

University of Windsor

Scholarship at UWindor

Electronic Theses and Dissertations

Theses, Dissertations, and Major Papers

1-1-2006

Modeling multiple time units delayed gene regulatory network using dynamic Bayesian network.

Zhengzheng Xing
University of Windsor

Follow this and additional works at: <https://scholar.uwindsor.ca/etd>

Recommended Citation

Xing, Zhengzheng, "Modeling multiple time units delayed gene regulatory network using dynamic Bayesian network." (2006). *Electronic Theses and Dissertations*. 7147.
<https://scholar.uwindsor.ca/etd/7147>

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.

Modeling Multiple Time Units Delayed Gene Regulatory Network Using Dynamic Bayesian Network

by

Zhengzheng Xing

A Thesis

Submitted to the Faculty of Graduate Studies and Research
through Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada
2006

©2006 Zhengzheng Xing



Library and
Archives Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-42338-7
Our file Notre référence
ISBN: 978-0-494-42338-7

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.


Canada

Abstract

Time delay is an important biological feature of gene regulation, and it is widely observed by biological experiments. Most of the current applications which use dynamic Bayesian network to model gene regulatory network assume that the time delay between regulators and their targets is one time unit in a time series gene expression dataset. In fact, multiple time units delay is indicated to exist in the gene regulation process. In this thesis, a method of using higher-order Markov dynamic Bayesian network (HMDBN) to model multiple time units delayed gene regulatory network is proposed. A learning framework using mutual information and genetic algorithm is designed to learn the structure of a HMDBN from time series gene expression data. When applied to real-world yeast cell cycle gene expression datasets, the predicted gene regulatory networks are strongly supported by biological evidence and consistent with the yeast cell cycle phase information.

Acknowledgments

The first person I would like to thank is my supervisor, Dr. Dan Wu. He brought me to the research area of bioinformatics and taught me the way of doing research. When I was frustrated by proposing my own idea about the thesis, his insightful discussions and trust help me out of the hardest time. Furthermore, he always encourages me to continue my research in the future. Dr. Dan Wu is an excellent supervisor and a good friend to me. I feel fortunate to be his student and finish the thesis under his supervision.

I would like to express my gratitude to my thesis committee members, Dr. Alioune Ngom, Dr. Andrew Hubberstey, and Dr. Robin Gras. Dr. Alioune Ngom's excellent courses build me a solid knowledge base in machine learning and computational molecular biology. I was always inspired by his comments during doing the thesis. Dr. Andrew Hubberstey provided me many constructive suggestions on the thesis from a biological point of view. Dr. Robin Gras offered his precious time to be the chair.

Finally, I am very grateful for my family whose love and patience enabled me to complete this work. In addition, I would like to thank all my colleagues who help me during my graduate studies.

Contents

Abstract	iii
Acknowledgments	iv
List of Figures	vii
List of Tables	viii
1 Introduction	1
1.1 Bioinformatics	1
1.2 Gene Regulatory Networks	2
1.3 Microarray and Gene Expression Data	6
1.4 Gene Expression Analysis	10
1.4.1 The First Level	10
1.4.2 The Second Level	11
1.4.3 The Third Level	11
1.5 Motivation	16
1.6 Outline	17
2 Bayesian Networks and Dynamic Bayesian Networks	18
2.1 Bayesian Networks	18
2.2 Structure Learning of BNs	21
2.3 First-order Markov Dynamic Bayesian Networks	24
2.4 Structure Learning of First-order Markov DBNs	25
3 Learning HMDBNs From Gene Expression Data	28
3.1 Higher-order Markov Dynamic Bayesian Networks	28
3.2 A Two Steps Learning Framework	31
3.2.1 Pre-processing: Discretization	31

3.2.2	Step 1: Mutual Information Matrix	33
3.2.3	Step 2: Genetic Algorithm	37
3.3	The Problem of Varied Size of Training Data	42
4	Experiments	47
4.1	A Nine-Gene Network	47
4.1.1	Dataset	47
4.1.2	Results	48
4.1.3	Parameter Selection	56
4.2	A Fourteen-Gene Network	59
4.2.1	Dataset	59
4.2.2	Result	59
5	Conclusion and Future Work	63
5.1	Contributions	63
5.2	Future Work	64
	Bibliography	65
	Appendix A Source Code	71
1.	K-mean Discretization	71
2.	Score Functions.	74
4.	Mutual Information Matrix.	76
4.	Genetic Algorithm	79
	VITA AUCTORIS	92

List of Figures

1.1	The process of gene expression.	4
1.2	DNA microarray.	6
1.3	Collecting gene expression data in cDNA micorarray experiments.	8
2.1	An example of a 5-node Bayesian network.	20
2.2	An example of a first-order Markov DBN.	25
2.3	An example of dataset shifting for learning the structure of first-order Markov DBNs.	26
3.1	An example of a HMDBN.	29
3.2	discrete profiles of six yeast genes using k-mean algorithm.	34
3.3	Mutual information matrix for a 2-node 3 rd -order Markov DBN.	36
3.4	Crossover for 3-node 3 rd -order Markov DBNs.	40
3.5	Knowledge guided mutation for a 2-node 3 rd -order Markov DBN.	42
3.6	Random mutation a 2-node 3 rd -order Markov DBN.	42
3.7	An example of training data size variation for different HMDBN structures.	43
3.8	An example of using dataset as a loop.	45
4.1	Gene expression profiles and yeast cell cycle phase information of Chou's dataset.	48
4.2	Fitness function converging plots for the 9-gene network.	50
4.3	Results Comparison of the 9-gene network using ML score	52
4.4	Results Comparison of the 9-gene network using MDL score.	57
4.5	Learning results of $r > 5$	58
4.6	Fitness function converging plots for the 14-gene network.	60
4.7	Results Comparison of the 14-gene network.	62

List of Tables

1.1	Gene expression dataset	9
3.1	Representation of a network structure in genetic algorithm . . .	39
4.1	Experiment Parameters.	49
4.2	Predicted regulation pairs in Figure 4.3(b)	51
4.3	Predicted regulation pairs in Figure 4.3(b)	55

Chapter 1

Introduction

1.1 Bioinformatics

With the completion of sequencing the human genome, the post-genome era comes. Instead of concentrating on sequencing the genome in a pre-genome era, now the challenge is to develop efficient methods to harvest the fruits hidden in the large amount of genomic data. The challenge inspired an emerging research field, *bioinformatics*, to appear. Bioinformatics involves biology, computer science, mathematics, and statistics to analyze genomic data, and to solve biological problems usually on the molecular level [53]. Besides the sequence data, many new experimental data produced by modern technologies, such as gene expression microarray, genetic manipulation of genes in cells and organisms, are being assembled. The abundance of data offers researchers a great opportunity to understand the secret of diseases, evolution, biological variations, *etc.* The field of bioinformatics supports a broad spectrum of research which includes determining the biological significance of the data, providing the expertise to organize it, and developing practical computational tools to mine the large volume and noisy data for new information [53].

