robot spotting. cDNA microarray is cheap to manufacture and easy to read compared to oligonucleotide microarray. However, cDNA microarray requires a large number of cDNAs to process the experiment.

There are two main types of gene expression microarray data: static and time series microarray data. In static expression experiments, a snapshot of the expression of genes in different samples is measured while in time series expression experiments, a temporal process is measured. Since gene expression is a temporal process [94], we choose time series data as the data source to learn a gene regulatory network in this thesis.

The microarray (DNA chip) technology has significant impact on genomics study. Many fields, including drug discovery and toxicological research, should benefit from the use of DNA microarray technology. The main shortcoming of microarray is that the measured values are not quite accurate, i.e., microarray data is noisy.

2.2 Literature Review

2.2.1 Pair-wise Methods

Pair-wise methods seek to discover the relationships among genes by pair-wise comparisons solely. They do not take into account interactions where the expression of one gene is achieved by the combined effects of multiple other genes. Arkin et al. [6]

proposed correlation metric construction (CMC). CMC computes the magnitude of gene pairs by cross-correlation. A distance matrix is constructed for each gene pair by comparing their similarities to other genes. Then a diagram is constructed to summarize the strength of interaction and predict mechanistic connections between the genes. Chen et al. [16] proposed activation/inhibition networks to find regulation based on whether peaks in one signal precede peaks in another signal. Chen et al. proposed grouping the genes with similar expression profiles. Then a prototype is generated for each group of genes by averaging the expression values of genes in the group. Each prototype represents a group of genes with similar expression patterns and is represented as a series of peaks. The correlations between prototype pairs are calculated to determine the type of regulatory relationships (activation, inhibition or unmatched) and measure the strength of the regulatory relationship between any two prototypes. Finally, the regulation matrix is generated by the scores.

2.2.2 Clustering

One of the main problems that hinder research on gene network reconstruction is the dimension problem, i.e. there are many genes with a few replicates. A useful approach is to cluster genes with similar expression patterns into clusters, then infer the regulatory relationship among the clusters. Researchers believe genes with similar expression patterns have similar functions or are involved in the same biological events [100]. Currently, several clustering methods are used for this purpose. Different clustering methods can generate very different results. Each combination of distance measurement and clustering algorithm tends to emphasize a different type of regularities in the data. There is no single criterion for choosing the best clustering method. How to choose the method depends on the particular emphasis.

Given clusters, there are also several methods to find the interactions among them. Wahde and Hertz [100] clustered 65 genes from rat CNA datasets into four "waves" using the FITCH hierarchical clustering algorithm. Then, by a genetic

algorithm, they built a four-nodes continuous time recurrent neural network. Chen et al. [16] reduced 3131 yeast genes into 308 clusters by average linkage clustering. Then, they used simulated annealing to optimize a qualitative network based on the timing of peaks in the data. Someren [88] reduced 2467 yeast genes into $t-1$ clusters and represented each cluster by a "prototype" gene calculated from the cluster. A linear model of the prototype genes is then generated by linear regression. Guthke et al. [40] proposed grouping genes into clusters, then find the representative genes for the clusters. The connections among the representative genes are modeled by differential equation. Toh et al. [97] proposed averaging the gene expression values of each cluster, then discover the regulatory relationships by Graphical Gaussian Modeling (GGM).

2.2.3 Boolean model

In a simplified way, gene expression level can be roughly represented as a binary state: either active (on,1) or inactive (off,0). The interactions among genes can be represented by Boolean functions which calculate the state of a gene from the activation of other genes regulating it. The result is a Boolean network. This idea was popular in the 1960s [56, 92, 93, 101], and has been incorporated into numerous recent papers [2, 3, 45, 61, 67, 89]. Somogyi and Sniegoski [89] showed that Boolean networks have features similar to those in biological systems, such as global complex behavior, self-organization, stability, redundancy and periodicity. The general approach of a Boolean network is to discretize gene expression values into Boolean values, then find a set of Boolean functions which describe the state changes of each gene. Liang et al. [67] proposed REVEAL (REVerse Engineering ALgorithm) to resolve the problem. REVEAL uses information theoretic principles to reduce the search space and establish how the given genes are connected in the networks, and then determines the functions that specify the interactions among genes. REVEAL needs to enumerate all possible state transitions to build a Boolean network. To de-

crease complexity, a maximum fan-in, k ($1 \leq k \leq n$ where n is the number of genes in the dataset), is applied to each gene. For each gene, all possible subsets with less than k genes are considered to be its candidate regulators. If a subset is found fully determining the state changes of the given gene, it is said to be the regulator of the gene. An implementation of the algorithm proved to be capable of reliably reproducing networks with $n = 50$ and $k = 3$ given 100 state transition pairs (out of 10^{15} possible pairs). working on a similar idea, Akutsu et al. [2] proposed a simpler algorithm which proves that only $O(\log n)$ state transition pairs (from 2^n pairs) are necessary and sufficient to identify the original Boolean network of n genes with high probability. Furthermore, Akutsu et al. [4, 5] extended a Boolean network to a qualitative network to model a gene network. Corresponding algorithms have also been proposed to learn the qualitative model. Shmulevich et al. [84] introduced the Probabilistic Boolean Network, which shares the appealing rule-based properties of Boolean networks, but are robust in the face of uncertainty.

Boolean networks allow large regulatory networks to be analyzed in an efficient way by making strong simplified assumptions on the structure and the dynamics of a genetic regulatory system [54]. It is a good starting point for realistic modeling of gene networks [107]. However, the system oversimplifies the gene regulation system and assumes the transitions to take place simultaneously, which is not the usual case in reality.

2.2.4 Linear model

Based on the assumption that the expression level of a gene at one time point is the weighted sum of expression levels of all genes at the previous (or current) time point, a gene network can be modeled as a set of linear equations. A linear genetic network directly models the effects of the combination of different input genes by means of a weighted sum of their expressed levels. The weight represent the relationships among genes. Zero weights indicate the absence of interaction and a positive or

negative weights corresponds to stimulation or repression. The absolute value of a weight corresponds to strength.

A general linear genetic network model is represented in the following equations [88]:

$$x_i(t+1) = \sum_{j=1}^{J} W_{i,j} x_j(t) \tag{2.1}$$

or [103]:

$$x_i(t) = \sum_{j=1}^{J} W_{i,j} x_j(t) \tag{2.2}$$

where $x_i(t)$ is the gene expression level of gene i at time instance t and $W_{i,j}$ is the influence weight of control of gene j on gene i.

2.2.5 Differential equation

Using a differential equation to model a gene network is computationally more intensive and requires the assumption of specific kinetic schemes. However, using smaller timesteps and continuous variables, a differential equation may get a more accurate physical representation of a gene network [15, 87].

A popular model is the linear differential equation [17]. Chen et al. [17] proposed a linear differential equation to model gene expressions. Both transcription and translation are modeled in the dynamic system by kinetic equations with feedback loops from translation product proteins to transcription, and incorporating the degradation of proteins and mRNAs; the system is as follows:

$$\frac{dr}{dt} = f(p) - Vr \tag{2.3}$$

$$\frac{dp}{dt} = Lr - Up \tag{2.4}$$

where the variables are functions of time t and defined as follows:

n: Number of genes in the genome

r: mRNA concentrations, n-dimensional vector-valued functions of t

p: Protein concentrations, n-dimensional vector-valued functions of t

f(p): Linear transcription functions, n-dimensional vector polynomials on p

L: Translational constants, $n \times n$ non-degenerate diagonal matrix

V: Degradation rates of mRNAs, $n \times n$ non-degenerate diagonal matrix

U: Degradation rates of Proteins, $n \times n$ non-degenerate diagonal matrix

Two methods are employed to construct the model from experimental data: Minimum Weight Solutions to Linear Equations (MWSLE), which determine the regulation by solving under-determined linear equations, and Fourier Transform for Stable Systems (FTSS), which refines the model with cell cycle constraints. Several extended models, the RNA model, the Protein Model and the Time delayed model have also been proposed.

S. Watanabe [102] and Hideyuki Maki [72] proposed an S-System to infer a gene network from sets of time-course data, each of which has resulted when a specific is disrupted. They proposed that the expression level of a gene is computed by the power-law function:

$$\frac{d}{dt}X_i = \alpha_i \Pi_{j=1}^{n} X_i^{g_{ij}} - \beta_i \Pi_{j=1}^{n} X_j^{h_{ij}} \tag{2.5}$$

where n is the total number of state variables or reactants. g_{ij} and h_{ij} are the interactive affectivity of X_j to X_i. The first term represents all influences that increase X_i whereas the second term represents all influences that decrease X_i. α

and β are some positive coefficients. The parameters are inferenced by genetic algorithm [72].

Wahde and Hertz [100] built a non-linear differential equation based on continuous-time recurrent neural networks:

$$\tau_i \hat{x_i} + x_i = g(b_i + \sum_j w_{ij} x_j) \tag{2.6}$$

where for i = 1,...,n, τ_i^{-1} is a rate constant, x_i is the expression level and $\hat{x_i}$ is its derivative with respect to time. For a set of n genes, there are $n \times n$ weights (w_{ij}), n bias terms (b_i) and n time constants τ_i. The equation can be extended to include higher order terms in a network. The non-linear activation function g is defined as:

$$g(z) = \frac{1}{1 + e^{-kz}} \tag{2.7}$$

k is set to 1 in the paper. A genetic algorithm is used to determine the parameters of the network.

2.2.6 Modeling gene networks with the Bayesian network

In recent years, several models based on the Bayesian network have been proposed for learning gene networks [33, 34]. Because of its suitability for learning gene networks, the Bayesian network is one of the most widely used models in the research area nowadays. In this section, I will define what a Bayesian network is, how a Bayesian network is learnt, and finally, how gene networks are learnt using a Bayesian network.

Bayesian Network

In probabilistic reasoning, random variables are used to represent events and/or objects in the world. A random variable can be thought of as the numeric result of operating a non-deterministic mechanism or performing a non-deterministic experiment to generate a random result. Computing the joint probabilities of given random variables requires the probabilities of every instantiation combination which is combinatorially explosive. Chain rule simplifies it by the following form:

$$P(X_1, ..., X_n) = \Pi_{i=1}^{n} P(X_i | X_1, ..., X_{i-1}) \tag{2.8}$$

or

$$P(X_1, ..., X_n) = \Pi_{i=1}^{n} P(X_i | X_{1+1}, ..., X_n) \tag{2.9}$$

where $X_1, ..., X_n$ are random variables.

- **Example:** As shown in Figure 2.1, the probability of the variables A, B, C, D and E can be represented as follows:
  $P(A, B, C, D, E) = P(A|B, C, D, E)P(B|C, D, E)P(C|D, E)P(D|E)P(E)$.

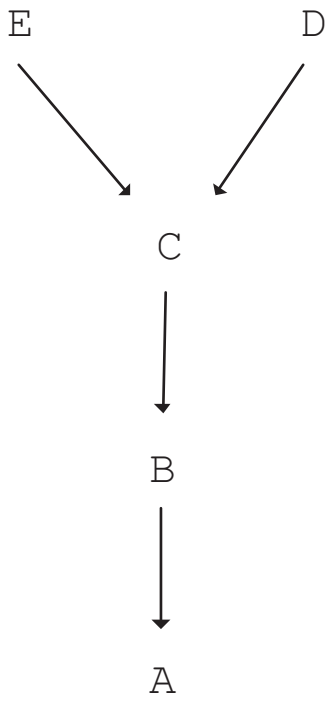

Figure 2.1: An example of a directed acyclic graph.

Bayesian networks (belief networks) take this process further by making the important observation that certain random variable pairs may become uncorrelated once information concerning some other random variable(s) is known. If $P(A|X_1, ..., X_n, U) = P(A|X_1, ..., X_n)$, it can be interpreted that A is determined by $X_1, ..., X_n$ regardless of the random variable U. With these conditional independencies, it is possible to simplify the computation of joint probabilities.

A Bayesian network is defined as follows:

- **A Bayesian network** is an annotated directed acyclic graph that encodes a joint probability distribution over a set of random variables $\mathcal{X} = \{X_1, ..., X_n\}$, where each X_i has a set of discrete values or continuous values. Formally, a Bayesian network for $\mathcal{X}$ is represented by $\mathcal{B} =< \mathcal{G}, \Theta >$ where $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ is a directed acyclic graph, $\mathcal{V} = \{V_1, ..., V_n\}$ is the vertex set and $V_i \in \mathcal{V}$ corresponds to a random variable X_i, $\mathcal{E} = \{e_1, ..., e_m\} \subset \mathcal{G} \times \mathcal{G}$ is the edge set and $e_i = (v_x, v_y) \in \mathcal{E}$ is a dependence between v_x and v_y, and $\Theta = \{\theta_1, ..., \theta_n\}$ is the parameters sets storing the conditional joint probability distribution over $\mathcal{X}$ and $\theta_i = \theta_{X_i|Pa(X_i)}$ is the conditional probability distribution of X_i given all the parents $Pa(X_i)$(denoted by $P(X_i|Pa(X_i))$. Each variable X_i is independent of its non-descendant(s) given all of its parents are instantiated in $\mathcal{G}$.

In the network $\mathcal{G}$, any joint distribution can be decomposed in the product form:

$$P(X_1, ..., X_n) = \Pi_{i=1}^{n} P(X_i|Pa(X_i)) \tag{2.10}$$

where $Pa(X_i)$ are the parents of X_i in $\mathcal{G}$.

- **Example:** As shown in Figure 2.1, the probability of the variables A, B, C, D

and E can be represented as:

$P(A, B, C, D, E) = P(A|B)P(B|C, D)P(C)P(D|E)P(E)$.

Given $\mathcal{D}$ and the corresponding structure $\mathcal{G}$, the parameter set Θ can be estimated [11] by encoding Θ in a prior distribution $P(\Theta)$. The distribution is then updated using $\mathcal{D}$, hereby obtaining the posterior distribution $P(\Theta|\mathcal{D})$ by applying Bayes' rule:

$$P(\Theta|\mathcal{D}) = \frac{P(\mathcal{D}|\Theta)P(\Theta)}{P(\mathcal{D})} \tag{2.11}$$

Based on Equation 2.10, θ_i can be estimated independently.

Learning Bayesian Network

Given $\mathcal{D}$, the problem of learning a Bayesian network structure can be stated as follows: Given a training set $\mathcal{D} = \{D_1, ..., D_n\}$ of independent instances of $\mathcal{X}$, find a network $\mathcal{B} =< \mathcal{G}, \Theta >$ that best explains $\mathcal{D}$. There are two main approaches for finding structures. The first approach learns a Bayesian network as a constraint satisfaction problem [36, 43, 77, 82, 86, 91]. In this approach, properties of conditional independence among variables are estimated by a statistical hypothesis test, such as χ^2-test [77]. A network is then built to exhibit the observed dependencies and independencies. The second approach, which is more popular, learns a Bayesian network as an optimization problem [21, 34, 36, 44, 48]: A statistically motivated scoring function, termed scoring metrics, such as *Minimum Description Length* (*MDL*) [62] or *Bayesian score* [21, 44], is introduced to evaluate a network with respect to $\mathcal{D}$, and the optimal network according to this score is computed. *Bayesian Score*($Score_B$) [34] is a popular score metrics, and it is defined as follows:

$$Score_B(\mathcal{G} : \mathcal{D}) = logP(\mathcal{G}|\mathcal{D}) \tag{2.12}$$

$$= log\frac{P(\mathcal{D}|\mathcal{G})P(\mathcal{G})}{P(\mathcal{D})} \tag{2.13}$$

$$= logP(\mathcal{D}|\mathcal{G}) + logP(\mathcal{G}) + C \tag{2.14}$$

where $P(\mathcal{D}|\mathcal{G}) = \int P(\mathcal{D}|\mathcal{G}, \Theta)P(\Theta|\mathcal{G})d\Theta$ is the marginal likelihood which averages the probability of the data over all possible parameter assignments to $\mathcal{G}$ and $C = logP(\mathcal{D})$ is a constant independent of $\mathcal{G}$. The particular choices of priors $P(\mathcal{G})$ and $P(\Theta|\mathcal{G})$ for each $\mathcal{G}$ are important to avoid over-fitting and to determine the exact Bayesian score.

An important property of Bayesian score or Minimum Distance Length (MDL) is decomposability in the presence of some priors:

$$Score_B(\mathcal{G} : \mathcal{D}) = \sum_i Score_b(X_i|Pa(X_i) : D) \tag{2.15}$$

It is infeasible to compute maximum likelihood as it involves computing marginal likelihood $P(\mathcal{D}) = \sum_G P(\mathcal{D}, \mathcal{G})$ which is the sum over an exponential number of models. Bayesian Information Criterion (BIC) is proposed to approximate the posterior:

$$logP(\mathcal{G}|\mathcal{D}) \approx logP(\mathcal{D}|\mathcal{G}, \hat{\Theta_G}) - \frac{logN}{2}\triangle G \tag{2.16}$$

$$logP(\mathcal{D}|\mathcal{G}) = \sum_i logP(X_i|Pa(X_i) : D) \tag{2.17}$$

where N is the number of samples, $\triangle G$ is the dimension of the models (the number of free parameters if $\mathcal{D}$ is fully observed, ie. without hidden variables) and $\hat{\Theta_G}$ is the Maximum Likelihood (ML) estimate of the parameters.

The decomposability of the score is crucial to learning a Bayesian network. With it, the learning problem, which is known to be NP-hard, can be solved by a set of local searches.

When applying the Bayesian network to gene network learning, there are some advantages compared to other methods:

- Bayesian networks are particularly useful for describing processes composed of locally interacting components [34]. That is, the value of each component directly depends on the values of a relatively small number of components.
- Statistical foundations for learning Bayesian networks from observations, and computational algorithms to do so are well understood and have been used successfully in many applications [34].
- Bayesian networks provide models of causal influence [34]. Although Bayesian networks are mathematically defined strictly in terms of probabilities and conditional independence statements, a connection can be made between this characterization and the notion of direct causal influence.
- Because of its firm statistic basis, the Bayesian network can deal with the stochastic aspects of gene expression and the noisy measurements of microarray data in a natural way [106].
- Bayesian networks are able to handle a large number of variables with only a few replicates [34, 48]. It is especially useful when learning gene networks, since microarray data generally have thousands or even tens of thousands genes but only tens of replicates. Besides, Bayesian networks are capable of estimating the confidence of different features in networks [34]. The absence of data often leads to the consequence that many networks explain the data equally well.

The confidence is useful for measuring to measure whether a statistic feature of the network is likely to be true.