Currently, the topics in bioinformatics include sequence analysis, computational evolutionary biology, measuring biodiversity, gene expression analysis, regulation analysis, protein expression analysis, analysis of mutations in cancer, structure prediction, comparative genomics, modeling biological systems, and high-throughput image analysis [52]. This thesis falls into the areas of gene expression analysis and regulation analysis.

1.2 Gene Regulatory Networks

Deoxyribonucleic acid (DNA) is a nucleic acid that contains the genetic instructions specifying the biological development of all cellular forms of life, and most viruses [1]. A DNA molecule is composed of two strands in the form of a double helix structure. Each strand is the sequence of combination of four *nucleotides*, Adenine (A), Thymine (T), Guanine (G) and Cytosine (C). A is complementary to T and G is complementary to C, which is also called Watson-Crick rule [26]. Each nucleotide along one strand of the DNA is matched with its complimentary nucleotide in the opposite position on the other strand .

Genetic information in DNA is coded in the sequences of nucleotides. The sequence can be viewed as an instruction book. Using the instruction book, various *proteins* can be generated. *Protein* is a molecular comprising a long chain of *amino acids*, which folds into a three-dimensional structure unique to a particular protein that has certain biological activities [53]. Every triplet of nucleotides in a ribo nucleic acid sequence is called a *codon*. Because there are four kinds of nucleotides, A, C, U, G, the number of possible codons is

$4 * 4 * 4 = 64$. Each codon specifies a single amino acid, while a single amino acid may correspond to several different codons. For example, the amino acid Lys can be represented by codon AAA or codon AAG. Totally, there are only 20 different amino acids to compose proteins. The set of rules which map a tri-nucleotide sequences (codon) to an amino acid is also known as the *genetic code*. Proteins are responsible for catalyzing most intracellular chemical reactions (enzymes), for regulating gene expressions (regulatory proteins), and for determining many features of the structures of cell, tissue, and virus (structural protein) [27].

A *gene* is a segment of DNA that specifies a unit of biological functions and usually corresponds to a protein [53]. In many species, only a small fraction of DNA appears to be the gene area. The other area of the DNA which has not been understood to contain genes or have a function is assumed to be the junk DNA. Genes contain the sequence that determine the amino acid sequence of proteins and the surrounding sequences that controls when and where the protein will be produced [52].

RNA is usually a single-stranded molecule, and similar to DNA, is composed by four nucleotides A, C, G, U. In RNAs, nucleotide U replaces nucleotide T. The complementary rule between nucleotides applies to both RNA and DNA as A paired with T/U and G paired with C. RNA serves as the template in the process of converting genes to proteins.

The process of converting the genetic information from DNA into proteins is called *gene expression*. Gene expression is accomplished by a series of events. The principle steps in gene expression are *transcription* and *transla-*

tion. In the transcription step, a segment of DNA is copied into a *messenger RNA (mRNA)*. The translation step in gene expression is the synthesis of proteins from mRNAs. The genetic information flows from DNA to RNA to proteins, as shown in Figure 1.1, is also known as the *central dogma of molecular biology*. During the process of gene expression, the amount of mRNA reflects the degree of genes' expression. Microarray gene expression data is obtained by utilizing mRNAs, which is explained in Section 1.3.

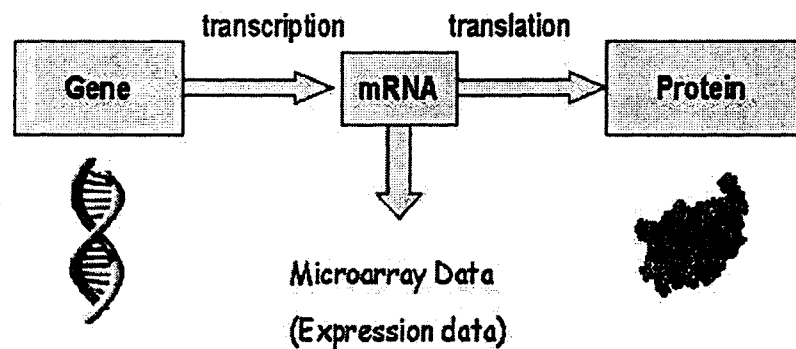


Figure 1.1: The process of gene expression.

Among all the genes in a cell, not all of them are active continuously. In a particular cell of an organism, at a certain time and under a specific condition, only a subset of genes are expressed at a high level. The levels of gene expression may differ from one cell type to another or according to the stages in the cell cycle. Viewing DNA as the a complex program, then genes can be seen as functions in the program. The output of every gene function is a protein. Inside the program, we need a mechanism to call various gene functions under different conditions. In general, the mechanism that control the expression of particular genes in response to external or internal signals is called *gene regulation* [27]. Any steps in the process of gene expression may be regulated, from transcription to post-translational modification of a protein.

Many genes in eukaryotes are regulated at the level of transcription. Although both negative and positive regulations occur, positive regulation is typical [27].

In transcriptional regulation, a special protein, called *transcription factor*, binds itself to the promoter region of a gene to activate its expression. Because transcription factors themselves are also the expression products of some genes, so, they can affect the expression of other genes through their protein products. The binding site of a gene can be recognized by multiple transcription factors, and the expression level of a gene is determined by a combination of these factors that bind to the binding site [32]. The regulatory interactions among genes and proteins form a complex network, called *gene regulatory network* [26].