- Learning gene networks is NP-hard [16]. The decomposability [34] of Bayesian networks ensures local searches achieve global optimization, thus making the learning easier.
- Hidden variables in a network and missing values in gene expression data are easy to handle with the Bayesian network. Many methods have been established for in learning Bayesian network with latent variables and missing values.
- Bayesian networks can capture many types of relationships among genes: linear, non-linear, combinatorial, stochastic and other types [106]. It remains unclear which types of relationships a gene regulatory system may pursue. The ability of Bayesian networks to grasp various types of relationships makes it appropriate for learning gene networks.

Learning Gene Network Using Bayesian Network

Friedman [34] proposed modeling a gene network as a Bayesian network: Each gene is a vertex and each regulatory relationship is an edge in the Bayesian network. As learning a sparse network is technically difficult, Friedman [34] proposed a two-step algorithm, The Sparse Candidate algorithm, to learn the structure and parameters: For each gene, (1) some candidate parents who are likely to be the parents of the target gene are selected; (2) The score Bayesian score for every possible subset of the candidate parent set is computed, and the best combination is searched for. In the first step, a general method using pair-wise correlation, such as Mutual Information (MI), is applied to find the genes with high dependence with the target genes. However, some dependence cannot be measured by MI. Thus, some weak parents are generated. Weak parents are parents to a target gene but does not have high dependence with the target gene. *Kullback-Leibler* (KL) divergence is used in the work, which can be improved iteratively using the learnt network as the prior

knowledge in the iterated learning process, to find better dependence between gene pairs. The second step can be done by some heuristic method such as hill climbing [34]. Friedman showed that the results obtained by the sparse candidate learning algorithm are biologically meaningful results by examining the results with a set of statistic measurements: robust test, order relation, Markov relation, and so on.

Since then, many works based on the Bayesian network frame work have been proposed, and biologically relevant results have been obtained. Hartemink [41] extended Friedman's work by adding these annotations to edges: "+", "-" or "+/-" which represent positive, negative or unknown regulation. Beal et al. [8] proposed including the unmeasured genes as the hidden factors to learn a gene network. They proposed implementing the step by state-space models (SSMs). Lee et al. [64] proposed a modularized learning approach based on the assumption that most genes are likely to be related to other genes in the same biological modules rather than the genes in different modules. They proposed finding overlapping modules in the genes, and learning the subnetworks in modules with a Bayesian network. Zhou et al. [108] proposed constructing the probabilistic gene regulatory networks that emphasize network topology using a reversible jump Markov chain technique. Rogers et al. [80] proposed inferencing the regulatory networks by the Bayesian regression approach, which works with continuous variables directly. Murphy & Mian [75] and Gransson & Koski [39] used the Dynamic Bayesian Network (DBN), which is an extension of the Bayesian network, to model gene networks. In this model, a gene at a time point is regulated by its parent in the previous time point. Thus, the acyclic limitation of the Bayesian network is overcome in DBN. Murphy et al. gave a thorough report in [75] on the application of DBN in learning gene networks. Imoto et al. [48] and Kim [57] further extended Bayesian networks and DBN by integrating nonparametric regression into the models, so that the methods can use continuous gene expression values instead of the discrete values in the general Bayesian network approaches. Their method is capable of capturing the non-linear relationships among

genes. Yu et al. [106] presented an influence score to measure the magnitudes of regulatory strength of the edges. It is useful for eliminating the false positives as well as distinguishing the positive or negative regulation of edges.

With more and more works using Bayesian networks as the framework to tackle the gene network reconstruction problem [30, 42, 78, 83], the Bayesian network is becoming a widely used approach in learning gene networks.

CHAPTER 3

Research Directions

Various types of gene regulatory network models have been proposed. In this chapter I will describe some important biological factors which are critical in the choice of network reconstruction methods.

3.1 Preliminaries

There are several issues mentioned by existing works.

3.1.1 Sparse network

It is estimated that each gene is regulated by four to eight genes, and is involved in about 10 biological events [88]. Based on this estimation, in a gene network with n genes, there are about $4n - 8n$ regulatory edges. Therefore, a gene network is a sparse network. Learning sparse networks is technically difficult as the search space is too big [34]. This thesis proposes several learning algorithms to effectively learn a gene network.

3.1.2 Regulatory feedback loops

A regulatory system contains many types of regulatory loops. A negative feedback loop tends to slow down a process while a positive feedback loop tends to accelerate it. Negative feedback helps maintain stability in a system in spite of external changes while positive feedback amplifies possibilities of divergences [96] to give the system the ability to access new points of equilibrium. The presence of positive and negative feedback loops is necessary for the maintenance of multiple steady states for biological system [19]. Therefore, the regulatory loops are important to the gene network. In our model, regulatory loops are taken into account with the incorporation of time lags in the gene regulatory system.

3.1.3 Stochastic nature

Stochastic effects have been shown to play an important role in cellular processes from gene regulation to signal transduction and metabolic pathways [74]. These effects, termed as molecular noise, give rise to a probabilistic description of system dynamics (chemical master equation), where reactions occur as discrete random events (Markov process). From this aspect, the stochastic model is more appropriate for learning gene networks. As existing analysis tools based on deterministic approaches are inadequate or inapplicable, there is a need for the development of formal analysis for stochastic systems. I choose Bayesian networks as the basic framework in the thesis. As I have described in Chapter 2, Bayesian networks are capable of handling the stochastic events.

3.1.4 Boolean/discrete or continuous

Continuous values reflect the real expression values of genes. Thus, continuous expression values are more meaningful in learning gene expressions and regulations. However, the high computational expense is a drawback of continuous values. The

noise in microarray data is another problem of using continuous values. On the other hand, using discrete values can simplify implementation and presentation.

Furthermore, by observation, most genes are in binary state in most time [9, 28, 46, 55, 58, 69, 70]; they are either expresses in maximum strength or silence. In practice, Boolean values have been proposed by numerous works [2, 3, 61].

Based on the above points, in this thesis, I use discrete expression values, and in my experiments, each gene takes on 2 expression values, i.e., Boolean values. Even now, the models proposed in the thesis are not limited to Boolean models but discrete models.

3.2 More directions

There are some important facts that have been ignored or overlooked in most published works.

3.2.1 Various time delays

Within regulation procedures, various events occur at different steps. Usually, the step of transcription (from DNA to mRNA) is quick while the time for translation varies from protein to protein [66]. Besides, the protein-DNA regulation is an accumulation process and the threshold differs for different regulation pairs [66]. Therefore, regulatory time delay differs among different regulation pairs. When learning a gene network from synchronized time serial data, such information is critical to obtaining the whole picture of the network.

Researchers have tried to incorporate such information into their models and assume that time delay is constant. Some works which make this assumption are: Someren [88], which learns a gene network by modeling it as a linear model; Murphy & Mian [75] and Gransson & Koski [39] where the dynamic Bayesian network (DBN) is used to model time delay in the gene network.

Some researches [17] have shown that different gene pairs have different time delays for gene regulation. Chen et al. [17] attempted to address the varying time delay issue of a gene network. They proposed modeling the regulation process and the various time delays using differential equations. However, their algorithm is of high computational complexity and no real experiments have been done based on their model.

I propose to model the various time delays by a time delayed model. A corresponding algorithm, the Time-Delayed Network Learning ($TDNL$) algorithm, is also proposed to learn the structure of the network.

3.2.2 Collaborations among regulators

The one-to-one parent-child dependence between a candidate regulator and the target gene has been used in many works to find the regulators of a gene [36]. However, it is reported that less than 20% regulations pairs have strong dependence in some datasets[29]. So, it is impossible to find all regulators of a specific gene by using parent-child dependence solely. Based on the theory of descendant based conditional independence, it is feasible to find the dependence among parents. Thus, we can find more regulators.

3.2.3 Complex

During the regulation process, the protein products of a set of genes form a complex before they interact with the target gene [73]. A complex is a physically interconnected protein combination. Loss of any element in a regulatory complex may result in lose of function in the whole complex. In addition, the proper function of the complex relies on the physical interaction of the components of the complex[73]. In a simplified gene regulatory network, proteins are omitted and represented by the corresponding genes. Thus, a complex is represented by a set of strongly interacting genes in the simplified gene network. The strong interaction demands the

parent selection method be based on multiple gene combinations instead of single ones. However, few existing methods consider this important fact. I propose finding a complex by the strong interaction among the genes in the complex. Then the complex can be used as a single candidate parent unit in the learning process.

3.2.4 Hidden variables

The regulation of genes includes controlling transcription, post-transcription, translation, post-translational protein degradation, and other processes [59]. Each regulation step should be taken into consideration of the gene network to be accurately described [59]. Currently, except for gene expression data (at the mRNA level), other data types are still absent in the genome scale or proteome scale. Regarding all components as hidden variables is impractical since learning the structure with so many hidden variables is difficult. I propose modeling a gene network with hidden variables as a Semi-Fixed Bayesian network. In the network, the regulator proteins of a gene are modeled as one hidden variable. A learning algorithm named semi-fixed EM is proposed to learn such a model.

CHAPTER 4

Learning Various Time Delay Gene Networks with the Time Delayed Bayesian Network

4.1 Introduction

Most research work in learning gene networks assumes that either there is no time delay in gene expression or there is a constant time delay. However, the biological literature [63] shows that different gene pairs have different time delays for gene regulation. To the best of my understanding, Chen et al. [17] is the only research group which has attempted to incorporate the various time delays factor into the gene network learning process[1]. They proposed modeling the regulation process and the various time delays using differential equations. However, their algorithm is of

[1]A recent paper [109] also mentioned the various time delays problem. The authors find the various time delays among genes by measuring the time difference of the initial expression change between a gene pair. However, the paper is published in 2005 while my work on learning gene networks with various time delays was finished in 2004. Therefore, in the rest the thesis, I do not mention it.

high computational complexity and no real experiments have been done based on their model.

To address the issue of various time delays in the gene regulatory system, I propose a learning framework based on Bayesian network enhanced with a various time delays model. As I described in Chapter 2, Bayesian network has been applied to discover gene networks and shown to have advantages over the other methods [34, 48, 75]. However, no work has been done to apply Bayesian network to address the various time delays issue of the gene networks. One possible explanation is that learning time-delay model introduces too many variables and thus makes the learning intractable.

In order to deal with various time delays problem, I propose a number of improvements to make the learning process more efficient and accurate: (1) an improved mutual information calculation method for measuring the dependency between two genes; (2) a random sampling approach to find more weak parents; (3) a new structure learning algorithm which is suited for learning a sparse network such as a gene network. Comparison with other methods shows that this algorithm detects more correct edges and is able to discover the time delays of the edges. In the real-life gene expression datasets to be discussed in Section 4.4, this method obtains 80% more correct edges than other methods do. In addition, unlike other methods, this framework can represent and detect directed loops that commonly occur in cell cycles as shown in experimental results on artificial and real-life gene expression datasets. Such loops are important for the regulations of global gene expression and stage-specific functions to produce a continuous cycle of cellular events [37, 90].

The rest of this chapter is organized as follows: The time delayed Bayesian Network and its suitability in modeling a gene network is discussed in Section 4.2. Section 4.3 describes the learning algorithm for a time delayed network. Experimental results are presented in Section 4.4, followed by the conclusion in Section 4.5.

4.2 Time delayed network and its transformation to the traditional Bayesian network

4.2.1 Time delayed network

It is well known that for a pair of genes g_i and g_j (suppose g_i regulates g_j), the change of expression level of g_i affects the expression level of g_j after a certain time interval. For example, in yeast, a gene $MCM1$ regulates another gene $CLN3$. Based on the gene expression microarray dataset by Spellman [90], each time the expression level of $MCM1$ changes, the expression level of $CLN3$ correspondingly changes about 30 minutes later.

Time delay intervals are different for different gene regulatory pairs. For example, human TNF-α and iNOS genes are regulated by AP-1 and NF-κB1. Their delays in expression after the activation of AP-1 and NF-κ1 are three and six hours, respectively [63]. It is further known that there should be an upper limit for the time delay in a gene network since the length of a cell cycle is limited.

The regulation of genes can form feedback loops (for example, $g_1 \rightarrow g_2 \rightarrow \ldots \rightarrow g_1$), which exist in many metabolism pathways and are critical in maintaining the stability of a gene network [19]. However, the traditional Bayesian network framework fails to represent and learn feedback loops.

To model a gene network, I propose the time delayed network which can capture various time delay relationships as well as discover directed loops spanning at least one time slice.[2] The time delayed network is an extension of Bayesian network. The time delayed network is defined as follows:

> Let k be the maximum time delay allowed for each regulation. A time delayed network can be described by $N =< G, \theta, \delta >$:

[2]The available gene expression datasets capture the gene expression levels of genes every seven to thirty minutes. Since the time interval between consecutive time slices is short, only a few loops can be formed within the same time slice.

- $G =< V, E >$ is a directed graph, where $V = \{V_1, V_2, \ldots, V_n\}$ is the set of variables of G, and E is the set of directed edges of G. Each variable V_i represents a gene, and each edge (V_i, V_j) represents the regulation process from V_i to V_j.
- For every edge $(V_i, V_j) \in E$, $\delta(V_i, V_j)$ represents the unique time delay for the edge (V_i, V_j). Note that $\delta(V_i, V_j)$ is a integer and $k \geq \delta(V_i, V_j) \geq 0$.
- θ is the parameter set of G that stores the conditional probability distribution $Pr(V_i \mid P_a(V_i))$ for every $V_i \in V$, where $P_a(V_i)$ is the parent set of V_i in G.
- A directed cycle is allowed if at least one of its edges has the time delay ≥ 1. Figure 4.1(a) shows an example of a directed cycle with four genes in a time delayed network.

4.2.2 Relationship between traditional network and time delayed network

Given a maximum time delay k (k is smaller than the time of a cell cycle), a variable at a time slice can only be affected by variables in the current time slice and the previous k time slices. For each variable V_i, let $V_{i,0}, V_{i,1}, \ldots, V_{i,k-1}, V_{i,k}$ be its states in the previous k time slices and the current time slice. Learning whether the edge (V_j, V_i) has a time delay Δ is equivalent to learning whether $(V_{j,k-\Delta}, V_{i,k})$ is an edge. The formal transformation is described as follows: Given a time delayed network $N =< G, \theta, \delta >$ where $G =< V, E >$, with the maximum time delay k, N can be represented using a traditional network $M =< H, \theta' >$ such that:

- $H =< V', E' >$, where V' is the vertex set and E' is the edge set.
- $V' = \{V_{i,t} \mid V_i \in V, t = 0, 1, \ldots, k\}$. Thus, each vertex $V_i \in V$ is transformed to $k + 1$ vertices $\{V_{i,0}, ..., V_{i,k}\}$.

- Consider a variable $V_i \in V$, with $Pa(V_i) = \{V_{i_1}, \ldots, V_{i_s}\}$ being the parent set of V_i in G. In H, the variable $V_{i,k}$ has s parents $V_{i_1,(k-\Delta_1)}, V_{i_2,(k-\Delta_2)}, \ldots, V_{i_s,(k-\Delta_s)}$ where Δ_j is the time delay $\delta(V_i, V_{i_j})$ associated with the edge between V_i and V_{i_j}. In the parameter set θ', the conditional probability distribution $Pr(V_{i,k}|V_{i_1,(k-\Delta_1)}, \ldots, V_{i_s,(k-\Delta_s)})$ of M is the same as the conditional probability distribution $Pr(V_i|V_{i_1}, \ldots, V_{i_s})$ of N.

Figure 4.1 shows an example of the transformation. It can be easily verified that the transformed network M is a directed acyclic graph and that the network M contains all the parameters of N. Once the network M is learnt, the parameters of the network N can be easily recovered. It is obvious that if the time delay $k = 0$, the time delayed network is indeed a traditional Bayesian network, and if $k = 1$ the time delayed network is a dynamic Bayesian network (DBN). Thus, the learning algorithm to be discussed later is also applicable to the learning of traditional Bayesian networks and DBNs.

A work related to my proposed model is k-DBN model. k-DBN was proposed by Boyen et al.[12] for finding hidden variables in a network. Though k-DBN was not used for learning causal relationships as is the case in a gene network, it can be extended to learn the structure of a gene network, allowing more than one edge with different time delays from gene g_i to gene g_j[3]. It is not the same as my method since I allow only one edge between a gene pair as time delay is unique for a gene pair.

[3]P. Dagum & A. Galper [22] and Tucker [99] also proposed models similar to k-DBN which admit multiple edges between a gene pair.

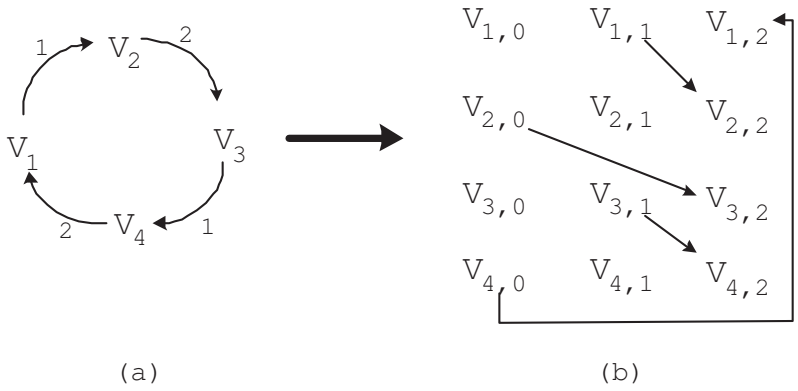

Figure 4.1: An example of network transformation is shown here. (a) The time delayed network contains four variables & four edges. The integer on each edge indicates the time delay, and the maximum time delay k is assumed to be 2. This network has one cycle: $V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_4 \rightarrow V_1$. (b) The transformed network contains 12 variables and four edges. Each variable V_i is transformed into three variables: $V_{i,0}$, $V_{i,1}$ and $V_{i,2}$. The edge (V_i, V_j), with time delay Δ, is transformed into edge $(V_{i,k-\Delta}, V_{j,k})$. For example, the edge (V_1, V_2) with time delay 1 is transformed into the edge $(V_{1,1}, V_{2,2})$. After the transformation, no cycle exists.

4.2.3 Dataset Transformation

Following the network transformation, the original training dataset D_N for the time delayed Bayesian network N needs to be transformed correspondingly. D_N is a time-series dataset with n variables $V_1, \ldots, V_n$. Each variable V_i is described by its states in m time slices, i.e., $v_{i,1}, v_{i,2}, \ldots, v_{i,m}$.

Recall that M is a Bayesian network with $(k+1)n$ variables, namely, $V_{1,0}, \ldots, V_{n,0}$, $V_{1,1}, \ldots, V_{n,1}, \ \ldots, \ V_{1,k}, \ldots, V_{n,k}$. Its training data can be expressed as a set of $(k+1)$ n-dimensional vectors. Given D_N with m time slices, I transform it into $(m-k)$'s training samples for M. The $(m-k)$'s training samples for M are denoted as D_M. Precisely, for $t = 1, 2, \ldots, m-k$, the t-th sample for D_M is $(v_{1,t}, \ldots, v_{n,t}, v_{1,t+1}, \ldots, v_{n,t+1}, \ldots, v_{1,t+k}, \ldots, v_{n,t+k})$. Figure 4.2 gives an example of the transformation from D_N to D_M.

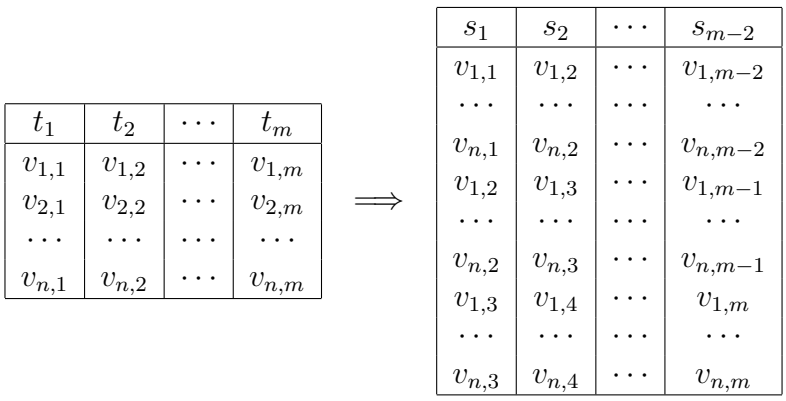

Figure 4.2: An example of dataset transformation. (a) This is the original dataset with n variables and m time slices. $v_{i,t}$ represents the state of the variable V_i at time slices t. Suppose the max delay k is 2. (b) This is the transformed dataset. The new dataset contains $n \times 3$ variables. Each variable V_i is transformed into 3 variables.

4.3 Time delayed network learning algorithm

Given the transformed dataset, the problem of learning the gene network N (which is a time delayed network) is reduced to the problem of learning a traditional Bayesian network M. As the gene network N is sparse [88] and the transformation makes it more sparse, learning the transformed network M could be very time consuming. This section presents an algorithm which speeds up the learning process.

A Bayesian network is normally represented by $M =< G, \theta >$, where $G = (V, E)$ is a directed acyclic graph and θ is the parameter set of G. An essential point of the Bayesian network is the decomposition of the joint probability of the random variables into conditional probabilities as shown below:

$$P(V_1, ..., V_n) = \prod_i P(V_i|Pa(V_i)) \tag{4.1}$$

where V_i is a variable and $Pa(V_i)$ is the set of parents of V_i in the network [34].

Learning a Bayesian network structure is often an optimization problem. A general approach is to define a score that describes the closeness of a possible structure to the observed data [36] and then find a structure that maximizes the score. I use Bayesian score [21] because of its important property of decomposability in the presence of full data [35], as shown below:

$$Score(G : D) = \sum_i Score(V_i|Pa(V_i)) \tag{4.2}$$

With the property of decomposability, the learning procedure can be decomposed into a number of local search procedures to search the parents for each variable independently. A commonly used technique is to divide the learning procedure into two steps. The first step is to find a small candidate parent set for each variable

based on the dependency between two variables. The second step is to select the parent set $Pa(V_i)$ for every variable V_i that maximizes $Score(V_i|Pa(V_i))$ from the candidate parent set. $Score(V_i|Pa(V_i) : D)$ is the measurement of the fitness that $Pa(V_i)$ is the parent set of V_i given the dataset D. In general, greedy or heuristic algorithms are used to accomplish this step [21, 34].

I propose a learning algorithm, the time delayed Network Learning (TDNL) algorithm, which transforms the time delayed network (as described in Section 4.2) into a traditional network and recovers the time delayed network by learning the transformed network. The learning procedure is based on the framework of the sparse candidate algorithm (SCA) [36]. SCA improves the first step by selecting a set of candidate parent for every variable V_i and enhancing the candidate parent set iteratively based on the network learnt from the previous iteration. I perform further improvements for the first step using the following: (1) Modified mutual information to measure the dependence between two genes. This measurement considers common structures that repeatedly appear over several iterations instead of only considering the structure appearing in a single iteration. (2) Random sampling to ensure that each variable has the chance to be a candidate parent of some gene. (3) A novel learning method, learning by modification, to efficiently select parents from a candidate parent set in the second step.

4.3.1 Choosing candidate parent sets

A general method for generating a small candidate parent set for a variable is to make use of the dependencies between the variable and its possible parents. One commonly used measurement for dependency between variables is mutual information (MI), defined below:

$$MI(X;Y) = \sum_{x,y} \hat{P}(x,y) \log \frac{\hat{P}(x,y)}{\hat{P}(x)\hat{P}(y)} \tag{4.3}$$

where X and Y are some variables, x and y are discrete values of X and Y respectively, and $\hat{P}$ represents the observed frequency of samples in the dataset. If a parent-child relationship exists between X and Y, $MI(X;Y)$ is expected to have a high value. However, because of the noise in the data and the combined effect of parents, some child-parent pairs sometimes have low MI values and may be excluded from the corresponding candidate parent sets.

To solve the problem, Friedman [36] proposed employing Kullback-Leibler(KL) divergency to measure the dependency between a pair of genes given a learned network from data. KL divergency is to measure the different between two probability distributions [36]. In [36], these distributions are observed distribution of the dataset and the distribution of the network learned from the dataset. The KL divergency can be considered as the dependency between a gene pair with respect to a network. A gene with high KL divergency to a target gene is possibly a weak parent of the target gene. It is defined as follows:

$$M_{Disc}(X, Y|M) = \sum_{x,y} \hat{P}(x, y) \log \frac{\hat{P}(x, y)}{P_M(x, y)} \tag{4.4}$$

where X and Y are variables, $M_{Disc}(X, Y|M)$ is the dependence between X and Y with respect to the network M, $\hat{P}$ denotes the observed frequency, P_M is the estimated probability given M, and x and y are the values of X and Y.

However, some true parents of a variable X may have small MI value and KL divergence values. Those parents of X are denoted as the weak parents of X. Such weak parents of variables may be left out in the candidate parent set in the SCA algorithm. In order to solve this problem, I propose the following: In each iteration, the following steps are executed: (1) The parents of X in the previous iteration become the candidate parents of X in the current iteration automatically. (2) The genes with high mutual information with X are selected to be the candidate

parents of X. The number of candidate parents selected in this step is depend on the number of candidate parents selected in step 1, so that the candidate parents selected by this two step is a constant number through the learning procedure. (3) Finally, some extra genes are randomly included into the candidate parent set of X. Steps (1) and (2) select the genes with high dependencies to X to be the candidate parents of X while (3) ensures that, after enough iterations, each weak parent has the chance to be included into the candidate parent set.

In every iteration, the candidate parent set contains at most one transformed variable for a gene since the time delayed network has no multi-edges existing between two genes. This property reduces learning complexity.

As described above, the selection of candidate parents is done partially on a random basis, leading to an unstable structure. That in turn might result in bias in the selection of parents in the next iteration, and thus, giving a biased network as output. In addition, this problem might make the learning process fail to converge. I propose a new criterion in place of KL divergence. This replacement reduces the degree of the random sampling and stabilizes the score so that the score improves smoothly.

The new criterion $\tau(X;Y)_t$ for variables X and Y can be computed as follows:

$$\tau(X;Y)_1 = MI(X;Y) \tag{4.5}$$

$$\tau(X;Y)_t = \alpha \times \tau(X;Y)_{t-1} + (1-\alpha) \times M_{Disc}(X;Y|M_{t-1}) \tag{4.6}$$

where t indicates the t-th iteration for $t > 1$, M_{t-1} represents the network learnt from the $(t-1)$-th iteration, and α is a parameter to adjust the accumulation rate where $0 \leq \alpha \leq 1$. Note that $\tau(X;Y)_t = MI(X;Y)$ if $\alpha = 1$, and $\tau(X;Y)_t = M_{Disc}(X;Y|M_{t-1})$ if $\alpha = 0$. Generally, I set $\alpha = 0.5$, which indicates that the sub-structures appearing repeatedly in previous iterations are of similar weights to

the sub-structures in the current iteration. An experiment in Section 4.4 illustrates the effect of the value of α to the learning procedure. The formulation of $\tau(X;Y)_t$ includes $\tau(X;Y)_{t-1}$ and thus recursively includes ($\tau(X;Y)_{t-2}$,...,$\tau(X;Y)_1$), each of which is of a different proportion. That is, τ_t is based on all learnt networks, from M_1 to M_{t-1}, and each network contributes a part to the final τ_t. The bias and the non-deterministic nature of random sampling in one network are complemented by other good networks, which avoid bias in the final network. It can be observed that correct structures appear repeatedly, and thus, they have greater chances of remaining in the current network. This reduces the effect of rapid change in KL divergence, resulting in improved convergence. Note that KL divergence is based only on the single previous network. Any bias in the previous network might affect the current network and even subsequent networks. It is easy to see τ_t is decomposable as both MI and M_{Disc} are decomposable.

Another significant problem in learning is that it is time consuming to make inference on $P_{M_{t-1}}(X,Y)$ in M_{t-1} for calculating $\tau(X;Y)_t$. Ong et al. [76] estimated that it would take nine months to do inference on a network with about 200 genes by the junction tree algorithm directly. I solve the problem by approximating it as following:

- If $Y \in Pa(X)$ in M_{t-1}, $P_{M_{t-1}}(X,Y) = P_{M_{t-1}}(X|Y)P_{M_{t-1}}(Y)$. Considering the time delay, $P_{M_{t-1}}(X,Y) = P_{M_{t-1}}(X|Y,\delta)P_{M_{t-1}}(Y)$. Similarly, $X \in Pa(Y)$ is calculated.
- Otherwise, X and Y are conditionally independent. Thus, I approximate it as $P_{M_{t-1}}(X,Y) \approx P_{M_{t-1}}(X)P_{M_{t-1}}(Y)$.

$P_{M_{t-1}}(X)$, $P_{M_{t-1}}(Y)$, $P_{M_{t-1}}(X|Y,\delta)$ and $P_{M_{t-1}}(Y|X,\delta)$ are already learnt in the previous iteration where $P_{M_{t-1}}$ denotes the probability distribution in network M_{t-1}. Since the approximation is quite fast, the whole algorithm can be applied to large datasets.

4.3.2 Structure learning

A transformed network can be learnt with any classical learning algorithm. I propose an algorithm, Learning by Modification (LBM), which is motivated by the K2 algorithm [21]. Due to the decomposability of Bayesian score, the best parents for each variable can be found independently. For every variable V_i, and its candidate parent set, K2 includes new variables as its parents one by one as long as the inclusion of the new variable can improve the score of the network maximally as compared to other candidates.

One problem of K2 is that it does not consider the combined effect of the parents to a particular variable in the early iterations. Sometimes, this problem leads to a low score network. In addition, there is no way to remove a selected false parent from the parent set. This effect may be seen in the example in Figure 4.3.

$K2$	learning by elimination	learning by modification
{A}	{A, C, D, E}	{A}
↓	↓	↓
{A, B}	{A, C, D}	{A, B}
	↓	↓
	{A, C}	{A, B, C}
		↓
		{B, C}

Figure 4.3: An example: Given a variable X and its candidate parent set $CPS = \{A, B, C, D, E\}$, in which B, C are the parents of X. Suppose the following subsets of CPS give the scores to X in descending order: $\{B, C\} > \{A, B\} > \{A, C\} > \{A, B, C\} > \{A, C, D\} > \{A, C, D, E\} > \{A\} > \{B\}$, and all other subsets give scores smaller than B. K2 selects {A} and {A, B} in order but misses $\{B, C\}$ since K2 fails to capture the combined effect of the two parents. Learning by elimination selects {A, C, D, E}, {A, C, D} and {A, C} in order while learning by modification selects {A}, {A, B}, {A, B, C} and {B, C} in order.

An alternative way, learning by elimination, involves initially setting the parent set as the candidate parent set and then iteratively deleting a variable from the parent set which maximizes the score. The process stops when further removal of any variable from the parent set would not increase the score. The algorithm takes into account the combined effect starting the beginning and removes variables that are not parents one by one. Still, false parents in the parent set may introduce noise which reduces the combined effect. If there are some false parent variables, a problem similar to that in k2 arises: the true parent might be removed at the beginning with no chance to return. See the second column of Figure 4.3 for an example.

In Figure 4.4, I propose a method called Learning by Modification (LBM) to overcome these problems. LBM is a heuristic algorithm for finding a parent set (PS) of V_i by maximizing $Score(V_i|PS)$. It is iterative in nature. In every iteration, it includes exactly one new variable into PS and then deletes zero or more variables. The two merits of LBM are as follows:

First, LBM ensures PS contains as few false parents as possible. As fewer parents would mean the combined effect of the true parents is less diluted, the true parents V_j of V_i would have a better chance of getting a higher $Score(V_i|PS \cup \{V_j\})$, and hence, they would get a better chance to be included into PS. In other words, it avoids the problem of backward selection where the parent set contains too many false parents.

Second, unlike K2, Step (a) of every iteration of LBM always includes a variable V_j into PS irrespective of whether $Score(V_i|PS \cup \{V_j\})$ is greater than $Score(V_i|PS)$ or not. This idea avoids the problem of K2 where the noise of the false parents may prevent the true parent of V_i being included into PS. Figure 4.3 illustrates the algorithm with a simple example.

As I assume no loop exists in the same time slice, I do not detect loops formed solely by edges with time delay 0. The algorithm is fast and accurate.

- Let CPS be the candidate parent set of V_i and $PS = \{\}$ be the parent set.
- Compute Bayesian score $S = Score(V_i|PS)$
- Repeat the following procedure until PS does not change:
  - Set $S = Score(V_i|PS \cup \{V_j\})$ and $PS = PS \cup \{V_j\}$ where $X \in CPS$ maximizes $Score(V_i|PS \cup \{V_j\})$.
  - Repeat the following procedure until PS does not change: Select a variable $V_j \in PS$ which maximizes $Score(V_i|PS - \{V_j\})$. If $Score(V_i|PS - \{V_j\}) > S$, then set $S = Score(V_i|PS - \{V_j\})$ and $PS = PS - \{V_j\}$.

Figure 4.4: Outline of the Learn by Modification algorithm.

4.4 Experimental results and comparison

I implement the TDNL learning algorithm using Bayes network toolbox (BNT)[4] in MATLAB.

This section presents the experiments on both artificial and real-life datasets to evaluate the effectiveness of TDNL learning algorithm. I also compare the learning performance of TDNL with both classical and recent learning algorithms: (1) K2 [21] is a hill-climbing algorithm which iteratively incorporates variables into current parent set one by one until the Bayesian score cannot be further improved; (2) REVEAL [67] is a dynamic Bayesian network (DBN) learning algorithm which enumerates all possible parent combinations with smaller size than a predefined threshold. (3) DBmcmc [47] is a MCMC (Markov chain Monte Carlo) learning approach. (4) GeneNetwork [104] learns gene network by reverse engineering inference approaches.

4.4.1 Structure learning for artificial datasets

This section illustrates the effectiveness of the TDNL algorithm on both synthetic and real data. As described in Section 3.1.4, a discrete model is chosen and each variable takes on Boolean values in the experiment. The experiments are conducted on four sets of artificial datasets whose underlining structures and parameters are randomly generated. Each set has 10 datasets with same number of genes. Each dataset contains 10, 50, 100 or 200 genes, and consists of 400 time slices. The specification of the each set of datasets is averaged into D_1, D_2, D_3 or D_4, as shown in Table 4.1[5].

[4] http://www.ai.mit.edu/∼murphyk

[5] The number of edge for each set is the average number of edges of 10 datasets in this set. The numbers of edges are rounded to integers.

Dataset	No. of genes	No. of edges	Max. No. of parents
1	10	17	4
2	50	135	4
3	100	189	4
4	200	377	4

Table 4.1: Specification of the synthetic datasets.

Dataset	K2	REVEAL	GeneNetwork	DBmcmc	TDNL
	C/T	C/T	C/T	C/T	C/T
1	5/9	4/11	9/39	5/17	10/18
2	12/28	N/A	19/104	16/93	47/98
3	41/127	N/A	21/182	23/191	82/169
4	97/286	N/A	18/296	28/216	146/342

Table 4.2: Performance comparison of the K2, REVEAL, GeneNetwork, DBmcmc and TDNL learning methods. N/A means the result could not be found out as the algorithm has not finished execution within a reasonable time (2 days). C indicates the number of true positive edges, T indicates the number of total learned edges.

True positive edges and the total learned edges (true positive + false positive) are used to measure the accuracy of the four approaches where C indicates the number of true positive edges and T indicates the number of total learned edges. Table 4.2 shows the results. As shown, TDNL got best performance in the sense of high true positive rate and low false positive rate. Also, when the size the datasets increases, the performance of TDNL can still be quite good.

I also conduct an experiment to show how α in τ calculation affects the learning procedure. Figure 4.5 illustrates the different learning variations given different α in a artificial dataset with 10 genes and 18 edges. It is obvious when α is close to either 0 or 1, the sensitivities (true positive / total learned edges) vary rapidly from one iteration to another while when α is between 0 and 1, the curves converge at the latter parts. This is because when α is close to 0 or 1, τ is close to MI or KL divergency. Random sampling brings variation to the learning procedure when the learning is dependent on the measurements from one iteration while the measurements from all iterations can stabilize the learning procedure.

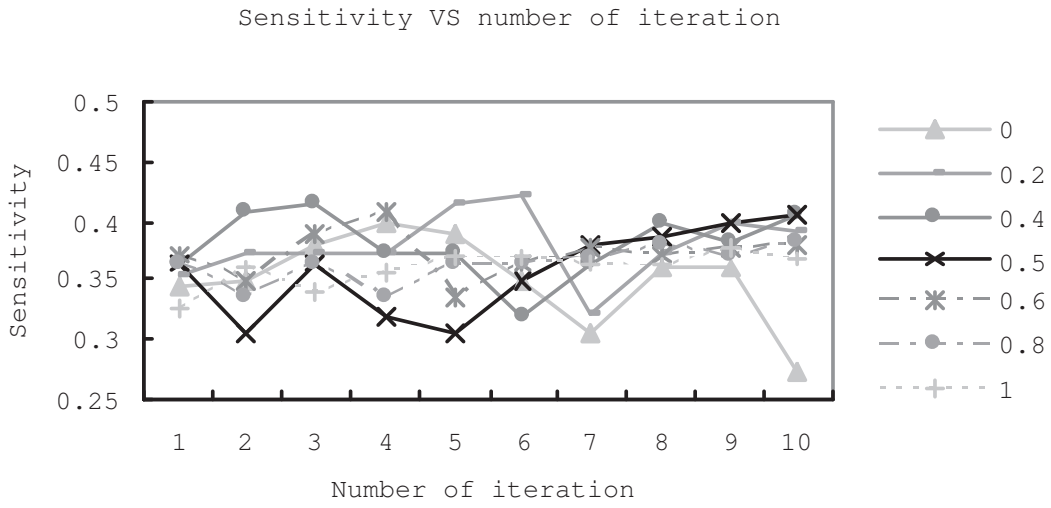

Figure 4.5: Convergence curves of sensitivity and number of iteration.