Gene regulatory network *dynamically* regulates the level of expression for each gene and often includes dynamic feedbacks. Malfunction of gene regulatory network is a major cause of human diseases. For example, more than 50 transcription factors have now been identified to be related to human cancer [32]. Although gene regulatory network is valuable to us, it is still unknown. A central goal of molecular biology is to understand the regulatory mechanisms and the synthesis of proteins [19]. Several approaches have been explored to construct gene regulatory network from available experimental data. Microarray gene expression data is a major data source that is used in reverse engineering of gene regulatory network.

1.3 Microarray and Gene Expression Data

A *DNA microarray* (also commonly known as gene chip, DNA chip, or gene array) is a collection of microscopic DNA spots attached to a solid surface, such as glass, plastic or silicon chip forming an array for the purpose of expression profiling, monitoring expression levels for thousands of genes simultaneously [52].

Figure 1.2 is an example of DNA microarray. The spots on the chip are arranged in a regular pattern usually forming a rectangular array. Usually, each spot corresponds to a particular gene, and the specific location of the spot is used as the identity of a gene. *Hybridization* is the fundamental basis of DNA microarray [26]. If two DNA stands are complementary to each other, they will hybridize to form a double strands union. Hybridization will still occur when one or both strands of the DNA are replaced by RNA as long as they are complementary. Every spot contains many copies of single strand DNAs (DNA segment) of a particular gene. In microarray experiments, the spot is used to detect the amount of mRNAs which hybridize with the single strand DNAs, and the spot is also called *probe*.

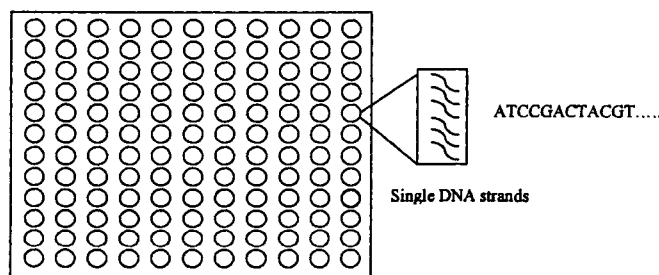


Figure 1.2: DNA microarray.

There are mainly two kinds of microarray, oligonucleotide microarray (single-channel microarray) and cDNA microarray (two-channel microarray) [53]. The basic ideas of the two kinds of arrays are similar. The difference is the way in which DNA fragments representing the genes are attached to the array.

Oligonucleotides are short sequences of nucleotides (RNA or DNA), typically with twenty or fewer bases. In oligonucleotide microarrays, the probes are oligonucleotides, and it is designed to match parts of the sequence of known or predicted mRNAs. Several companies such as Affymetrix, Healthcare, and Agilent provide commercial oligonucleotide microarrays that cover complete genomes. These microarrays give estimations of the absolute value of gene expression. If we have two samples from different conditions, one control sample and one experimental sample, two separate microarrays need to be used. Therefore, Oligonucleotides array is also called single-channel microarray.

Complementary DNA (cDNA) microarrays are similar to the oligonucleotide arrays. Instead of using short oligonucleotides as probes, each spot contains a cDNA segments clone from a known gene, usually of hundreds of bases. cDNA is a single stranded DNA synthesized from a mature mRNA template which only has exons of genes [53]. cDNA microarray allows multiple experimental samples, such as control sample and experimental sample, to hybridize at the same time, as long as different dyes are used. Therefore cDNA microarray is also called two-channel microarray.

Figure 1.3 depicts the process of using cDNA microarray to collect gene expression data in a series of experiments. In a single experiment, experimental sample and control sample are prepared, and mRNA in the samples

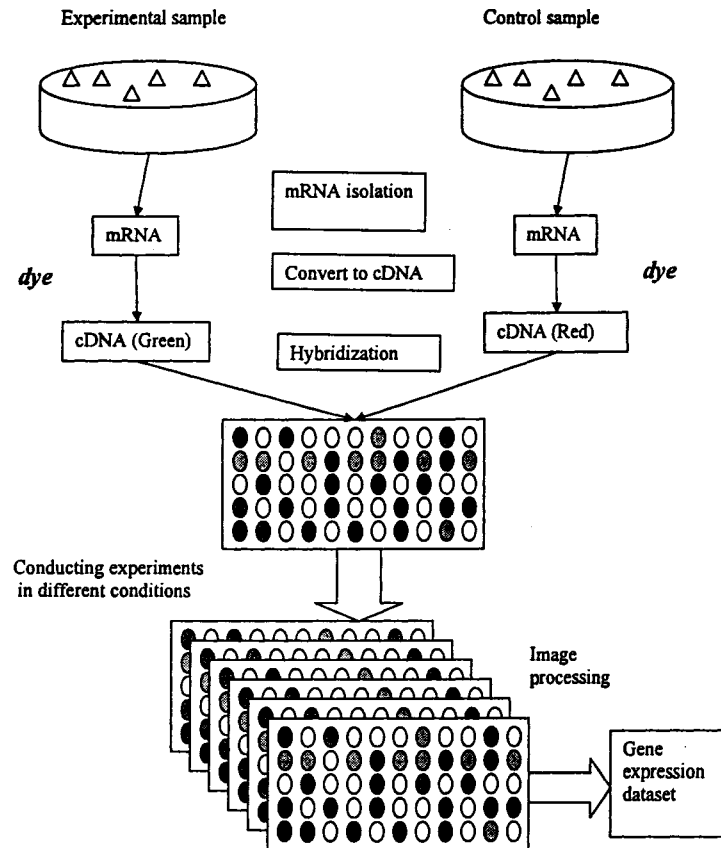


Figure 1.3: Collecting gene expression data in cDNA micorarray experiments.

are isolated first. The concentration of a particular mRNA in the sample is a result of the *expression* of its corresponding gene. So, the amount of a particular mRNA is a measure of the expression level of its corresponding gene. Then, mRNA in the sample is transformed into its cDNA because cDNA is more stable than mRNA. Experimental and control samples are labeled with different fluorescent dyes. At last, the two samples are put onto the surface of a cDNA microarray to let the cDNA in the samples to hybridize with the probes on the array. Because the microarray chips contains many DNA probes, when a experimental sample is applied to a chip, different probes will match all mRNAs in the sample. After hybridization, different spots appear in dif-