4.4.2 Structure learning on yeast subnetwork

To demonstrate the learning power of TDNL in real-life gene expression data, I conduct experiments on two yeast subnetworks [37] in *S. Cerevisiae*.

The first one, denoted as N_{13}, is a well-studied yeast cell cycle transcriptional network backbone[37]. The backbone comprises of 13 important regulators which control the main events of the cell cycle: SWI4, SWI6, MBP1, MCM1, NDD1, SWI5, ACE2, CDC28, CLN3, CLB2, SIC1, FKH1 and CLN2[6]. The network structure is shown in Figure 4.6(a) where each dotted circle indicates a transcriptional factor and each directed line indicates a regulatory event between two transcriptional factors: $CLN3-CDC28$ kinase activate transcription factors, SBF (comprised of $SWI4$ and $SWI6$) and MBF (comprised of $SWI6$ and $MBP1$), to begin the cell cycle. SBF and MBF then activate about 200 genes in late G1 and S phases including $NDD1$ and $CLN2$. $CLN2$ is one of the important factors that activate $CLB2-CDC28$ kinase. The complex $CLB2-CDC28$ inactivates SBF and MBP to shut off G1/S events and the cell goes to G2 phase. $CLB2-CDC28$ continues to activate a transcription factor containing $MCM1$, $FKH1$ and $NDD1$. This factor activates a set of genes including $SWI5$ and $SCE2$ which activate $SIC1$. $SIC1$ and some other genes inhibit $CLB2-CDC28$. Details of the genes and cell cycle regulatory network may be found in [85]. Though the subnetwork is small, it is a good example to demonstrate learning power since it is essential to yeast development and differentiation [37]. The subnetwork controls cell cycle, regulates global gene expression and diversifies stage-specific functions to produce a continuous cycle of cellular events [37]. The absence of any regulatory event will stop the cell cycle. The subnetwork contains some loops which stabilize and stimulate the development of the cell cycle.

The second one, denoted as N_{105}, is a yeast cell cycle regulatory network for 9 cell cycle regulators and 96 cell cycle regulated genes. The regulatory relationships

[6]FKH2 is not included since it has too many missing expression values.

between the 9 regulators and all 105 genes are partially discovered by biological literatures [18, 109] [7].

[7]The original dataset contains 116 genes[109]. Some redundant genes are removed by the authors.

	K2	REVEAL	DBmcmc	GeneNetwork	TDNL
	C/T	C/T	C/T	C/T	C/T
Y_{s13}	2/12	5/17	12/33	8/31	14/29
Y_{s105}	14/59	N/A	22/69	33/146	39/84
Y_{c13}	6/20	4/16	10/31	7/21	15/33
Y_{c105}	29/84	N/A	16/72	10/66	33/81

Table 4.3: Comparison of learning performance. T indicates the number of total learnt edges and C indicates correct predicted edges. In row of Y_{s105} and Y_{c105}, the total learnt edges are the total edges between regulators and target genes. N/A means the result could not be found out as the algorithm has not finished execution within a reasonable time (2 days).

For the experiment, I chose as data sources two microarray gene expression datasets of the *S. Cerevisiae* genome: (1) is denoted as D_s, published in [90]. The microarray expression dataset contains 76 replicates time series gene expression measurements from the synchronized cells. I select the CDC_15 containing 25 time slices with a time interval of 10 minutes between successive time slices. More details of the dataset may be found in [90]. (2) is denoted as D_c, published in [18], contains 16 time slices and the time interval is 10 minutes. Therefore, we get four datasets, Y_{s13}, Y_{s105}, Y_{c13} and Y_{c105}, where 13 and 105 indicate the network structure is from N_{13} or N_{105} while s and c indicate the source of the microarray dataset is from D_s or D_c. The continuous expression levels are discretized into 2 discrete levels, with 0 and 1 indicating low expression and high expression respectively. An expression level is discretized to 1 if it is greater than 0, and 0 otherwise. Since the time for a single cell cycle of *S. Cerevisiae* is about 1.5 to 2 hours, it is reasonable to set the maximum time delay to be four time slices, i.e., 40 minutes.

I follow the verification method used by Kim et al. [57]: a learnt edge from gene X to gene Y is considered correct if there is an edge from the transcriptional factor containing X to a transcriptional factor containing Y in Figure 4.6(a), or they are in the same transcriptional factor in Figure 4.6(a)[8]. Table 5.3 summarizes the result of comparison. The learning performance of TDNL is significantly better than that of other methods. Moreover, TDNL can find loops in networks. I illustrate an example, which is the learning result of Y_{s13} by TDNL in Figure 4.6(b): there is a loop: $CDC28 \rightarrow MCM1 \rightarrow SWI5 \rightarrow SIC1 \rightarrow CDC28$. In addition, most time delays of learnt edges by TDNL are also reasonable. The regulations happening in the same or adjacent phases of a cell cycle have shorter time delays (for example, MCM1 → CLB2 and SWI5→SIC1 have time delays of 1 and 2, respectively) while the regulations spanning more than one phase have longer time delays (for example, SIC1→CDC28 has a time delay of 4).

[8]This is easy to understand since the proteins in a transcriptional factor or a complex interact and activate each other to function properly.

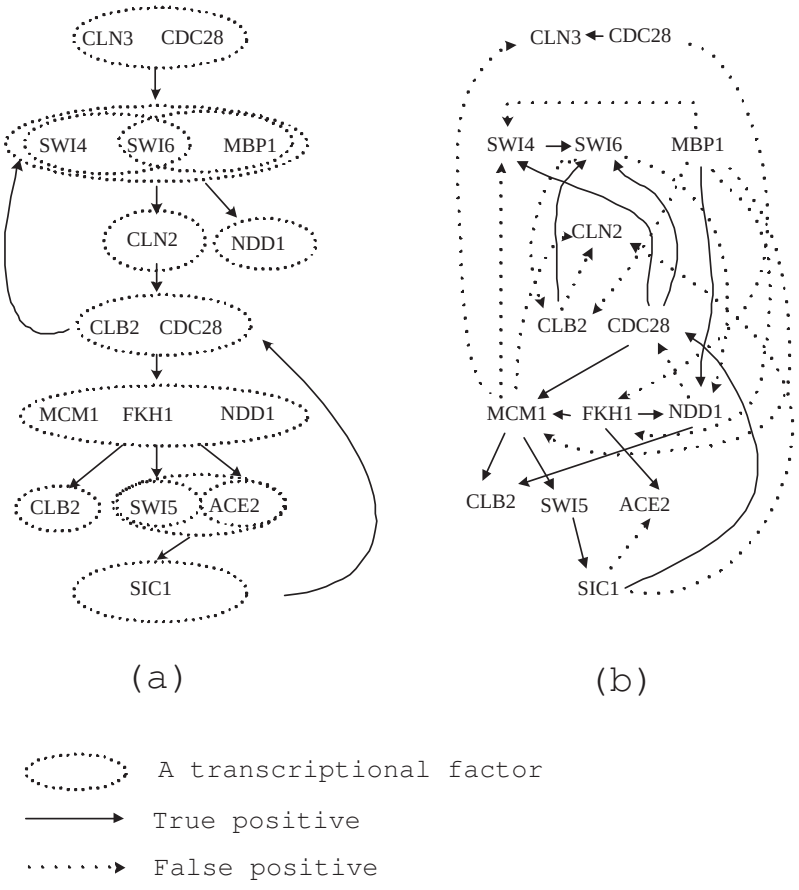

Figure 4.6: Learning performance of TDNL on a real gene subnetwork. (a) The yeast cell cycle transcriptional regulatory subnetwork. (b) The network structure learnt by TDNL, which contains 29 edges of which 14 are correct.

4.4.3 Markov relation and confidence analysis

In order to study the strength of the relationship between two genes in the networks learnt by the learning algorithm, a widely used statistical feature, Markov relation [33, 34], is analyzed. Markov relation indicates whether a gene Y is in the Markov blanket of another gene X (denoted by $Y \in MB(X)$). The Markov blanket of X is the minimum set of genes that shield X from the rest of the genes in the network. More precisely, $Y \in MB(X)$ if and only if there is either an edge between them or both are parents of another variable. A Markov relation indicates that the two genes are related in some joint biological interaction [34]. The statistical confidence which measures the likelihood of the feature is estimated by the bootstrap method [34] described below:

- For i = 1... m :
  - Generate a "perturbed" version of the input transformed dataset by re-sampling ρ instances where ρ is smaller than the number of instances of the transformed dataset.
  - Apply the learning algorithm on the perturbed dataset and induce a network N_i.
- For each Markov relation f, the confidence is calculated as follows:

$$conf(f) = 1/m \sum_{i=1}^{m} f(N_i) \tag{4.7}$$

  where $f(N_i)$ is 1 if f is a feature in N_i and 0 otherwise.

I initialize ρ as 35 and m as 100 in the experiment. A high confidence value of a feature for two genes indicates that the learning algorithm can consistently recover the relationship between them.

The high-ranking Markov relations, learned from Y_{s13}, with confidence ≥ 0.8 are listed in Figure 4.7 for the TDNL algorithm. Of the 26 relationships, 17 can

Markov relations	Confidence	Comments
SWI5, SIC1	1.0	*
CLN2, SIC1	1.0	
CLN3, CLB2	1.0	
CDC28, NDD1	0.97	*
MCM1, NDD1	0.97	*
NDD1, SWI6	0.96	
ACE2, SWI5	0.95	*
CLB2, FKH1	0.94	*
ACE2, SWI4	0.92	
CLN2, CLB2	0.92	*
FKH1, MBP1	0.89	
MCM1, SWI6	0.89	
CDC28, SIC1	0.88	*
SWI4, SWI6	0.87	*
CLN3, MCM1	0.81	
CLN3, CDC28	0.85	*
MBP1, SWI5	0.85	
ACE2, SIC1	0.85	*
CLB2, MCM1	0.84	*
FKH1, NDD1	0.83	*
ACE2, FKH1	0.83	*
CDC28, MCM1	0.82	*
CLB2, NDD1	0.82	*
MCM1, SWI5	0.81	*
FKH1, SWI5	0.8	*
CLN2, MCM1	0.8	

Figure 4.7: Top Markov relations list. * indicates the relation is verified by Figure 4.6(a).

be verified. Note that the transcription factors involved in the cell cycle regulation events [37], SBF (SWI4 and SWI6), MBF (SWI6 and MBP1) and SFF (containing NDD1, FKH1, etc.) are discovered in Figure 4.7. This is due to the fact that the genes in a transcription regulating complex co-regulate the target genes and they have strong Markov relations. As shown in Figure 4.7, most Markov relations can be verified by the biological literature.

4.5 Conclusion

In this chapter, the traditional Bayesian network is enhanced with a time-delay model in order to represent various time delays gene networks. The increase in the number of parameters to be learnt is accounted for by the proposed efficient learning algorithm. The algorithm produces better performance over both the artificial and yeast gene expression datasets than other algorithms do. The algorithm could also detect directed loops spanning a long time period in the yeast gene expression dataset. However, as the learning process is iterative, it is time consuming when the dataset is big. For example, when learning a dataset with 10 gene, it needs about 10 iterations and takes less than one minute. However, when learning a datasets with 50 genes, it needs more than 20 iterations and takes more than 10 minutes. Also, the power of random sampling would decrease when a large scale network is being learnt since the sampling space is enlarged.

CHAPTER 5

Learning gene networks by conditional dependence

5.1 Introduction

Probabilistic methods such as Bayesian networks have been proposed for used in learning gene networks, and they have yielded some reasonable results [33, 34]. However, biologically significant results could only be obtained when the gene networks are learnt from some selected small subsets of genes [29, 68].

One main reason for the ineffectiveness is due to the well-known fact that microarray actually measures mRNA expression levels instead of protein expression levels. Hence, microarray can only provide partial information of a gene network. Another reason is weak parents. In a real biological gene network, some parent genes have little pair-wise correlation with their target genes [29]. These parents are called weak parents. In this chapter, I examine whether more information can be extracted from microarray data to facilitate gene network learning. I propose enhancing the learning by integrating the following three types of hidden information:

1. Recent works [29, 34, 68] have studied the regulatory relationship among genes based on the pairwise correlation of their expression profiles. We may be able to extract other types of relationships from the expression data. In this chapter, I propose using the conditional dependence to extract more regulation relationships. Basically, if two genes g_1 and g_2 regulate the same target gene g, due to the combined effect of g_1 and g_2, we should expect g_1 and g_2 to be dependent given the gene g. This idea allows us to find regulation pairs without strong pair-wise correlation.

2. During the regulation process, the protein products of a set of genes form complexes before they interact with the target gene [73]. The loss of any element in a complex may destroy the function of the whole complex. Hence, we should consider the factor of the complex when we select parents. However, few existing methods consider this important factor. This chapter propose finding complexes by conditional dependence. Then, the information of the predicted complexes can be used to predict the parents of each gene.

3. The time delays of gene regulations are different for different genes [29, 109]. Most existing models assume either there is no time delay or a constant time delay for gene regulations in the gene network, which is not true in a real biological regulatory system. I propose integrating variable time delay into conditional and unconditional dependence measurements to improve the accuracy of the gene network prediction.

The rest of the chapter is organized as follows: a two-step algorithm is described in Section 5.2. The experimental results on gene expression datasets are presented in Section 5.3. The conclusion from the current work as well as its implications are given in Section 5.4.

5.2 Conditional dependence learning algorithm

This chapter proposes a two-step learning method known as the Conditional Dependence (CD) learning algorithm to learn gene networks. A brief overview of the algorithm is as follows: In the first step, parent-child correlation is used together with the conditional dependence among parents to select a small subset of genes as candidate parents. In this step, collaborations among parents are used to find weak parents. Finding variable time delay relationships is also integrated in this step. In the second step, the subunits among candidate parents and explored parents are searched based on both subunits and single genes. Based on the mechanism of a regulatory complex [73] and the theory of conditional dependence, the elements in a subunit should have strong conditional dependence with each other. This property helps in the finding of subunits. The two steps are discussed in detail in the following subsections.

5.2.1 Candidate parent selection

As stated in Section 4.3.1, the correlation between two genes is generally used to select candidate parents of a target gene. If the increase (decrease) of the expression level of a gene always comes with the increase or decrease of the expression level of another gene, we say they are highly correlated. If two genes g and g' have a low correlation, it is assumed that g cannot be a parent of g', and vice versa. Several measurements have been suggested to model the correlation such as Pearson correlation [29], Mutual Information (MI) [34], KL divergence [34], and random sampling based mutual information [68]. These pair-wise correlation may generate weak parents [36, 68]. The following is an example:

- **Example:** In yeast, genes NDD1, MCM1 and FKH1 form a complex and regulate SWI5. As shown in Figure 5.1, only FKH1 has a similar expression profile as SWI5 while NDD1 and MCM1's expression profiles do not correlate

with SWI5. Thus, NDD1 and MCM1 are the weak parents of SWI5.

Since all the aforementioned measurements are based on pairwise correlation, they are prone to miss out a lot of weak parents. To extract weak parents, I propose to studying correlations between parents in addition to pairwise correlation.

Consider two genes X and Y. Suppose both of them are not parents or ancestors of a target gene Z, then the correlation of X and Y is expected to be similar with or without knowledge of Z. On the other hand, if both of them are parents (or ancestors) of Z, X and Y are expected to be more correlated given Z while less correlated without the knowledge of Z. An example is shown below. In the example, the relationship between two genes X and Y is indicated by the dot plot where each dot in the figure is a pair of expression values of X and Y at some particular time.

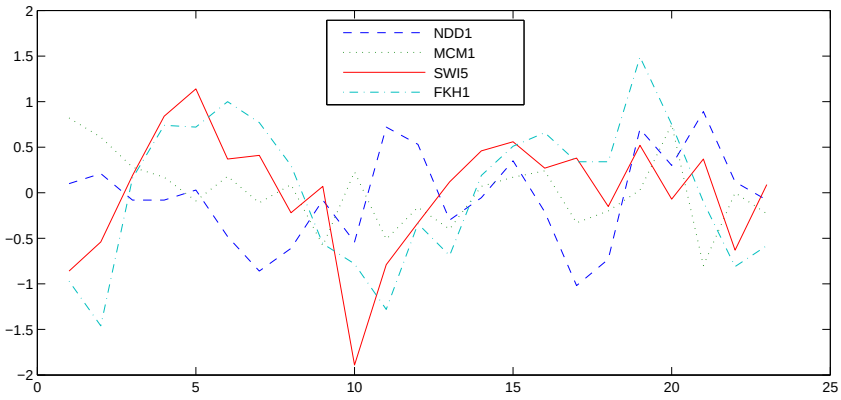

Figure 5.1: Gene expression profile comparison of NDD1, MCM1, FKH1 and SWI5.

- **Example:** Consider the genes NDD1 and FKH1 between which there is no regulatory relationship. The dot plot of the two genes is shown in Figure 5.2(a). The dots are arbitrarily distributed without any trend. I test the changes of correlations between NDD1 and FKH1, given one of the following three genes: (1) A randomly generated gene g_r whose expression value is randomly generated; thus there should be no regulatory relationship between g_r and NDD1/FKH1. (2) Gene UNG1 which is not regulated by NDD1 and FKH1. (3) Gene SWI5 which is regulated by NDD1 and FKH1. As shown in Figures 5.2(b) and (c), the correlation between NDD1 and FKH1 does not change much when given $g_r \geq 0$, $g_r < 0$, $UNG1 \geq 0$ or $UNG1 < 0$. However, when given either $SWI5 \geq 0$ or $SWI5 < 0$, the correlation between NDD1 and FKH1, as shown in Figure 5.2(d), is different from that shown in Figure 5.2(a). Moreover, the points are grouped into two clusters where points of $(NDD1, FKH1|SWI5 \geq 0)$ are mainly in one cluster which lies in the area of $(NDD1 < 0, FKH1 > 0)$, and the points of $(NDD1 : FKH1|SWI5 < 0)$ are mainly in another cluster which lies in the area of $(NDD1 > 0, FKH1 < 0)$. Both clusters trend from bottom-left to top-right.