UID	NAME	10 min	30 min	50 min	70 min	80 min	90 min
YAL001C	TFC3	-1.04	-0.173	-0.257	-0.617	0.0865	-0.979
YAL002W	VPS8				-0.414	0.508	-0.103
YAL003W	EFB1	-0.185	-0.039	0.125	0.08	0.655	-0.277
YAL005C	SSA1	-1.735	-2.676	-2.865	-3.058	-1.729	-3.262
YAL007C	ERP2				0.074	0.61	0.142
YAL008W	FUN14	0.394	0.35	0.442	-0.38	0.788	-0.605
YAL009W	SPO7		-0.419	0.291	-0.161	0.259	-0.683
YAL010C	MDM10		1.585		-0.406	-0.048	-0.023
YAL012W	CYS3	-0.289	-0.717	-0.292	0.034	0.879	-0.434

Table 1.1: Gene expression dataset

ferent colors, and the intensity of each spot is related to the expression level of the particular gene corresponding to that spot. After an image processing step, the colors are transformed into gene expression values. Usually, a series of experiments are conducted under several different experimental conditions, such as temperatures, times, outside stimulations. A series of experimental data is organized into a *gene expression dataset*.

Currently, many gene expression dataset, such as yeast cell cycle gene expression dataset [6, 47], human cell cycle gene expression dataset [51], are published and can be accessed freely on the Internet. Table 1.1 is an segment of the gene expression dataset of Spellman *et al.* [47]. The first two columns in the table are the IDs and names of genes. From the third column, every column represents the expression levels of all the genes measured at a specific time in the yeast cell cycle. Every row in the table is the gene expression profile of a single gene collected in a time series. The blank cells in the table are missing values.

1.4 Gene Expression Analysis

Gene expression data provides a snap shot of the gene expression levels of a large amount of genes in genome scope and under many different experimental conditions. It provides us a great opportunity to interpret the underlying biological mechanisms that control the expression of genes. Analyzing gene expression dataset has become a complete new research area in bioinformatics. In this section, a literature review on gene expression analysis is provided.

Gene expression data has been analyzed on at least three levels of increasing complexity. First, the level of single genes, where one intends to establish whether each gene in isolation behaves differently in a control versus a treatment situation. The second level considers gene combinations, where clusters of genes are analyzed in terms of common functionalities, interactions, co-regulation, and so forth. The third level is to uncover the gene regulatory network, or gene regulation pathway [2]. It is the ultimate goal of gene expression data analysis.

1.4.1 The First Level

In the first level, a mathematical description of the biophysical processes in terms of a system of coupled differential equations is provided to describe the processes of transcription factor binding, diffusion, protein and RNA degradation [4]. Constructing differential equations requires detailed understanding of the interaction agents as well as the parameters of the biochemical reaction. Therefore, this approach is restricted to describing very small systems [19]. In Zak *et al* [54], it was mentioned that even for a system of only 3 genes, the

differential equations are not identifiable when only gene expression data is observed.

1.4.2 The Second Level

The most popular method in the second level is clustering. The basic idea of clustering is to group genes into clusters based on the similarities among their gene expression patterns over different experimental conditions. Genes in the same cluster are co-expressed. In clustering analysis, it is assumed that co-expressed genes are co-regulated. Thus, genes in the same cluster may have similar functions or are involved in related biological processes. Defining the meaning of similarity is an important issue in all clustering methods. Eisen *et al.* [11] first proposed to apply clustering to gene expression data analysis. Following that, a lot of clustering methods have been designed to group gene expression data in different ways. D'Haeseleer *et al.* [8] provided a good survey of clustering in gene expression analysis. Although clustering provided us a fast cheap way to extract useful information from a large scale gene expression dataset, it did not lead to a fine solution of the interaction process. It only indicates which genes are co-regulated, but in a cluster, whether a gene is the regulator or the regulatee can not be distinguished [19].

1.4.3 The Third Level

In the third level, it aims to address the ultimate goal of gene expression data analysis, constructing the *gene regulatory network*, or *gene regulation pathway*, from the data. The problem of reverse engineering of gene regulatory networks

from gene expression data can be attacked using two different forms of data: time series data and steady-state data of gene knockouts [26]. The method proposed in this thesis belongs to constructing gene regulatory network from time series gene expression data.

Several machine learning (data mining) methods have been explored to mine gene regulatory network from gene expression data. *Relevance network* [3] is constructed by computing comprehensive pair-wise mutual information, and then adding undirected edges between gene pairs with a high mutual information above a threshold. This method can only construct a network structure without directions. *Boolean network* was first applied to model gene regulatory network by Liang *et al.* [30]. In a boolean network model, all factors in a genetic regulatory network are represented by boolean variables, which can only take on two possible values, on and off; all relationships between variables are required to be logical. These restrictions limit the ability of boolean networks to handle noise data, and to model a gene regulatory network which can not be represented as exactly logic functions but rather an inherently stochastic process. Other methods, such as *decision tree* [29], *neural network* [49], have also been applied. In particular, Bayesian network and dynamic Bayesian network have been proved to be useful tools to model gene regulatory networks.

Friedman *et al.* [12] was credited with first proposing and using Bayesian network (BN) to gene expression analysis. When learning a BN structure from the gene expression data, they treat each measurement as a sample from a distribution and do not take into account the temporal aspect of the measurement. A heuristic learning algorithm is designed to estimate a network

structure from limited gene expression data efficiently without using any prior knowledge. When looking for results, they do not use a single best full structure, but use a bootstrap method to attract the common low dimension network structures from a set of full network structures which all fit the data. Applied to a yeast time series gene expression dataset containing 800 genes in 76 measurements, it shows some success in extracting central regulatory pathways in yeast. This exploration demonstrates the potential of applying BNs to model gene regulatory networks and also exposes some disadvantages. One of the important problems is that although delicate learning algorithm and bootstrap method are used, the accuracy of predicated relationships is still very low [38]. It indicated that the method is not suitable to be applied to large amount of genes where the predication will become extremely unreliable.

Pe'er *et al.* [39] extended the framework of Friedman *et al.* [12] to learn a Bayesian network from perturbed gene expression data. It shows the ability of BN to handle both time series data and steady-state data with gene mutation and deletion.

Segal *et al.* [43] proposed a module network procedure, a method based on Bayesian network for inferring regulatory modules from gene expression data. This model is a mixture of Bayesian network and clustering. It is capable of handling large amount of genes as clustering, and at the same time it can model a fine structure of gene regulation as Bayesian network.

Unlike Friedman *et al.*, who only use gene expression data, some researchers explored the possibility to mine knowledge from multiple data sources using Bayesian networks. Hartemink *et al.* [16] constructed the BNs not only from

the gene expression data but also combining with the location data to enhance the accuracy of the predicted network structure. Tamada *et al.* [48] developed a method to integrate microarray gene expression data and DNA sequence information into a Bayesian network model. Imoto *et al.* [22] proposed a method to add protein-protein interactions, protein-DNA interactions, binding site information, and existing literature knowledge into BNs.

Usually, gene expression data are discretized into two or three categories before learning a Bayesian network from the data. Instead of using discretized variables, a method of using Bayesian network and nonparametric regression to handle continuous variables was proposed [21].

Although these applications of Bayesian networks can provide some useful biological information, the dynamic nature of gene regulatory networks was ignored. Dynamic Bayesian network (DBN), an extension of (static) Bayesian network to model temporal processes, is more suitable to represent gene regulatory networks [14]. Furthermore, DBN is not restricted to be an acyclic graph as BN, which makes it possible to model an important property of gene regulatory network, the regulatory feedback.

DBN was first introduced to model gene regulatory network by Murphy *et al.* [35] and Friedman *et al.* [14]. In Murphy *et al.* [35], extracting the dynamic interactions among genes from gene expression data was discussed from a very theoretical point of view. In Friedman *et al.* [14], a simplified dynamic Bayesian network was defined and a learning framework for DBNs was proposed. Following that, Ong *et al.* [37] applied DBN to analyze the regulatory pathways in E.Coli. Perrin *et al.* [40] proposed a method to handle missing

data and continuous variables in dynamic Bayesian network and applied it to extract gene regulations in DNA repair network of the E.coli [40]. Kim *et al.* [24] combined DBN with a non-parametric regression method to construct nonlinear regulatory relationships from continuous data and tested the method on a yeast cell cycle gene expression dataset [47]. Zou *et al.* [55] proposed a new method to increase the accuracy of prediction of DBN through estimating the accurate transcriptional time lag.

Although many methods of using BNs or DBNs to model gene regulatory networks have been developed, evaluating their effectiveness has not been well studied. Currently, comparing the predicted regulatory network with known biological knowledge from biological experiments is a widely used method. This evaluation method lacks of formal standards because of the insufficiency of current biological knowledge. Some researchers suggested to validate the methods through simulation studies. Smith *et al.* [46] designed a simulator to generate data representing a complex biological system, and then test various Bayesian network learning algorithms on the simulated datasets to evaluate the effectiveness. Husmeier *et al.* [19] proposed a method of using dynamic Bayesian network and Bayesian learning with Markov chain Monte Carlo to infer networks from simulated data. In the simulation studies, the issues about how the network learning performance varies with the size of training data, the degree of inadequacy of prior assumptions, the experimental sampling strategy and the inclusion of further, sequence-based information are discussed.

1.5 Motivation

In the dynamic process of gene regulation, time delay is an important feature and is observed by many biological experiments [9]. Time delay could exist in various situations in a gene regulation process. One of the possibilities is that one gene is transcribed and translated into a protein, and it then activates or inhibits its target gene to express. This implies that the expression dependence between the two genes is not simultaneous, but with a time delay. Dynamic Bayesian network [15] can model the situation that one event causes another event in the future, which makes it suitable to model time delayed gene regulatory networks. Although there are some applications of using DBN in gene regulation analysis [25, 35, 37, 40], in their DBNs, the time delay between all the regulators and their targets is assumed to be one time unit of a time series gene expression dataset. Although it was indicated that multiple time units delay in gene regulation is possible, the computational cost for incorporating this information into the learning process of DBN is expensive [37]. Zou et al. [55] proposed an approach to handle multiple time units delay using DBN, however, the structure of the learned DBN ignored the scenario that several regulators co-regulate one gene with different time lags. Besides DBN, other methods, such as decision tree [29], clustering [41] have also been tried to capture multiple time units delay in gene regulatory networks.

In this thesis, we propose using higher-order Markov DBN (HMDBN) to model multiple time units delayed gene regulatory networks. A two step heuristic learning framework is designed. First, a mutual information matrix is computed to select gene pairs with a particular time lag as candidate pairs. Then, genetic algorithm is applied to search for an optimal network structure using

mutual information matrix as prior knowledge. Genetic algorithm interacts with mutual information matrix through population initialization and genetic operators. A particular problem in learning the structure of a HMDBN is that the size of applicable training data varies when scoring different network structures. One solution to this problem is suggested in this thesis. In order to verify the effectiveness of the proposed method, we apply the learning framework to real-life gene expression datasets and compare the results with biological evidence.

1.6 Outline

This thesis is organized as follows. In Chapter 2, the background of Bayesian network and dynamic Bayesian network is introduced. In Chapter 3, a new type of DBN, higher-order Markov DBN is proposed, and the learning framework for it is proposed. The experiments results on *Saccharomyces cerevisiae* gene expression datasets are analyzed in Chapter 4. In Chapter 5, the conclusion and future work are presented.

Chapter 2

Bayesian Networks and Dynamic Bayesian Networks

2.1 Bayesian Networks

Bayesian networks are interpretable and flexible models for representing probabilistic relationships between multiple interacting variables [20]. A Bayesian network is composed of two components, a graphical component (the structure of a BN) and a numerical component (the parameters of a BN). At a qualitative level, the structure of a BN describes the relationships between variables in the form of conditional independence relations. At the quantitative level, relationships between variables are described by conditional probability distributions [20]. The two components define a joint probability distributions over a set of random variables together. In this thesis, we use capital letters for variable names and boldface capital letters for sets of variables. Formally, considering a set $\mathbf{X} = \{X_1, \dots, X_n\}$ of random variables, a *Bayesian network* (BN) ¹ is a graphical representation of a *joint probability distribution* of \mathbf{X} [12].

¹When we say Bayesian networks, it means static Bayesian networks.