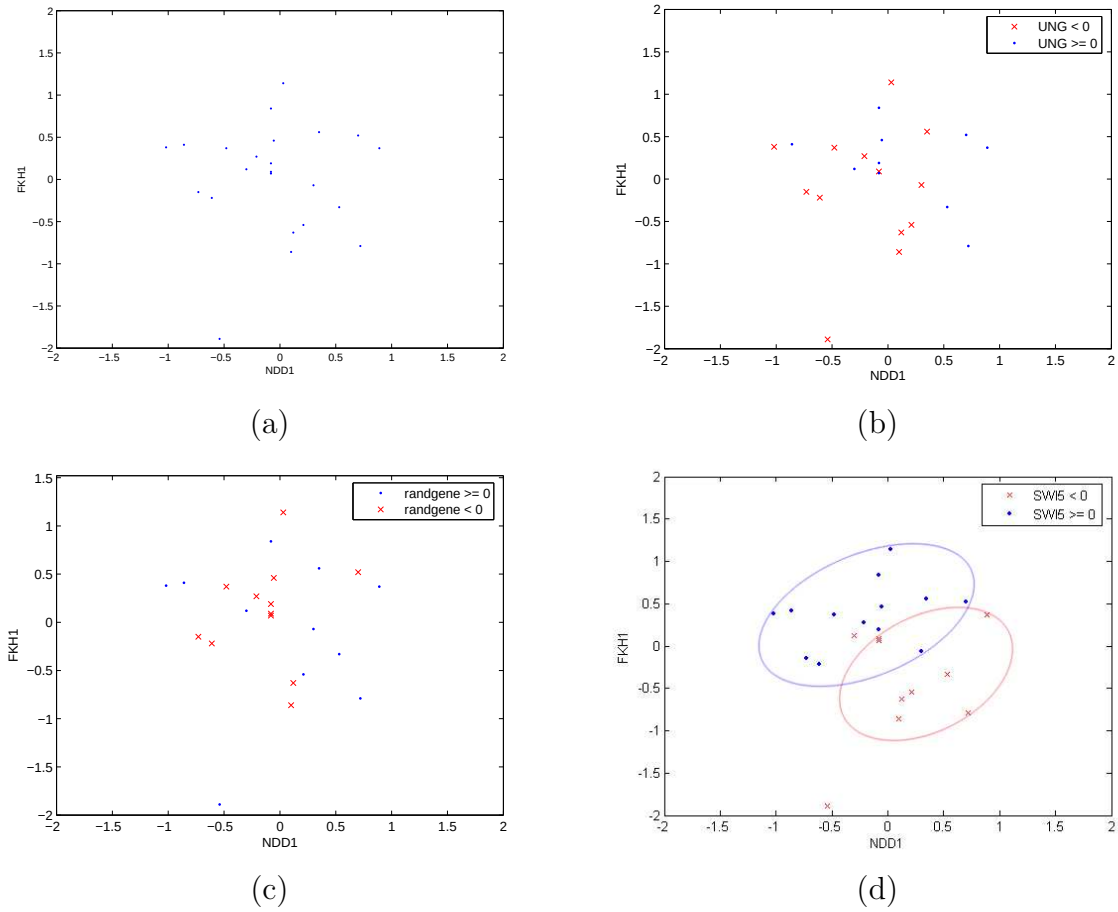

Figure 5.2: (a) shows the dot plot of the expression levels of NDD1 and FKH1. Each point represents the expression levels of NDD1 and FKH1 at some particular time slot. It is clear that there is no correlation between NDD1 and FKH1. (b) and (c) show the dot plot of NDD1 and FKH1 given the gene UNG or a randomly generated gene. A point is labeled by '$\times$' or '$\cdot$', depending on whether the expression level of UNG (or the random gene) is negative or not in the particular time slot. '$\times$' and '$\cdot$' are randomly distributed. (d) shows the dot plot of NDD1 and FKH1 given the gene SWI5. The plot shows that '$\times$' and '$\cdot$' are in different distributions and can be clustered into two groups. This proves that FKH1 and NDD1 are co-parents of SWI5.

I try to capture correlation changes by Conditional Relative Entropy (CRE). Consider three genes: $X, Y and Z$. I denote the conditional relative entropy between X and Y given Z as $CRE(X, Y|Z)$, and define it as follows:

$$\sum_z P(z) D(P(X, Y|z)||P(X, Y)) \tag{5.1}$$

$$= \sum_z P(z) \sum_{x,y} P(x, y|z) \log \frac{P(x,y|z)}{P(x,y)} \tag{5.2}$$

$$= \sum_{x,y,z} P(x, y, z) \log \frac{P(x,y,z)}{P(x,y)P(z)} \tag{5.3}$$

where $D(p||q)$ is the relative entropy which measures the distance between the probability distributions p and q. $P(X, Y|Z)$ and $P(X, Y)$ are the probability distributions of the gene pair (X, Y) with and without, respectively, the information of Z.

Note that if $CRE(X, Y|Z)$ is non-zero, the probability distribution of $P(X, Y)$ is different from that of $P(X, Y|Z)$. X and Y are possible co-parents of Z. Based on this idea, even when both X and Y are weak parents of Z, conditional relative entropy provides a mechanism for us to extract them.

There is another type of weak parents proposed by Friedman [36]. Suppose $Pa(A) = \{B, C\}$ and $Pa(C) = \{D\}$, it is possible that $MI(A : C) > MI(A : D) > MI(A : B)$. In this case, B is weak. Based on the conditional dependence, the presence of A does not change the dependence between C and D in any significant manner while the dependence between B and C changes significantly. Hence, CRE is able to deduce that B is a candidate parent/ancestor of A.

In summary, for a target gene Z, a gene X is included in the candidate parent set of Z, noted by $CPS(Z)$, if X has a strong correlation with the target gene Z or if X has a strong conditional dependence with some other gene Y given the presence of the target gene Z (Y is preferred as a gene which has high correlation with Z). The details of the candidate selection procedure are shown in Figure 5.3.

For every gene Z, compute the candidate parent set $CPS(Z)$ of Z as follows:

1. Calculate the MI between the target gene and other genes. Then the genes with high MI scores are included in $CPS(Z)$.

2. For each variable $X \in CPS(Z)$, find the variables Ys such that the $CRE(X, Y|Z)$ is high. These variables Ys are possible weak parents, and thus, they are also included in $CPS(Z)$.

3. Find all variable pairs (X, Y) such that $CRE(X, Y|Z)$ is high. Include X, Y in $CPS(Z)$. This step selects weak parents which have strong conditional dependencies with each other but small MI scores with Z. (This step can be used solely to find the candidate parent set.)

Figure 5.3: Candidate parent selection procedure.

5.2.2 Learning structure from candidate parent sets

For every gene Z, given the candidate parent set $CPS(Z)$, the next step is to learn the actual parents of Z. Basically, we need to find a subset of $CPS(Z)$ which optimizes the Bayesian score. This problem is NP-hard [34]. Currently, heuristic methods such as hill climbing and learning by modification (LBM) are used for the task [34, 68]. All the methods implicitly assume that every gene affects Z independently.

In a biological system, the parent gene first encodes a regulatory protein and the regulatory protein performs gene regulation. There are several ways for regulatory proteins to regulate the target gene Z: (1) Some proteins have specific binding sites at the upstream of the target genes. (2) Some proteins are combined to form a regulatory factor or a complex, and then bind to a specific site and regulate the target gene. (3) Combination of (1) and (2).

In other words, some proteins have to combine as a unit to regulate the target gene Z. These proteins individually are not strongly correlated with Z. However, if they are considered as a unit, the correlation between them and the gene Z is significantly larger [73]. Such a unit is biologically termed a complex of proteins. Loss of any member in a complex often leads to a dramatic decrease in functionality. Therefore, each member in a complex is indispensable to anyone else. A good example of intensive collaboration is basal transcription factors [52]. For a basal transcription factor complex, the presence of all members is necessary to reconstitute accurate transcription. The issue of the complex is not accounted for in traditional methods.

Since proteins are hidden in microarray data, I propose representing the effect of a complex of proteins by the set of corresponding genes, defined as a complex of genes. Then, the parents of X can be learnt as follows: First, among all parents in $CPS(Z)$, the complexes of genes are searched. Then, the parents are found based on both complexes (individual genes are considered singleton complexes).

The algorithm is shown in Figure 5.4.

As mentioned above, a gene in a complex is dependent on other genes in the complex to function. Therefore, we can assume each member of a complex is possibly strongly conditionally dependent on most members in the complex. Based on this knowledge, I propose modeling a complex by a conditional dependence based maximal semi-clique, which is defined below:

For a target gene Z, let $G =< V, E >$ be a graph where $V = CPS(Z)$ and $(X, Y) \in E$ if $CRE(X, Y|Z)$ is bigger than a certain threshold. The weight of (X, Y) is $CRE(X, Y|Z)$. For any $V' \subseteq V$, let G' be the subgraph of G induced by V'. G' is called a semi-clique if every vertex in V' is linked to more than half the nodes in V'[26]. G' is a maximum semi-clique if including any more vertex violates the semi-clique. To find the maximal semi-cliques in $CPS(Z)$, I propose a heuristic algorithm based on the method proposed by Elidan et al. [26].

As observed by Elidan et al.[26], any semi-clique of size 4 or above contains a semi-clique of size 3. I prove the observation as follows:

- **Proof:** $G' =< V', E' >$ is a semi-clique. Suppose G' does not contain any 3-clique. Select a vertex $X \in V'$ randomly and divide V into two parts: $X_i \in V_1$ if there is an edge between X_i and X, otherwise $X_i \in V_2$. It is easy to check that $|V_1| > \frac{1}{2}|V|$ and $|V_2| < \frac{1}{2}|V|$ since X connects to more than half the variables in V. Now consider a variable $X' \in V_1$. X' cannot connect to any variable in V_1; otherwise, there will be a 3-clique[1]. However, even if X' connects to all variables in V_2, X' connects to less than half the variables in V, and G is not a semi-clique. Thus, it is proven.

I find all 3-cliques and then expand each semi-clique into a maximal semi-clique by a greedy algorithm. The detail of the semi-clique finding algorithm is shown

[1]Proof: Suppose X' connects to Y' where $Y' \in V_1$. Since each vertex in V_1 connects to X, there is a 3-clique (X, X', Y').

in Figure 5.5. In addition, all singleton genes and strongly conditional dependent gene pairs in $CPS(Z)$ are also considered as complexes.

For each gene Z,

1. Find all possible complexes in $CPS(Z)$ (including all singleton genes and gene pairs with strong conditional dependence).
2. Let $PS = \{\}$ where PS will be the set of complexes: $PS = \{s_1, s_2, ..., s_k\}$.
3. Select a complex s_i where $Score(\bigcup_{(PS \cup s_i)}) > Score(\bigcup_{PS})$ and maximum $Score(\bigcup_{(PS \cup s_i)})$, PS = {PS, s_i}. ($\bigcup_{PS}$ is the union of PS. Bayesian score is used in the implementation.)
4. Delete s_j from PS where $Score(\bigcup_{(PS - s_j)}) > Score(\bigcup_{PS})$ and maximum $Score(\bigcup_{(PS - s_j)})$. Repeat this step until removing any s_j from PS cannot increase the score.
5. Repeat Steps 3 and 4 until $Score(\bigcup_{PS})$ cannot further improve.
6. Report $\bigcup_{PS}$ as the set of parents of Z

Figure 5.4: Parent selection procedure

For each gene Z, find the maximal semi-cliques among its candidate parents CPS(Z).

1. Let $G' =< V', E' >$ where $V' = CPS(Z)$ and for every $X, Y \in V'$, $(X, Y) \in E'$ if there is a link between X and Y.
2. Find all 3-cliques by brute force.
3. For each 3-clique,
   - I Include one gene which can add a maximum number of links to the semi-clique.
   - II If there are more than one gene meeting the criterion, choose the one which can add maximum weight to the semi-clique.
   - III Repeat I and II until no gene can be added.

Figure 5.5: Semi-clique selection procedure.

5.2.3 Variable time delay

It is well known that gene regulation does not happen instantly and there is some time delay during the regulation process. Also, the time delay in the regulation of different gene pairs can be different [17, 68]. Therefore, without considering various time delays, it is difficult to capture a gene network correctly. In Chapter 4, I proposed learning a gene network with various time delays by transforming it into a network with no time delay. Though the transformation can solve the various time delay problem, the trade-off is that a network with more variables has to been learnt, and thus, the learning time and space requirement both increase.

Instead of performing a transformation, I now propose integrating the computation of time delays between genes with the computation of MI and CRE, as shown below:

$$MI(X : Z, \delta_x) = \sum_{\delta_x} \sum_{i} P(x_i, z_{i+\delta_x}) \log \frac{P(x_i, z_{i+\delta_x})}{P(x_i)P(z_{i+\delta_x})} \tag{5.4}$$

$$CRE(X : Y, \delta_x, \delta_y | Z) = \sum_{\delta_x, \delta_y} \sum_{i} P(x_{i-\delta_x}, y_{i-\delta_y}, z_i) \log \frac{P(x_{i-\delta_x}, y_{i-\delta_y}, z_i)}{P(x_{i-\delta_x}, y_{i-\delta_y})P(z_i)} \tag{5.5}$$

where x_i, y_i and z_i are the expression levels of X, Y and Z respectively at every time slice i, δ_x and δ_y are the time delays between variables (X, Z) and (Y, Z) respectively, a pair of $(x_i, z_{i+\delta})$ indicates a value pair of variable X at time slice i and variable Z at time slice $i + \delta$, and a triple $x_{i-\delta_x}, y_{i-\delta_y}, z_i)$ indicates a value triple of variables X, Y and X at time slice $i - \delta_x$, $i - \delta_y$ and i respectively.

5.3 Experiment

I implement the CD learning algorithm using BNT in MATLAB.

This section presents the experiments evaluating the CD learning algorithm on both artificial and real-life datasets. I also compare its performance with the existing learning methods: K2 [21], REVEAL [35, 39, 67, 75], DBmcmc [47], GeneNetwork [104] and TDNL.

5.3.1 Structure learning on artificial datasets

In this experiment, I use the same sets of synthetic datasets used in Section 4.4. The details of each set of datasets are shown in Table 5.1.

Dataset	No. of genes	No. of edges	Max. no. of parents
1	10	17	4
2	50	35	4
3	100	189	4
4	200	377	4

Table 5.1: Artificial datasets.

Dataset	K2	REVEAL	GeneNetwork	DBmcmc	TDNL	CD
	C/T	C/T	C/T	C/T	C/T	C/T
1	5/9	4/11	9/39	5/17	10/18	11/18
2	12/28	N/A	19/104	16/93	47/98	59/104
3	41/127	N/A	21/182	23/191	82/169	104/181
4	97/286	N/A	18/296	28/216	146/342	179/364

Table 5.2: The performance of parents selection from artificial datasets. T indicates the number of total learnt edges while C indicates the number of correct edges. N/A indicates the experiment is not available due to long running time (¿ 2 days).

The performance of K2, REVEAL, GeneNetwork, DBmcmc, TDNL and the CD learning algorithm on the four sets of synthetic datasets is shown in Table 5.2. From the table, the CD learning algorithm gives the best performance. More importantly, the performance of the CD learning algorithm is still quite good even when the network is big.

I also test the performance of the CD learning algorithm for datasets with different number of time slices. Figure 5.6 shows the convergence curves. The performance is indicated by sensitivity (= number of learnt true edges / number of total true edges) which is the proportion of corrected learned edges among all correct edges. It shows that independent of the number of variables in the dataset, performance improves as the number of time slices increases. In addition, the curves converge when the number of time slices is in the range of 200 to 400.

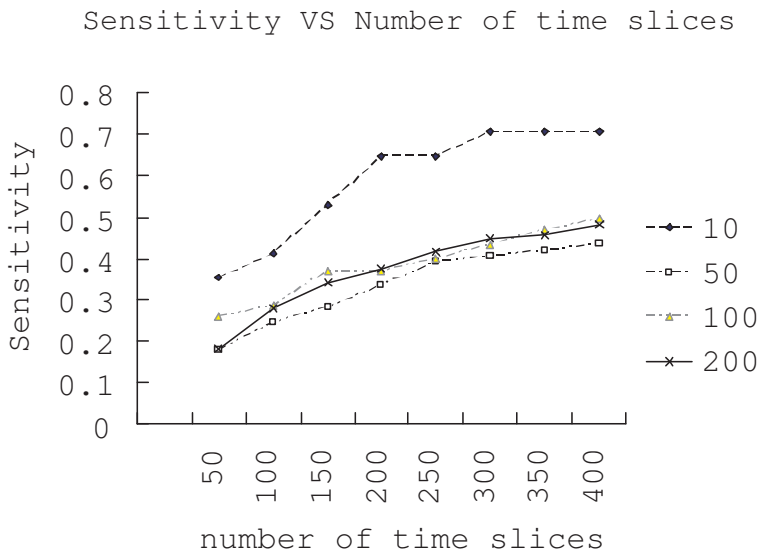

Figure 5.6: Convergence curves of sensitivity and number of time slices. Sensitivity = number of learnt true edges / number of total true edges. Sensitivities increase rapidly between 50 and 200 slices and start to converge between 150 to 300 slices.

5.3.2 Structure learning on yeast datasets

I apply the CD learning algorithm on the four yeast microarray datasets which are used in Chapter 4.4 to demonstrate the learning power of the algorithm.

The accuracy of the yeast cell cycle transcriptional network

The performance comparison between CD learning algorithm and K2, REVEAL, GeneNetwork, DBmcmc and TDNL is shown in Table 5.3. The verification criterion is same as that used in Section 4.4.As shown, the CD learning algorithm gives the best performance in each dataset.

	K2	REVEAL	DBmcmc	GeneNetwork	TDNL	CD
	C/T	C/T	C/T	C/T	C/T	C/T
Y_{s13}	2/12	5/17	12/33	8/31	14/29	19/34
Y_{s105}	14/59	N/A	21/69	33/146	37/84	41/75
Y_{c13}	6/20	4/16	10/31	7/21	13/31	14/28
Y_{c105}	29/84	N/A	16/72	10/66	33/81	39/83

Table 5.3: Comparison of learning performance. T indicates the number of total learnt edges and C indicates correct predicted edges. In row of Y_{s105} and Y_{c105}, the total learnt edges are the total edges between regulators and target genes. N/A means the result could not be found out as the algorithm has not finished execution within a reasonable time (2 days).

5.4 Conclusions

In this chapter, I have discussed several important issues in learning gene networks: collaboration among regulators, presence of regulatory complex and the various time delays problem. An new learning method, the CD learning algorithm, has been proposed to capture these issues. The varying time delay information is also integrated in the learning procedure. Experiments on both artificial and real-life datasets proved the effectiveness of the proposed algorithm. The conditional dependence approach can be used to solve other problems. For instance, conditional dependence may be useful in finding the co-expressed genes whose expression profiles are not correlated. This idea may also be used in finding co-expressed genes and important regulators in cell cycle data.

CHAPTER 6

Semi-fixed Bayesian network and semi-fixed structure EM algorithm

6.1 Introduction

When referring to gene regulation, most works simply model a gene being directly regulated by other genes. The regulation of genes actually occurs at various levels including transcription, translation, splicing, posttranslational protein degradation, and other processes [59]. To give a correct description of a gene network, we cannot simply consider gene expression level (that is, the level of mRNA transcription).

From biological knowledge, we know that protein plays a key role in a gene network [105]. In fact, protein-DNA interaction and Protein-Protein interaction are the main activities in a gene regulatory system [59]. Therefore, when predicting microarray gene expression data, proteins should be included as important factors. In Chapter 5, I have included the effect of proteins implicitly in the learning procedure. In this chapter, I propose considering the effect of proteins explicitly: Although their expression levels are still difficult to measure on a large scale, it would be good if we

could model them as hidden variables. The following are the advantages of modeling proteins as hidden variables:

- The model will be more meaningful, more interpretable, and closer to a real-life system [7, 27, 31, 32].
- In the model, proteins are decision-relevant. A network that does not consider hidden variables may omit some dependencies [7, 27].

Introducing hidden variables introduces advantages but also increases the complexity of network learning. Moreover, microarray gene expression datasets often have missing values. Considering the above challenging issues, the Bayesian network is a natural choice as it supports several principled methods for learning the relationships with incomplete data, both hidden variables and missing values [32]. The Bayesian network has been successfully applied to learn structures with hidden variables in several applications [7, 27, 31, 32]. The most widely used method for structure learning is the Expectation Maximization (EM) algorithm [32]. In the E step, the algorithm calculates the score of each possible structure using the structure and parameters learnt from the previous iteration. Selection of the structure and parameters to maximize the score is done in the M step. The procedure is repeated until convergence criteria are met. In this problem, such kind of learning is difficult since the algorithm needs to learn the relationship among the hidden variables and the observed variables. In addition, it is difficult to predetermine the correct number of hidden variables. To learn the optimum number of hidden variables is computationally complex. Besides, in the presence of hidden variables, the network is no longer decomposable, and this makes the learning difficult [32], as we will describe in Section 6.3).

I propose a system which models a gene network as a directed graph with hidden variables. In the model, the number of hidden variables is predefined using biological knowledge; in addition, the relationships between hidden variables and

observed variables are partially fixed. I also propose a modified EM algorithm that takes advantage of the semi-fixed structure to decompose the network, and thus allows us to learn the network efficiently. Also, an approximation method to perform inference on the joint probability of two genes is presented in order to speed up the learning procedure.

6.2 Modeling a gene network as a semi-fixed network with hidden Variables

In a gene regulation system, the regulation process can be divided into two main steps. The first step is gene expression, which is represented by $g_i \rightarrow r_i \rightarrow p_i$ where g_i is a gene, and r_i and p_i are the corresponding mRNA and Protein respectively. $g_i \rightarrow r_i$ is the transcription and $r_i \rightarrow p_i$ is the translation process. Transcription efficiency, which is represented by mRNA level, can be measured by a microarray. The second step is gene regulation, which is represented by $p_i \rightarrow g_j$. In this step, the generated protein p_i, possibly in collaboration with some other proteins, regulates the target gene g_i. These two steps are the most important steps in the gene regulatory system. All other steps such as protein degradation simply adjust the expression and regulation strengths. Let $Pa^j = \{g_1, ..., g_k\}$ denote a parent set of gene g_j such that each $g_i \in Pa^j$ regulates gene g_j. we note that all proteins $\{p_1, ..., p_k\}$ in Pa^j act in a combinative manner to regulate g_j. In other words, the proteins may combine to form a complex and then bind to the binding site of the target gene and regulate it, or bind to their own binding sites and then collaboratively regulate the target gene, or a mixture of the above two ways. An example of a network is shown in Figure 6.1(a). Considering a single node representing the proteins' combined influence as the direct regulator for the target gene, the model can be further simplified as shown in Figure 6.1(b).

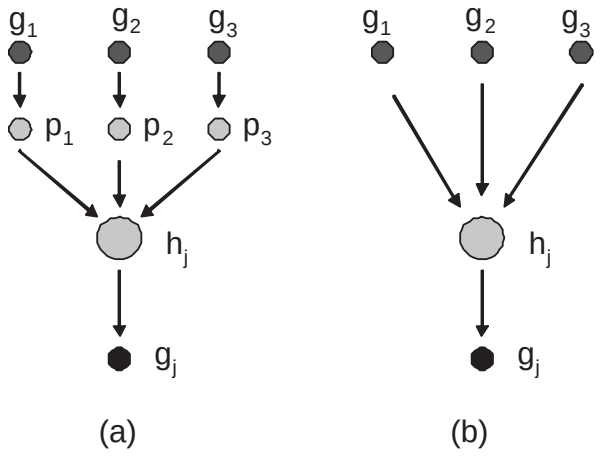

Figure 6.1: Simplified gene regulation system. (a) Gene expression system can be simplified as the interaction of genes and proteins. (b) The system can be further simplified since the combined proteins are the direct regulator of the target genes.

Based on the above discussion, for each gene g_j, I propose a hidden variable h_j, which represents the combination of a set of regulatory proteins expressed from Pa^j. Thereupon, Pa^j regulates h_j, and h_j regulates g_j. Based on the simplified regulation system, each gene is regulated by at most one hidden variable and the hidden variables are regulated by one or more genes. Such a model states that interaction exists only between $Gene \leftrightarrow Protein$ or among proteins, and there are no edges from gene to gene or hidden variable to hidden variable. This model is termed a semi-fixed network and is formally defined as follows:

- A semi-fixed network $N =< V, E >$ where vertices $V = O \cup H$, $O = \{g_1, ..., g_n\}$ is a set of observed variables representing the expression levels of genes, and $H = \{h_i, ..., h_n\}$ is a set of hidden variables representing the expression levels of combined proteins and E is the edge set.

- For each $e_k \in E$, $e_k = (g_i, h_j)$ or (h_j, g_i). Thus, N is a bipartite graph on two partitions O and H.

- For each $g_i \in O$, there is exactly one incoming edge (h_i, g_i) while there can be many outgoing edges.

- For each $h_i \in H$, there is exactly one outgoing edge (h_i, g_i) while there can be many incoming edges $\{(g_{i_1}, h_i), (g_{i_2}, h_i), ..., (g_{i_k}, h_i)\}$ where $Pa(h_i) = Pa^i = \{g_{i_1}, g_{i_2}, ..., g_{i_k}\}$.

Compared to learning a general network with several hidden variables, learning a semi-fixed network is easier since the number of hidden variables is fixed and the relationships between hidden variables and observed variables are partially known.

6.3 Semi-Fixed Structure EM Learning Algorithm

When there is no hidden variable in the network and no missing values in the gene expression data, the probability of the variables given a network structure can be expressed as the production of the probabilities of independent sub-networks:

$$P(X_1...X_n : N_o, D) = \prod_i P(X_i|Pa(X_i)) \tag{6.1}$$

where X_1 to X_n are observed variables and N_o denotes the network without hidden variables. D is a complete dataset.

The decomposability property reduces the learning difficulty in the following manner. With score functions such as minimum description length (MDL) and Bayesian scoring metric, learning the structure of a network can be decomposed to learning each independent sub-network separately [32, 36]. In contrast, in the case of incomplete data, the decomposability property is not valid and that makes the learning difficult [32]. However, if all parameters of all hidden variables are assigned, hidden variables are observed and thus in a sense, the incomplete dataset becomes "complete".

A useful property of a semi-fixed network is that when all parameters of hidden variables have been assigned (as the number of hidden variables and partial relationships of hidden variables, and observed variables are known to ensure that all hidden variables can be assigned parameters), the network can be decomposed into independent subnetworks. Each subnetwork comprises a gene g_j, a hidden variable h_j, and a parent set of h_j $(Pa(h_j))$(as shown in Figure 6.1(b)). The probability is decomposed as:

$$P(g_1...g_n, h_1...h_n : N_{o,h}, D) = \prod_i P(h_i|Pa(h_i)) \prod_i P(g_i|h_i) \tag{6.2}$$

where $g_1, g_2, ..., g_n$ are observed variables (genes), $h_2, h_2, ..., h_n$ are hidden variables and $N_{o,h}$ denotes the network with hidden variables. D is an incomplete dataset.

Based on different decomposition methods, the decomposition of scores is also different. In the work, Bayesian score is used. When the network N_o has no hidden variable, the score is decomposed as follows:

$$Score(N) = \sum_i Score(g_i|Pa(g_i)) \tag{6.3}$$

For a semi-fixed network $N_{o,h}$, hidden variables are included. We can show that its score can be decomposed because the hidden variables h_is are attached to the corresponding genes as follows:

$$Score(N_{o,h}) = \sum_i Score(h_i|Pa(h_i)) + \sum_i Score(g_i|h_i) \tag{6.4}$$

Proof: Given all hidden variables, $N_{o,h}$ is decomposable. By Formula 6.3, $Score(N_{o,h}) = \sum_i Score(v_i|Pa(v_i))$ where v_i is the variables, including both genes and proteins, and in $N_{o,h}$: $\{v_1, v_2, ..., v_{2n}\} = \{g_1, g_2, ..., g_n, h_1, h_2, ..., h_n\}$. Therefore, it is easy to see $\sum_i Score(v_i|Pa(v_i)) = \sum_i Score(h_i|Pa(h_i)) + \sum_i Score(g_i|Pa(g_i))$ where $Pa(g_i) = h_i$ in $N_{o,h}$. Thus, Formula 6.4 is proved.

With the aim of computing a network $N_{o,h}$ which maximizes $Score(N_{o,h})$ using the property in Equation 6.4, I propose an EM algorithm known as the Semi-fixed Structure EM ($SSEM$) algorithm to learn such a network. The principle of the EM algorithm is as follows: In each iteration, given an initial network structure N_{i-1} and a parameter set θ_{i-1} (which includes both the parameters of observed variables θ^o_{i-1} and hidden variables θ^h_{i-1}), the first step is to calculate the missing values and

hidden variables to build a complete dataset. The second step is to find a better structure N_i and a better parameter set θ_i based on the new complete dataset. A detailed description of the algorithm is shown at the end of this chapter.

For step 1, the missing values and hidden variables can be filled in as follows:

> Given N and θ, for a missing value of a variable v_i, supposing $Pa(v_i)$ is the parent set of v_i in N and $D = \{d_1, ..., d_k\}$ is the discrete values of v_i, we fill in the missing value of v_i by a vector $S = (s_1, ..., s_k)$ where $s_j = P(v_i = d_j | Pa(v_i))$.

This is illustrated in the following example:

Example: The variable v_1 has parents v_2 and v_3. As shown in Table 6.1 (a), there is a missing value in an instance: $\{v_1, v_2, v_3\} = \{(), 1, 1\}$ where () indicate a missing value. Suppose $Pr(v_i = 0 | v_2 = 1, v_3 = 1) = 0.4$ and $Pr(v_i = 1 | v_2 = 1, v_3 = 1) = 0.6$, we fill up the instance by $\{v_1, v_2, v_3\} = \{(0.4, 0.6), 1, 1\}$. This means the filled value adds 0.4 count to the case $v_1 = 0$ and 0.6 count to the case $v_1 = 1$. As shown in Table (b), there are 1.4 cases for which $v_1 = 0$ and 2.6 cases for which $v_1 = 1$. In other words, $Pr(v_1 = 0) = 1.4/4 = 0.35$ and $Pr(v_1 = 1) = 2.6/4 = 0.65$.

v_1	v_2	v_3
	1	1
1	0	1
1	1	1
0	1	1

(a)

v_1	v_2	v_3
(0.4, 0.6)	1	1
1	0	1
1	1	1
0	1	1

(b)

Table 6.1: Example of filling in missing values.

The values of hidden variables are treated as missing values too and are computed in a similar fashion. Given the filled dataset, N_{i-1} and θ_{i-1} , step 2 can learn N_i and θ_i using a general structure learning method. The general method to learn the structure from a complete dataset is to decompose the network into independent subnetworks. Then, the structure of each subnetwork is learnt independently. Since we have fixed the partial structure, we can decompose it as independent subnetworks, each of which has a target gene, a hidden variable and a parent set of hidden variables (similar to Figure 6.1 (b)). The main objective is to find the optimal parents for each hidden variable. Learning parents from a large number of candidates is difficult. As a gene network is a sparse network [88], a popular technique is to first measure the dependencies of candidates to the target variable and choose the best k genes as candidate parents. Then, the search for the parents from the candidate parent set can be performed. Friedman et al. [36] proposed calculating the dependency by KL-divergence (as Equation 6.5):

$$M_{disc}(X, Y|M) = \sum_{x,y} \hat{P}(x, y) \log \frac{\hat{P}(x, y)}{P_M(x, y)} \tag{6.5}$$

where X and Y are variables, $M_{Disc}(X, Y|M)$ is the dependence between X and Y with respect to the network M, $\hat{P}$ denotes the observed frequency, P_M is the estimated probability given M, and x and y are the values of X and Y.

Though we can compute $P_M(g_i, h_i)$ given the filled dataset, we cannot compute $P(g_j, h_i)$ to measure the dependency of g_j and h_i, as h_i is hidden. I propose an alternate way to measure the dependency: in each subnetwork, the target gene is the only descendant of the hidden variable. The probabilities of the hidden variable is passed to the target gene. Therefore, $MI(g_j, h_i|M)$ can be reflected by $MI(g_j, g_i|M)$. Thus, MI can be calculated as follows:

$$MI(g_j, g_i|M) = \sum_{g_j, g_i} \hat{P}(g_j, g_i) \log \frac{\hat{P}(g_j, g_i)}{P_M(g_j, g_i)} \tag{6.6}$$

where $\hat{P}(g_j, g_i)$ is the observed probability distribution of genes g_j and g_i while $P_M(g_j, g_i)$ is the estimated probability distribution of gene g_j and g_i in the network M. Performing inference on joint probabilities in a big dataset is quite time intensive. Thus, I approximate the joint probability of g_i and g_j as follows:

- If $g_j \in Pa(g_i)$, then $g_j \rightarrow h_i \rightarrow g_i$. $P_M(g_j, g_i) = P_M(g_i|g_j)P_M(g_j) \approx P_M(g_i|h_i)P_M(h_i|g_j)P_M(g_j)$. Note that in this case, $P_M(g_j, g_i) \neq P_M(g_i, g_j)$.

- Otherwise, g_i and g_j are conditionally independent and $P_M(g_j, g_i)$ can be approximated as $P_M(g_j)P_M(g_i)$.

By the above approximation, only the joint probabilities of the gene pairs with parent-child relationships need to be calculated.

The detail of the iterative algorithm is shown in Figure 6.2:

- In iteration i, given N_{i-1}, θ_{i-1} and the original incomplete dataset D^0.
- In E step, the missing values and the values of hidden variables of D^0 are filled in based on the inference of N_{i-1} and θ_{i-1}. Then, we obtain a complete dataset D_{i-1}. The structure N_i is learnt based on D_{i-1} by maximizing $Score(N_i : \theta_{i-1}, D_{i-1})$. Because of decomposability, we learn the parents for each gene g_i independently:

  (1) $CPS_i = PS_i$. Initially, $PS_i = \emptyset$

  (2) Include $(k - |CPS_i|)$ genes with the best MI scores with respect to g_i into the candidate parent set CPS.

  (3) Revise PS_i using LBM so that $PS \in CPS$ with the maximum $Score(SS_i : \theta_{i-1}, D_{i-1})$, where SS_i is a substructure $PS \rightarrow h_i \rightarrow g_i$.

  - Repeat the procedure until convergence.

- In the M step, θ_i is learnt based on N_i which maximizes $Score(N_i : \theta_i, D_{i-1})$.
- Repeat the procedure until convergence or till the predefined number of iterations is reached.

Figure 6.2: Outline of SSEM.

In the above procedure, the network structure and parameters are refined iteratively. In each iteration, the parent set of a target gene is initially set as the parents of last iteration and revised by improving the Bayesian score. Bayesian score is calculated by likelihood. Thus, the likelihood of the network is not decreased by each iteration.

To get the initial network structure N_0 and parameters θ_0:

- learn D^0 as a complete dataset. Pa^i for each g_i and the dependent probabilities are obtained.
- for each gene g_i, add a hidden variable h_i. Set the $Pa(h_i) = Pa^i$ and $P(h_i|Pa(h_i)) = P(g_i|Pa^i)$.
- Complete dataset D_0 by filling D^0 based on the network structure and dependent probabilities obtained.
- N_0 and θ_0 can be learnt from D_0.

6.4 Experimental results and comparison

The learning algorithm is implemented in MATLAB and BNT[1]. In the following, I present the experimental results on both artificial and real-life gene expression datasets.

6.4.1 Experiment on artificial datasets

[1]Bayes network toolbox, http://www.ai.mit.edu/~murphyk

Dataset	No. of genes	No. of edges	Max. No. of parents
1	10	14	4
2	20	35	4

Table 6.2: Specification of the synthetic datasets.

As the artificial datasets used in previous chapters are generated based on the principle that one gene is regulated by another gene directly and there is no hidden variable in the system, SSEM is not appropriate for this type of data. I generate artificial datasets by simulating gene expressions with hidden variables: a network containing k genes and k proteins is built, then the parameter of the network is randomly assigned; a dataset is generated based on the structure and parameter; I then delete the data rows of proteins and revise the network: an edge from a gene to a protein is redirected as a set of edges from the gene to the children of the protein while the edges from genes to proteins and proteins to genes are deleted. I generate 20 datasets, 10 of which has 10 genes and 10of which has 20 genes. The specifications of the two sets of datasets are averaged in Table 6.2. I then apply K2, REVEAL, GeneNetwork, DBmcmc, TDNL, CD and SSEM to them. The learning result is shown in Table 6.3. As shown, SSEM gives the best performance.

Dataset	K2	REVEAL	GeneNetwork	DBmcmc	TDNL	CD	SSEM
	C/T	C/T	C/T	C/T	C/T	C/T	C/T
1	1/8	1/11	3/29	2/16	7/20	9/21	10/18
2	4/23	2/21	4/32	8/35	14/38	20/45	23/41

Table 6.3: The performance of parents selection from artificial datasets. T indicates the number of total learnt edges while C indicates the number of correct edges.