The graphical structure G of a BN consists of a set of *nodes* and a set of directed *edges*. The nodes represent random variables. The edges indicated conditional dependence between variables. If there is a directed edge from node A to node B , then A is called the *parent* B , and B is called the *child* of A . Figure 2.1 is an example of the structure of a five-node Bayesian Network. In Figure 2.1, we have a set of nodes $V = \{A, B, C, D, E\}$, and a set of edges $E = \{(E, B), (A, B), (A, D), (B, C)\}$. Node E and node A do not have any parents. Node E and A are the parents of node B . Node D is the child of node A and node C is the child of node B . The graphical structure of a BN has to be a *directed acyclic graph* or *DAG*, which is defined by the absence of directed cycles.

The graph G encodes the markov assumption: each variable X_i is independent of its non-descendant, given its parents in G . This assumption enables a BN to represent a joint probability distribution of \mathbf{X} in a more compact way. Let $\mathbf{X} = \{X_1, \dots, X_n\}$ be a set of random variables represented by the nodes $i = 1, \dots, n$ in a BN, and let $\text{PA}(i)$ to be the set of parents of variable X_i , then,

$$P(X_1, X_2, \dots, X_n) = \prod_{i=1}^n P(X_i | \text{PA}(i)). \quad (2.1)$$

In this way, instead of using a space in the order of $O(2^n)$ to present the joint probability, only a space in the order of $O(2^k)$ is require to represent a Bayesian network, where k is the maximum number of parents of a variable [34]. Take Figure 2.1 as an example, according to the markov assumption, variable C is independent of its non-descendant variables A, E, D given its parent, variable B , and it is represented as $I(C; A, D, E | B)$. Similarly, we have $I(A; E)$,

$I(B; D|A, E)$, $I(D; B, C, E|A)$, and $I(E; A, D)$. The network structure represent the joint probability in the form of a production,

$$P(A, B, C, D, E) = P(A)P(B|A, E)P(C|B)P(D|A)P(E). \quad (2.2)$$

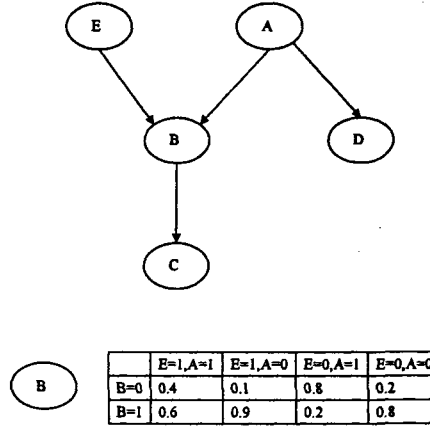


Figure 2.1: An example of a 5-node Bayesian network.

The variables in BNs can be continuous or discrete. For discrete variables, the parameters of BNs are a set of *conditional probability tables*. Every node is associated with a conditional probability table given its parents. In Figure 2.1, every variable can only take boolean values, and the table in the bottom of the figure is the conditional probability table of variable B . In the table, the probabilities of all possible assignments of $P(B = b|E = e, A = a)$, where $b, e, a = 0, 1$ are shown. For continuous variables, the parameters are a set of *conditional probability distribution* for every variable. The set of local conditional probability tables or distributions for all the variables, together with the set of conditional dependence assumptions described by the structure of the BN, define a full joint probability distribution for the network.

When applying BNs to model gene regulatory networks, we associate nodes

with genes and the values of the nodes with the expression values of genes, while the directed edges indicates interactions between the genes. For instance, the network structure of Figure 2.1 suggests that gene E and gene A co-regulate gene B , that gene B mediates the interaction between gene E , A and gene C ; that gene A regulate gene D . With the parameters of a BN, different type of regulations, such as inhibition or activation can be differentiated.

2.2 Structure Learning of BNs

Learning Bayesian networks from training data can be considered in several different settings. If the structure of a Bayesian network is known, the task is to learn the probability table for each variable from the training data, it is known as *parameters learning*; otherwise, the structure needs to be learned from the training data first, it is known as *structure learning*. Furthermore, sometime, not all the variables in a Bayesian network are observable, some are not presented in the training dataset or the dataset is not complete, it involves learning Bayesian network with hidden variables. Below, the basic idea of learning the structure of BNs from complete dataset is reviewed.

Learning the structure of a (static) Bayesian network from training dataset can be seen as a search problem to find an optimal structure in the search space that maximizes a score function. The score function measures the fitness of a given structure with respect to the dataset. The basic idea of learning the structure of a BN is to enumerate all the possible structures and choose the one with the highest score. However, the number of network structures increases super-exponentially with the number of nodes, and the optimization is a NP-hard problem [5]. If we have a training dataset of N variables, the number of

possible structures of the Bayesian network composed by N variables is given by the formula below,

$$f(n) = \sum_{i=1}^n (-1)^{i+1} C_n^i 2^{i(n-i)} f(n-i). \quad (2.3)$$

For $n=2$, the number of possible structure is 3; for $n=3$, it is 25; for $n=5$, it is 29,000; and for $n=10$, it is approximately $4.2 \cdot 10^{18}$ [7]. In order to solve the problem of large search space, heuristic search methods, such as hill-climbing, simulated annealing, have to be resorted to [17].

Among the existing different score functions, *Maximum likelihood* (ML) [14] is a widely used score. It is defined as below [14]:

$$ML = \sum_i \sum_{j_i} \sum_{k_i} N_{i,j_i,k_i} \log \frac{N_{i,j_i,k_i}}{\sum_{k_i} N_{i,j_i,k_i}}, \quad (2.4)$$

where N_{i,j_i,k_i} is the number of cases in the training dataset when node X_i takes value j_i and its parents take the values k_i . The higher the score for a structure, the better the structure is.

The problem of ML score is that it prefers complex structures because adding more parents to a node can not decrease the likelihood [42]. In order to find a sparse structure, one solution is to limit the maximal number of parents. Another solution is to use *Minimal description length* (MDL) score, which is proposed to penalize complex structures based on ML score [28]. The MDL score is defined by the following equations [28, 50],

$$MDL = DL_{data} + DL_{model}, \quad (2.5)$$

$$DL_{data} = -ML, \quad (2.6)$$

$$DL_{model} = \sum_{n_i \in n} [k_i \log_2(n) + d(s_i - 1) \prod_{j \in F_{n_i}} s_j], \quad (2.7)$$

where n is the number of nodes ; for node n_i , k_i is the number of its parents, F_{n_i} is its set of parents, s_i is the number of states it can be in, and s_j is the number of values a particular variable in F_{n_i} can take on; d is the number of bits needed to store a numerical value. MDL score is the sum of two parts. The first parts is DL_{data} , which is used to evaluate how the structure fits the training dataset. It is in fact the negative ML score. The second part is DL_{model} , which is used to evaluate the complexity of a structure. Because the structure with lower MDL is better, DL_{model} is a penalty for complex structure.

BIC score (Bayesian Information Criterion approximation) [36] is an variation of MDL score. It is defined as below,

$$BIC = ML - \frac{d}{2} \ln M, \quad (2.8)$$

where d is the number of bits needed to store a numerical value, and M measures the complexity of the network structure. It is designed for approximation for large amount of data. Compared to BIC score, MDL is more suitable for learning small training dataset.

2.3 First-order Markov Dynamic Bayesian Networks

Dynamic Bayesian network (DBN) is an extension of (static) Bayesian network to model temporal processes [14]. Assuming $\mathbf{X} = \{X_1, \dots, X_n\}$ is a set of attributes changing in a temporal process of T time slices, random variable $X_i[t]$ denotes the value of attribute X_i at time slice t , and $\mathbf{X}[t]$ denotes the set of variables $\{X_i[t] \mid 1 \leq i \leq n\}$, for $0 \leq t \leq T - 1$. A DBN represents the joint probability distribution over the variables $\mathbf{X}[0] \cup \mathbf{X}[1] \dots \cup \mathbf{X}[T - 1]$ [14]. Because the distribution of a DBN is extremely complex, in [14], a simplified DBN, called *first-order Markov DBN*, is proposed based on the following two assumptions:

1 First Order Markovian:

$$P(\mathbf{X}[t] \mid \mathbf{X}[t - 1], \mathbf{X}[t - 2], \dots, \mathbf{X}[0]) = P(\mathbf{X}[t] \mid \mathbf{X}[t - 1])$$

2 Stationary: $P(\mathbf{X}[t] \mid \mathbf{X}[t - 1])$ is independent of t .

First order Markovian assumption means that given variables in $\mathbf{X}[t - 1]$, variables in $\mathbf{X}[t]$ are independent of variables in $\mathbf{X}[t - 2] \cup \mathbf{X}[t - 3] \cup \dots \cup \mathbf{X}[0]$. Stationary assumption means that the dependence between $\mathbf{X}[t]$ and $\mathbf{X}[t - 1]$ is stationary. These two assumptions together suggest that in a first-order Markov DBN, arrows are only allowed to appear between adjacent time slices and the structure between two adjacent time slices remains the same as time evolves. Figure 2.2(a) is an example of a first-order Markov DBN, in which there are three nodes A , B , C evolving in T time slices. Every node in a particular time slice represents a variable. Arrows represent probabilistic dependence. In Figure 2.2(a), we can see that the structures between all the adjacent time

slices are same. Because of the reduplicate structures, usually, a *rolled representation* which only shows two time slices is used. The rolled representation of Figure 2.2(a) is shown in Figure 2.2(b). The structure between two time slices is called *inter-slice structure*. Applying the structure in Figure 2.2(b) to model a gene regulatory network, the arrows from node $A[t-1]$ to $C[t]$ and from $C[t-1]$ to $C[t]$ represent that gene C is co-regulated by gene A and itself with one time unit delay. It is interesting that DBN can model a regulation feedback that gene C is regulated by itself.

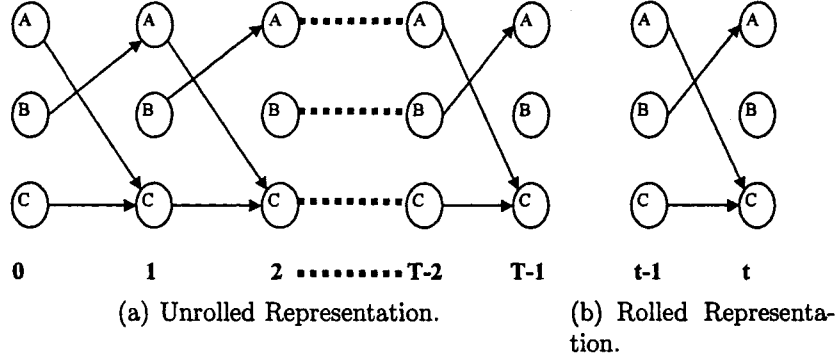


Figure 2.2: An example of a first-order Markov DBN.

2.4 Structure Learning of First-order Markov DBNs

Learning the structure of a dynamic Bayesian network shares the same idea with learning the structure of a (static) Bayesian network. Friedman [14] extended the score functions of BN to evaluate the structure of first-order Markov DBN. For the structure between two adjacent time slices, the same score functions, such as ML, MDL, for (static) Bayesian network can be used. The difference is that for dynamic Bayesian network, the training dataset needs

to be shifted. Given a time series training dataset, to compute the ML score for a inter-slice structures, N_{i,j_i,k_i} has to be counted from the *transition cases*. For the first-order Markov DBN, because the parents of a given node are all from previous adjacent time slice, the transition cases are obtained by shifting the data of the child one time unit back to align with the data of its parent [34].

	10 m	20 m	30 m	40 m	50 m	60 m	70 m	80 m	90 m	100 m
A	1	3	2	2	3	2	1	1	3	1
B	1	1	1	3	1	3	1	2	3	1
C	2	2	2	3	3	3	3	1	1	1

(a) A time series gene expression dataset.

	10 m	20 m	30 m	40 m	50 m	60 m	70 m	80 m	90 m
B	1	1	1	3	1	3	1	2	3
A	3	2	2	3	2	1	1	3	1
	20 m	30 m	40 m	50 m	60 m	70 m	80 m	90 m	100 m

(b) Shifted dataset for node A.

	10 m	20 m	30 m	40 m	50 m	60 m	70 m	80 m	90 m
A	1	3	2	2	3	2	1	1	3
C	2	2	2	3	3	3	3	1	1
C	2	2	3	3	3	3	1	1	1
	20 m	30 m	40 m	50 m	60 m	70 m	80 m	90 m	100 m

(c) Shifted dataset for node C.