6.4.2 Experiments on real-life data

To demonstrate the learning ability of SSEM, I apply it to two yeast gene expression datasets, Y_{s13} and Y_{c13} which are used in previous two chapters (Y_{s105} and Y_{c105} are not used because of the long running time which is more than two days). The structure of the network is shown in Figure 6.3. Since various time delays exist in the regulation system [63], a gene in time slice i may regulate another gene in time slice i to $i + k$ (k indicates the maximum time delay). Thus, I transform the dataset with various time delay to the dataset without time delays, as described in Section 4.2, then learn it by SSEM. Since a complex is more complicated than a single gene or protein, I set more states to a complex. In this experiment, I set two states to a gene while four states for a complex. The learning performance is compared with that of K2, REVEAL, GeneNetwork, TDNL and CD learning algorithm. The result is shown in Table 6.4.

	K2	REVEAL	DBmcmc	GeneNetwork	TDNL	CD	SSEM
	C/T	C/T	C/T	C/T	C/T	C/T	C/T
Y_{s14}	2/12	5/17	12/33	8/31	14/29	18/34	20/33
Y_{c14}	6/20	4/16	10/31	7/21	13/31	14/28	18/30

Table 6.4: Comparison of learning performance. T indicates the number of total learnt edges and C indicates correct predicted edges.

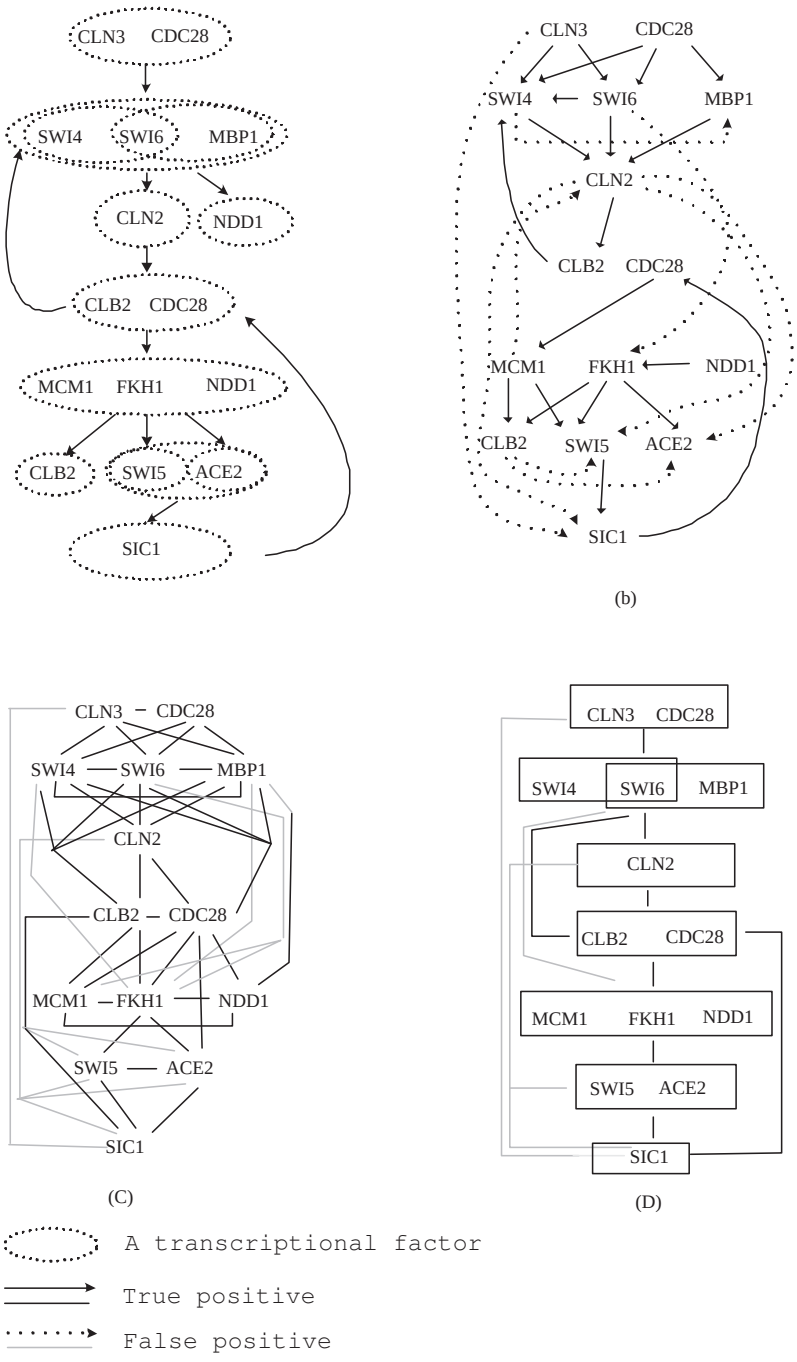

Figure 6.3: Learning performance of $SSEM$ on a real-life gene network. (a) Yeast cell cycle transcriptional regulatory subnetwork. (b) The structure learnt by $SSEM$. There are 29 edges with confidence no smaller than 0.6. Among them, 20 edges are verified as true positives by (a). (c) Markov features with confidence no smaller than 0.6 learnt by $SSEM$. There are 49 Markov features. Among them, 39 features can be verified. (d) Learnt cell cycle regulatory network which is simplified from (c).

Given an insufficient dataset, a Bayesian network may give a set of models which explain the data equally well [78]. For further analysis, I use statistical confidence to measure the likelihood of a learnt edge or a statistic feature [30, 34], which is described in Chapter 4.

Figure 4.6(b) shows the 29 edges learned from Y_{s13} by $SSEM$ with confidence greater than 0.6. Among them, 20 edges can be verified by Figure 4.6(a).

Besides the direct regulatory relationship between genes, I also employ a popular statistic feature, Markov relation, which has been described in Section 4.4. A Markov feature indicates whether two genes are involved in the same biological event and it has been regarded as a criterion by several works to evaluate the performance of gene network reconstruction [30, 34]. A Bayesian network is a model of dependencies among random variables, rather than causality. $A \rightarrow B$ and $B \rightarrow A$ are the alternative ways of describing that A and B are not independent on each other. Markov relation is helpful in discovering dependencies regardless of the directions of the edges. Moreover, in a gene regulatory system, there are many transcription factors and transcription complexes consisting of two or more proteins working collaboratively. Such collaborations cannot be uncovered by directed regulatory edges but can be discovered by Markov features.

When a gene (or a complex) activates or inhibits a transcription complex, it is possible that there is no direct edge from the regulator to any gene of the complex and vice versa. For example, $CDC28$ activates $SWI4-SWI6$ and $SWI6-MBP1$ together with $CLN3$ and inhibits them together with $CLB2$. Meanwhile, there is no direct edge from $CDC28$ to $SWI4$, $SWI6$ and $MBP1$ in YPD. The use of the hidden variables in this model ensures that we can find such regulatory relationships, for example, $CDC28 \rightarrow SWI4$, $CDC28 \rightarrow SWI6$, $CDC28 \rightarrow MBP1$, $SWI4 \rightarrow CLN2$, $SIC3 \rightarrow CLB2$, etc.

I list the Markov features with confidence, which are learned from Y_{s13} by SSEM, greater than 0.6, as shown in Figure 6.3(c). There are altogether 49 Markov

features, in which 39 can be verified by Figure 6.3(a). It is clear that there are strong interactions among the genes comprising a complex and between complexes which are connected in Figure 6.3(b). I then simplify the network using the transcriptional factors or complexes, which function as a unit in the regulatory system. I take them as the node and the Markov relations between them as indirected edges, as shown in Figure 6.3(d). I discover the complete cell cycle regulatory network, together with four false edges between $CLN3 - CDC28$ and $SIC1$ (which is weak since between them there are only one Markov feature between $CLN3$ and $SIC1$), $CLN2$ and $SIC1$, $CLN2$ and $SWI5 - ACE2$, and SBF/MBF and SFF.

6.5 Conclusion

The semi-fixed hidden variable model introduces hidden variables to model the important components of a gene network, i.e., regulatory proteins. Modelling hidden variables which exist in the system can reduce the learning data needed [10] while omitting hidden variables will usually miss some dependencies in the model given limited amount of data [27]. Also, without modelling hidden variables, dependencies among co-children of a regulators are not blocked. Therefore, there might be some co-children who are selected to be parents of a gene. These false parents will be the noise to prevent true parents being selected. Moreover, this model makes the network decomposable even when the proteins which are not measured are considered and parts of the gene network are fixed using biological knowledge. Thus, a single hidden variable for each gene is meaningful as it can find more dependencies with limited data, avoid false positive from co-children and make the model more realistic. Compared to the current work on gene networks, I have integrated the biological knowledge to build a semi-fixed hidden variable model which is effective and reflects real-life gene regulatory systems. It could be a basis of more complicated models to model gene networks. It increases learning performance and is capable

of finding several regulatory relationships which are difficult to be discovered with traditional methods. As this model can model protein complexes which plays a key role in many biological systems, this model may be employed in other applications in computational biology. In addition to the model, this chapter has presented an effective learning algorithm to learn this model. The main disadvantage of this model is that it needs more computing resource to model the hidden variables and it requires more iterations to converge. The running time for learning a dataset with 100 genes might take more than 2 days. An more efficient algorithm is needed for big datasets.

CHAPTER 7

Conclusion and Future Work

7.1 Conclusion

Learning gene networks is an important and difficult task. With the use of microarray data, it has gained even more attention and has become one of the central tasks in the post-genome era.

Several facts hinder the research on learning gene networks from microarray gene expression data: the NP-hard property, the data (dimension) problem, the stochastic nature, and so on. Though many methods have been proposed to tackle these problems, few of them could give reasonable results on large-scale data. Moreover, some important biological facts have been overlooked in previous works: various time delays, collaboration among parents in a biological regulation system including complexes and hidden variables (proteins), etc.

In the thesis, I have tried to tackle these problems using the Bayesian network framework together with the following enhancements:

- **Time delayed Bayesian network:** Most research work on learning gene networks either assumes that there is no time delay in gene expression or

that there is a constant time delay. My proposed model goes a step further and shows how Bayesian Networks can be applied to represent various time delay relationships. In my approach, the traditional Bayesian network is enhanced with a time-delay model in order to represent various time delays in a gene network. Transformation is applied to shift the time-delayed network learning problem to the traditional Bayesian network learning problem. The intractability of the network learning algorithm is handled with the use of improved mutual information criteria. The increase in the number of parameters to be learnt is accounted for by my proposed efficient structure learning algorithm, "Learning By Modification", which is suited to learning the sparse structure of a gene network.

- **Conditional dependency learning algorithm:** When we learn a gene network from a large dataset, one key problem is that the parent-child correlation is insufficient to help us extract the correct regulatory relationships because of the complexity of the regulatory system. In my proposed model, in addition to parent-child correlation, I take two additional factors into account: (1) collaboration among regulators, and (2) formation of a regulatory complex. I use conditional dependence to extract co-regulatory relationships between regulatory pairs. A new learning method, called the Conditional Dependence Learning algorithm, is proposed to extract the underlying structure of the gene network based on the regulatory complex. The various time delays information is also integrated in the learning procedure.

- **Semi-fixed hidden variable model:** Most existing works learn gene networks by assuming one gene provokes the expression of another gene directly. This leads to an over-simplified model. In my proposed model, I show that gene regulation is a complex problem with many hidden variables. My proposed semi-fixed model represents the gene network as a Bayesian network

with hidden variables. The semi-fixed hidden variable model introduces hidden variables to model the important components of a gene network, i.e., proteins. The model provides for a decomposable network. Moreover, parts of the network are fixed using biological knowledge. Compared to the current works on gene networks, the semi-fixed hidden variable model is effective since it takes into account important biological knowledge and thus reflects a real-life gene expression regulatory system. In addition, I have presented an effective algorithm that is suitable to learning the partially fixed networks that the model reflects.

7.2 Discussion

This thesis has take into account some important biological information, and has obtained biologically significant results. However, there are still some drawbacks to the work:

- Some information was omitted in the proposed models which are still simplified models even though they may be more sophisticated than previous ones. AGene regulation and expression systems are complicated. Given the gene expression data available, we cannot monitor the intermediate steps of gene regulation such as protein translation efficiency, protein degeneration rate and so on. With partial information, it is difficult to produce a complete picture. Even if we include proteins as hidden variables, we cannot model all hidden information such as protein translation level and protein degeneration rate because of learning difficulty. Some additional information is needed before we can obtain a more in-depth understanding.

- The number of genes in a genome may reach hundreds of thousands. Currently, our method can complete the learning of several thousands genes in about one week. This running time is not reasonable when we apply the algorithms in

a dataset with hundreds of thousand genes. In particular, the procedure of inferencing parameters is quite slow, and a more efficient algorithm needs to be devised.

- All models in the thesis have obtained some biologically significant results. However, they are far from enough for building a genome-wide network to predict biological relationships among genes and discover the regulation mechanism. Even in a subnetwork with tens or hundreds genes, only about half of the regulatory relationships can be discovered. Given a big genome, the accuracy is assumed to decrease further. More effective algorithms are needed.

7.3 Future Work

Despite its limitations, the work in this thesis may be used as the basis for future developments:

- Integrating with more biological information. Recent papers [49, 50, 95] have reported that combining different types of biological data sources is useful in determining the structure of a gene network. The frequently used data sources are protein interaction, transcription factor, cell cycle information, and so on. Given more related information, the models proposed in this thesis could be more realistic and the results they produce would become more biologically relevant.

- Combining multi datasets. One of the main problems in learning gene networks is the insufficiency of microarray data. Accurate results cannot be deduced from thousands of genes tested in tens of measurements. If we could combine several datasets with or without the same time intervals, we would have a higher chance of obtaining correct pictures of networks. Several issues need to be considered to achieve efficient dataset combinations: 1) The datasets

may be obtained under different scenarios. The expression and regulation patterns of the datasets may be different. There will be some overlap and some inter-supplement, which needs to be handled carefully. 2) The datasets may have different time intervals. For example, in Spellman's dataset [90], the time intervals of the subsets varies from 7 minutes to 30 minutes. To combine them, one possible solution is to interpolate the datasets to let them have the same time interval. The interpolated data would serve as given data, thus the new data would not create any new information but only contain the information from the given data.

- Continuous value. In the thesis, I have worked with discrete values. Though there is no evidence to prove that continual value models give more significant results, the loss of the information in the process of discretization may lead to some bias in the resulting network. I plan to work on a new method that can handle continual values. The probability distribution can be calculated by a density function given the continuous values.

- Use of gene over-expression and disruption data. Such types of data provide the chance to observe the effect of specific genes directly. The goodness of these data has been reported in some recent papers [1, 78, 79, 81].

- More effective and efficient algorithms. Finding candidate parents is critical for final learning performance. Mutual information cannot reflect real regulatory dependency between gene pairs. Conditional relative entropy (CRE) can find more appropriate dependency between gene pairs. However, calculating CRE requires extra time. When the number of genes grows, running time increases. Sampling may be a good approach to speed up CRE calculation; this approach is used in joint probability inference and has been proved useful. In addition, association rules mining might be a good choice in candidate parent selection as they consider all possible parent combinations of the target gene.

A recently published association rule mining algorithm, FARMER [20], is especially useful when applied to microarray datasets. It uses row enumeration instead of column enumeration. In each microarray dataset, a row is a replicate of the data while a column is a variable. A microarray dataset generally has many variables but only tens of replicates. Therefore, FARMER is quite efficient and effective when applied to microarray datasets.

BIBLIOGRAPHY

[1] T. Akutsu, S. Kuhara, O. Maruyama, and S. Miyano. Identification of Gene Regulatory Networks by Strategic Gene Disruptions and Gene Overexpressions. In *the ninth annual ACM-SIAM symposium on Discrete algorithms*, pages 695 – 702, 1998.

[2] T. Akutsu, S. Miyano, and S. Kuhara. Identification of Genetic Getworks from a Small Number of Gene Expression Patterns Under the Boolean Network Model. In *Pacific Symposium on Biocomputing (PSB)*, pages 17–28, 1999.

[3] T. Akutsu, S. Miyano, and S. Kuhara. Algorithms for Identifying Boolean Networks and Related Biological Networks Based on Matrix Multiplication and Fingerprint Function. *Journal of Computational Biology*, 7(3/4):331–343, 2000.

[4] T. Akutsu, S. Miyano, and S. Kuhara. Algorithms For Inferring Qualitative Models of Biological Networks. In *Pacific Symposium on Biocomputing (PSB)*, pages 293–304, 2000.

[5] T. Akutsu, S. Miyano, and S. Kuhara. Inferring Qualitative Relations in

Genttic Networks and Metabolic Pathways. *Bioinformatics*, 16(8):727–734, 2000.

[6] A. Arkin, P. shen, and J. Ross. A Test Case of Correlation Metric Construction of A Reaction Pathway from Measurements. *Science*, 277:1275–1279, 1997.

[7] G.H. Barbara, A. Jameson, and F. Witting. Learning Bayesian Networks with Hidden Variables for User Modeling. In *Proceedings of the International Joint Conferences on Artificial Intelligence Workshop "Learning About Users"*, pages 29–34, 1999.

[8] M.J. Beal, F.Falciani, Z. Ghahramani, C. Rangel, and D.L. Wild. A Bayesian Approach to Reconstructing Genetic Regulatory Networks with Hidden Factors. *Bioinformatics*, 21(3):349–356, 2005.

[9] S. R. Biggar and G.R. Crabtree1. Cell Signaling Can Direct Either Binary or Graded Transcriptional Responses. *EMBO journal*, 20:3167–3176, 2001.

[10] J. Binder, D. Koller, S. Russell, , and K. Kanazawa. Adaptive Probabilistic Networks with Hidden Variables. *Machine Learning*, 29:213–244, 1997.

[11] S.G. Bottcher and C. Dethlefsen. Learning Bayesian Networks with R. In *Internaltional Workshop on Distributed Statistical Computing(DSC2003)*, 2003.

[12] X. Boyen, N. Friedman, and D. Koller. Discovering the Hidden Structure of Complex Dynamic Systems. In *Uncertainty in Artificial Intelligence*, 1999.

[13] P. Brazhnik, A.D.L. Fuente, and P. Mendes. Gene Networks: How To Put The Function in Genomics. *TRENDS in Biotechnology*, 1(20):467–472, 2002.

[14] A. Brazma and J. Vilo. Gene Expression Data Analysis. *FEBS letters*, 480:17–24, 2000.

[15] K.C. Chen, T.Y. Wang, H.H. Tseng, C.Y.F. Huang, and C.Y. Kao. A Stochastic Differential Equation Model for Quantifying Transcriptional Regulatory Network In Saccharomyces Cerevisiae. *Bioinformatics*, 21(12):2883–2890, 2005.

[16] T. Chen, V. Filkov, and S.S. Skiena. Identifying Gene Regulatory Networks from Exprimental Data. In *International Conference on Research in Computational Molecular Biology (RECOMB)*, pages 94–103, 1999.

[17] T. Chen, H.L. He, and G.M. Church. Modeling Gene Expression with Differential Equations. In *Pacific Symposium on Biocomputing (PSB)*, volume 4, pages 29–40, 1999.

[18] R.J. Chou, M.J. Campbell, E.A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T.G. Wolfsberg, A.E. Gabrielian, D. Landsman, D.J. Lockhart, and R.W. Davis. A Genome-Wide Transcriptional Analysis of The Mitotic Cell Cycle. *Molecular Cell*, 2:65–73, 1998.

[19] O. Cinquin and J. Demongeot. Positive and Negative Feedback: Striking a Balance Between Necessary Antagonists. *Journal of Theoritical Biology*, 216:229–241, 2002.

[20] G. Cong, AKH. Tung, X. Xu, F. Pan, and J. Yang. FARMER: Finding Interesting Rule Groups in Microarray Datasets. In *ACM SIGMOD international conference on Management of data*, pages 143–154, 2004.

[21] G.F. Cooper and E. Herskovits. A Bayesian Method for The Induction of Probabilistic Networks from Data. *Machine Learning*, 9:309–347, 1992.

[22] P. Dagum and A. Galper. Time Series Prediction Using Belief Network Models. *International Journal of Human-Computer Studies*, 42(6):617–632, 1995.

[23] P. D'haeseleer. *Reconstructing Gene Networks from Large Scale Gene Expression Data*. PhD thesis, University of New Mexico, 2000.

[24] P. D'haeseleer, S. Liang, and R. Somogyi. Genetic Network Inference: From Co-Expression Clustering to Reverse Engineering. *Bioinformatics*, 16(2):707–726, 2000.

[25] B.E. Dutilh. *Analysis of Data from Microarray Experiments,the State of the Art in Gene Network Reconstruction*. PhD thesis, Department of Theoretical biology and Bioinformatics, Utrecht University, 1999.

[26] G. Elidan, N. Lotner, N. Friedman, and D. Koller. Discovering Hidden Variables: A Structure-Based Approach. In *Neural Information Processing Systems (NIPS)*, pages 479–485, 2000.

[27] G. Elidan, N. Lotner, N. Friedman, and D. Koller. Discovering Hidden Variables: A Structure-Based Approach. In *Neural Information Processing Systems (NIPS)*, pages 479–485, 2000.

[28] S. Fiering, E. Whitelaw, and D.I.Martin. To Be or Not To Be Active: The Atochastic Nature of ENhancer Action. *Bioessays*, 22:381–387, 2000.

[29] V. Filkov, S. Skiena, and J.Z. Zhi. Analysis Techniques for Microarray Time-Series Data. In *International Conference on Research in Computational Molecular Biology (RECOMB)*, pages 124–131, 2001.

[30] Friedman. Inferring Cellular Networks Using Probabilistic Graphical Models. *Science*, 303(6):799–805, 2004.

[31] N. Friedman. Learning Belief Networks in The Presence of Missing Values and Hidden Variables. In *Proc. 14th International Conference on Machine Learning*, pages 125–133, 1997.

[32] N. Friedman. The Bayesian Structure EM Algorithm. In *Uncertainty in Artificial Intelligence*, pages 129–138, 1998.

[33] N. Friedman. Inferring Cellular Networks Using Probabilistic Graphical Models. *Science*, 33:799–805, 2004.

[34] N. Friedman, M. Linial, I. Nachman, and D. Peer. Using Bayesian Networks to Analyze Expression Data. In *International Conference on Research in Computational Molecular Biology (RECOMB)*, pages 127–135, 2000.

[35] N. Friedman, K. Murphy, and S. Russell. Learning the Structure of Dynamic Probabilistic Networks. In *Uncertainty in Artificial Intelligence*, pages 139–147, 1998.

[36] N. Friedman, I. Nachman, and K. Peer. Learning Bayesian Network Structure from Massive Datasets: the "Sparse Candidate" Algorithm. In *Uncertainty in Artificial Intelligence*, pages 206–215, 1999.

[37] B. Futcher. Transcriptional Regulatory Networks and Yeast Cell Cycle. *Current Opinion in Cell Biology*, 14:676–683, 2002.

[38] P. Goldsbrough. *Biotechnology in Agriculture*. Lecture Notes, Department of Horticulture & landscape architechture Purdue University, West Lafayette, IN USA, 2 edition, 2001.

[39] L. Gransson and T. Koski. Using a Dynamic Bayesian Network to Learn Genetic Interactions. *Technical Report, Graduate School of Biomedical Research, Linkoping University.*, 2002.

[40] R. Guthke, U. Moller, M. Hoffmann, F. Thies, and S. Topfer. Dynamic Network Reconstruction from Gene Expression Data Applied to Immune Response During Bacterial Infectioin. *Bioinformatics*, 21(8):1626–1634, 2005.

[41] A.J. Hartemink, D.K. Gifford, T.S. Jaakkola, and R.A. Young. Using Graphical Models and Genomic Expression Data to Statistically Validate Models of Genetic Regulatory Networks. *Pacific Symposium on Biocomputing (PSB)*, 6:422–433, 2001.

[42] J. Hasty, D. McMillen, F. Isaacs, and J.J. Collins. Computational Studies of Gene Regulatory Networks: In Numero Molecular Biology. *Nature Reviews Genetics*, 2(4):268–279, 2001.

[43] D. Heckerman. A Tutorial on Learning with Bayesian Networks. *Microsoft Research Technical Report*, (MSR-TR-95-06), 1995.

[44] D. Heckerman, D. Gerger, and K.M. Chickering. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning*, 20(3):197–243, 1995.

[45] S. Huang. Gene Expression Profiling, Genetic Networks, and Cellular States: An Intergrating Concept for Tumori-genesis and Drug discovery. *Journal of Molecular Medicine*, 77:469–480, 1999.

[46] D.A. Hume. Probability in Transcriptional Regulation and Its Implications for Leukocyte Differentiation and Inducible Gene Expression. *Blood*, 96:2323–2328, 2000.

[47] D. Husmeier. Sensitivity and Specificity of Inferring Genetic Regulatory Interactions From Microarray Experiments with Dynamic Bayesian Networks. *Bioinformatics*, 19(17):2271–2282, 2003.

[48] S. Imoto, T. Goto, and S. Miyano. Estimation of Genetic Networks and Functional Structures Between Genes by Using Bayesian Networks and Nonparametric Regression. In *Pacific Symposium on Biocomputing (PSB)*, volume 7, pages 175–186, 2002.

[49] S. Imoto, T. Higuchi, T. Goto, K. Tashiro, S. Kuhara, and S. Miyano. Combining Microarrays and Biological Knowledge for Estimating Gene Networks via Bayesian Network. In *Computational Systems Bioinformatics conference*, 2003.

[50] G.V. Irit, A. Tanay, D. Raijman, and R. Shamir. The Factor Graph Network Model for Biological Systems. In *International Conference on Research in Computational Molecular Biology (RECOMB)*, 2005.

[51] S.M. Jane and J.M Cunningham. Molecular Mechanism of Hemoglobin Switching. *International Journal of Biochemical Cell Biology*, 28(11):1197–1209, 1996.

[52] P.A.O. Sharp J.D. Parvin, H.T. Timmers. Promoter Specificity of Basal Transcription Factors. *Cell*, 68(6):1135–1144, 1992.

[53] H.D. Jong. Modeling and Simulation of Genetic Regulatory System: A Literature Review. *Journal of Computational Biology*, 9(1):67–163, 2002.

[54] Hidde De Jong. Modeling and Simulation of Genetic Gegulatory Systems: A Literature Review. *Journal of computational biology*, 9(1):67–103, 2002.

[55] R. Karmakar and I. Bose. Graded and Binary Responses in Stochastic Gene Expression. *Physical Biology*, 1:197–204, 2004.

[56] S.A. Kauffman. Metabolic Stability and Epigenesis in Randomly Constructed Genetic Nets. *Journal of Theoretical Biology*, 22:437–467, 1973.

[57] S. Kim, S. Imoto, and S. Miyano. Dynamic Bayesian Network and Nonparametric Regression for Nonlinear Modeling of Gene Networks from Time Series Gene Expression Data. *Biosystems*, 75:57–65, 2004.

[58] M.S. Ko, H. Nakauchi, and N. Takahashi. The Dose Dependence of Glucocorti-Coid-Inducible Gene Expression Results from Changes in Tthe number of Transcriptionally Cctive Templates. *EMBO Journal*, 9:2835–2842, 1990.

[59] F.A. Kolpakov, E.A. Ananko, G.B. Kolesov, and N.A. Kolchanov. Genenet: a Gene Network Database and its Automated Visualization. *Bioinformatics*, 14:529–537, 1998.

[60] W.D. Laat and F. Grosveld. Spatial Organization of Gene Expression: The Active Chromatin Hub. *Chromosome Research*, 11:447–459, 2003.

[61] H. Lahdesmaki, I. Shmulevich, and O. Yli-Harja. On Learning Gene Regulatory Networks Under the Boolean Network Model. *Machine Learning*, 52:147–167, 2003.

[62] W. Lam and F. Bacchus. Learning Bayesian Belief Networks: An Approach Based on the MDL Principle. *Computational Intelligence*, 10:269–293, 1994.

[63] A.K. Lee, S.H. Sung, Y.C. Kim, and S.G. Kim. Inhibition of Lipopolysaccharide-Inducible Nitric Oxide Synthase TNF-α and COX-2 Expression by Sauchinone Effects on I-κBα Phosphorylation, C/EBP and AP-1 Activation. *British Journal of Pharmacology*, 139:11–20, 2003.

[64] P.H. Lee and D. Lee. Modularized Learning of Genetic Interaction Networks from Biological Annotations and mRNA Expression Data. *Bioinformatics*, 21(11):2739–2747, 2005.

[65] P.P. Levings and J. Bungert. The Human β-Globin Locus Control Region. *European Journal of Biochemistry*, 269:1589–1599, 2002.

[66] B. Lewin. *Genes*. Oxford University Press, 7 edition, December 1999.

[67] S. Liang, S. Fuhrman, and R. Somogyi. REVEAL,a General Reverse Engineering Algorithm for Inference of Genetic Network Architectures. In *Pacific Symposium on Biocomputing (PSB)*, volume 3, pages 18–29, 1998.

[68] TF. Liu, WK. Sung, and A. Mittal. Learning Multi-Time Delay Gene Network Using Bayesian Network Framework. In *The 16th IEEE International Conference on Tools with Artificial Intelligence*, 2004.

[69] M. Louis and A. Becskei. Binary and Graded Responses in Gene Networks. *Science stke*, 143:pe33–33, 2002.

[70] M. Louis and A. Becskei. Binary and Graded Responses in Gene Networks. *Science stke*, 143:pe33–33, 2002.

[71] M. Schena M, D. Shalon, R.W. Davis, and P.O. Brown. Quantitative Monitoring of Gene Expression Patterns with a Complementary DNA Microarray. *Science*, 270(5235):467–470, 1995.

[72] Y. Maki, D. Tominaga, M. Okamoto, S. Watanabe, and Y. Eguchi. Development of a System for the Inference of Large Scale Genetic Networks. In *Pacific Symposium on Biocomputing (PSB)*, volume 6, pages 446–58, 2001.

[73] E. Martinez. Multi-Protein Complexes in Eukaryotic Gene Transcription. *Plant Molecular Biology*, 50(6):925–47, 2002.

[74] H.H. McAdams and A. Arkin. Stochastic Mechanisms in Gene Expression. *Proceedings of the National Academy of Sciences*, 94(3):814–819, 1997.

[75] K. Murphy and S. Mian. Modelling Gene Expression Data Using Dynamic Bayesian Networks. *Technical Report, Computer Science Division, University of California, Berkeley, CA.*, 1999.

[76] I. M. Ong, J. D. Glasner, and D. Page. Modelling Regulatory Pathways in E. coli from time Series Expression Profiles. *Bioinformatics*, 18:241–248, 2002.

[77] J. Pearl and T.S. Verma. A Theory of Inferred Causation. In *International Conference on Principles of Knowledge Representation and Reasoning*, pages 441–452, 1991.

[78] D. Pe'er, A. Regev, G. Elidan, and N. Friedman. Inferring Subnetwork from Perturbed Expression Profiles. *Bioinformatics*, 17(Suppl. 1):s215–s224, 2001.

[79] J.J. Rice, U.H Tu, and G. Stolovitzky. Reconstructing Biological Networks using Conditional Correlation Analysis. *Bioinfomatics*, 21(6):765–773., 2005.

[80] S. Rogers and M. Girolami. A Bayesian Regression Approach to the Inference of Regulatory Netowkrs from Gene Expression Data. *Bioinformatics*, 21(14):3131–3137, 2005.

[81] J. Rung, T. Schlitt, A. Brazma, K. Freivalds, and J. Vilo. Building and Analysing Genome-Wide Gene Disruption Networks. *Bioinformatics*, 18(Suppl.2):S202–S210, 2002.

[82] R. Sanguesa and U. Cortes. Learning Causal Networks from Data: A Survey and A New Algorithm for Recovering Possibilistic Causal Networks. *AI Communications*, 10(1):31–61, 1997.

[83] E. Segal, M.Shapira, A. Regev, D. Pe'er, D. Botstein, and D. Koller. Module Networks: Identifying Regulatory Modules and Their Condition-Specific Regulators from Gene Expression Data. *Nature Genetics*, 34(2):166–176, 2003.

[84] L. 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.

[85] I. Simon, J. Barnett, N. Hannett, C.T. Harbison, N.J. Rinaldi, T.L. Volkert, J.J. Wyrick, J. Zeitlinger, D.K. Gifford, T.S. Jaakkola, and R.A. Young. Se-

rial Regulation of Transcriptional Regulators in the Yeast Cell Cycle. *Cell*, 106(6):697–708, 2001.

[86] K. Sivakumar, R. Chen, and H. Kargupta. Learning Bayesian Network Structure from Distributed Data. In *SIAM International Data Mining Conference*, pages 284–288, 2003.

[87] P. Smolen, D.A. Baxter, and J.H. Byrne. Modeling Transcriptional Control in Gene Networks – Methods, Recent Results and Future Directions. *Bulletin of Mathematical Biology*, 62:247–292, 2000.

[88] E.P.V. Someren, L.F.A. Wessels, and M.J.T. Reinders. Linear Modeling of Genetic Networks from Experimental Data. In *International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 355–366, 2000.

[89] R. Somogyi and C.A. Sniegoski. Modeling The Complexity of Genetic Network: Understanding Multigene and Pleiotropic Regulation. *Commplexity*, 1(6):45–63, 1996.

[90] P.T. Spellman, G. Sherlock, and B. Futcher. Comprehensive Identification of Cell Cycle-Regulated Genes of the Yeast Saccharomyces Cerevisiae by Microarray Hybridization. *Molecular Biology of the Cell*, 9:3273–3297, 1998.

[91] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. The MIT Press, 2nd edition, 2001.

[92] M. Sugita. Functional Analysis of Chemical Systems *in vivo* Using a Logical Circuit Equivalent. *Journal of Theoretical Biology*, 1:415–430, 1961.

[93] M. Sugita. Functional Analysis of Chemical Systems *in vivo* Using a Logical Circuit Equivalent: II. The Idea of a Molecular Automaton. *Journal of Theoretical Biology*, 4:179–172, 1963.

[94] S. Miyano SY. Kim, S. Imoto. Inferring Gene Networks from Time Series Microarray Data Using Dynamic Bayesian Networks. *Brief Bioinform*, 4(3):228–35, 2003.

[95] Y. Tamada, S.Y. Kim, H.D Bannai, S. Imoto, K. Tashiro, S. Kuhara, and S. Miyano. Estimating Gene Networks from Gene Expression Data by Combining Bayesian Network Model with Promoter Element Detection. *Bioinformatics*, 19(Supl. 2):ii227–ii236, 2003.

[96] D. Thieffry and R. Thomas. Qualitative Analysis of Gene Networks. In *Pacific Symposium on Biocomputing (PSB)*, volume 3, pages 77–88, 1998.

[97] H. Toh and K. Horimoto. Inference of A Genetic Network by A Combined Approach of Cluster Analysis and Graphical Gaussian Modeling. *Bioinformatics*, 18(2):287–297, 2002.

[98] T.M. Townes and R.R. Behringer. Human Globin Locus Activation Region(LAR): Role in Temporal Control. *Trends Genet.*, 6(7):219–23, 1990.

[99] A. Tucker, X. Liu, and A. Ogden-Swift. Evolutionary Learning of Dynamic Probabilistic Models With Large Time Lags. *International journal of intelligent system*, 16(5):621–645, 2001.

[100] M. Wahde and J. Hertz. Course-Grained Reverse Engineering of Genetic Regulatory Networks. *Biosystems*, 55:129–136, 2000.

[101] C. Walter, R. Parker, and M. Yčas. A Model for Binary Logic in Biochemical Systems. *Journal of Theoretical Biology*, 15:208–217, 1967.

[102] S. Watanabe, Y. Maki, Y. Eguchi, D. Tominaga, and M. Okamoto. Algorithms for Inference of Genetic Networks AIGNET. In *Intl. Workshop on Genome Informatics*, pages 274–275, 1998.

[103] D.C. Weaver, C.T. Workman, and G.D. Stromo. Modeling Regulatory Networks with Weight Matrices. *Pacific Symposium on Biocomputing (PSB)*, pages 112–123, 1999.

[104] C.C. Wu, J.C. Huang, H.F. Juan, and S.T. Chen. GeneNetwork: An Interactive Tool for Reconstruction of Genetic Networks Using Microarray Data. *Bioinformatics*, 20(18):3691–3693, 2004.

[105] J.J. Wyrick and R.A. Young. Deciphering Gene Expression Regulatory Network. *Current Opinion in Genetics and Development*, 12:130–136, 2002.

[106] J. Yu, V.A. Smith, P.P. Wang, A.J. Haremink, and E.D. Jarvis. Advances to Bayesian Network Inference for Generating Causal Networks from Observational Biological Data. *Bioinformatics*, 20(18):3594–3603, 2004.

[107] C.H. Yuh, H. Bolouri, and E.H. Davidson. Genomic Cis-Regulatory Logic: Experimental and Computational Analysis of A Sea Urchin Gene. *Science*, 279:1896–1902, 1998.

[108] X.B. Zhou, X.D. 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.

[109] M. Zou and S.D. Conzen. A New Dynamic Bayesian Network (DBN) Approach for Identifying Gene Regulatory Networks from Time course Microarray Data. *Bioinformatics*, 21(1):71–79, 2005.