Figure 2.3: An example of dataset shifting for learning the structure of first-order Markov DBNs.

In Figure 2.3, an example of shifting a training dataset to obtain transition cases is illustrated. Figure 2.3(a) is a gene expression dataset of 3 variables, gene A, B, C . This dataset contains the gene expression of 10 time slices in a time series from 10 to 100 minute with equal intervals. Given this dataset, computing the ML score for the structure of a first-order Markov DBN in Figure 2.2, the structures of every node given its parents need to be evaluated respectively according to the Equation 2.4. Node A has a single parent node

B. To obtain the transition cases where node *B* affects node *A* in 10 minutes later, as shown in Figure 2.3(b), the data of node *B* from 10 to 90 minute and the data of node *A* from 20 to 100 minute is used. Every column in the shifted dataset is a transition case. For node *C*, who has two parents, node *A* and node *C* itself from previous time slice, the shifted dataset is shown in Figure 2.3(c). A transition case is composed by the values of node *A* and *C* at the m minute and the value of node *C* at the $m + 10$ minute.

Chapter 3

Learning HMDBNs From Gene Expression Data

3.1 Higher-order Markov Dynamic Bayesian Networks

As suggested in [15], first-order Markov DBN can be extended to allow higher order interactions among variables. In a r^{th} -order Markov DBN, given a node $X_i[t]$, its parents can be chosen from the set of variables $X[t-r] \cup \dots \cup X[t-1]$ [15]. In this thesis, the two assumptions of first-order Markov DBN are extended for r^{th} -order Markov DBN as below:

1 r^{th} Order Markovian:

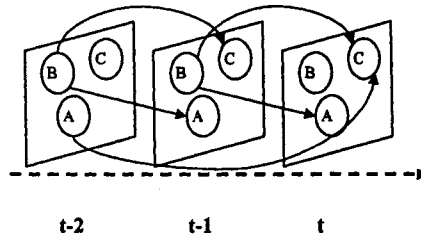
$$P(\mathbf{X}[t] | \mathbf{X}[t-1], \mathbf{X}[t-2], \dots, \mathbf{X}[0]) = P(\mathbf{X}[t] | \mathbf{X}[t-1], \dots, \mathbf{X}[t-r])$$

2 Stationary: $P(\mathbf{X}[t] | \mathbf{X}[t-1], \dots, \mathbf{X}[t-r])$ is independent of t .

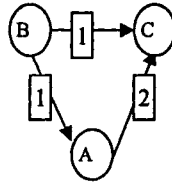
r^{th} order Markovian assumption means that given variables $\mathbf{X}[t-1], \dots, \mathbf{X}[t-r]$, $\mathbf{X}[t]$ are independent of all the variables in the time slices before time slice $t-r$.

Compared with First order Markovian, it can be seen that variables in time slice t are dependent on the variables in all r previous time slices instead of only depending on the variables in time slice $t - 1$. Stationary assumption means that the dependence between $\mathbf{X}[t]$ and $\mathbf{X}[t - 1], \dots, \mathbf{X}[t - r]$ remains the same when time evolves.

Figure 3.1(a) is an example of an 2^{nd} -order Markov DBN ($r = 2$). It is



(a) A 2^{nd} -order Markov DBN.



(b) Compact Representation for the 2^{nd} -order Markov DBN

Figure 3.1: An example of a HMDBN.

represented by 3 time slices in its rolled representation. Generally, a r^{th} -order Markov DBN can be presented in $r + 1$ time slice in a rolled representation because the structures of every $r + 1$ time slices are identical. In Figure 3.1(a), there is an arrow from node A in slice $t - 2$ to node C in slice t . This arrow is not allowed in a first-order Markov DBN.

Most of the applications using DBN in gene regulation modeling use only first-order Markov DBNs [25, 37, 40]. Those applications assume that all the regulators regulate their targets with a delay of one time unit in a time se-

ries gene expression dataset. In fact, multiple time units delay is indicated to exist in a gene regulation process [9]. Using first-order DBNs to model gene regulatory networks will ignore some regulation pairs with more than one time units delay. In the thesis, a method of using r^{th} - order Markov DBN to model multiple time unit delayed gene regulatory network is proposed. Unlike first-order Markov DBN, it can model the situation that one gene regulates another gene with $1, \dots, r$ time units delay. For example, if using DBN in Figure 3.1(a) to model a gene regulatory network, it represents that gene A and gene B co-regulate gene C with two time units delay and one time unit delay respectively; gene B regulates gene A with one time unit delay. The suitable length of time delay in the scope of $1, \dots, r$ is learned from the gene expression dataset rather than using a fixed one.

In Figure 3.1(a), imaging an extra arrow from gene B to gene A across two time slices, then there will be two arrows from gene B to gene A . One is across two time slices and one is across one time slice. This situation is allowed in the definition of a 2^{nd} -order Markov DBN. But when using it to model a gene regulatory network, the situation that gene B regulates gene A with two time unit delay and one time unit delay simultaneously is hard to be explained. In the proposed method, using a HMDBN to model a gene regulatory network is based on the assumption that one gene can only be the parent of another gene with one particular time lag. Based on this assumption, the 2^{nd} -order Markov DBN in Figure 3.1(a) can be conveniently represented in a compact way as shown in Figure 3.1(b), in which the arrow from node A to node C attached with number 2 represents the arrow from node A to node C across two time slices in Figure 3.1(a).

Given a time series gene expression dataset of N genes in T time slices, to learn a r^{th} -order Markov DBN from the dataset is the task of learning a DBN structure composed by $(r + 1) \times N$ nodes. First-order Markov DBN ($r = 1$) is the simplest case of learning a structure of $2 \times N$ nodes. When r increases, the search space becomes extremely larger. Currently, most available time series gene expression data only contains a few dozen time slices. It means the size of training data is small. The problem of learning an optimal structure from a large space using limited training data makes learning the structure of a r^{th} -order Markov DBN ($r > 1$) an extremely challenging task.

3.2 A Two Steps Learning Framework

In order to address the problem of large search space, a two steps heuristic learning framework to learn the structure of a r^{th} -order Markov DBN ($r > 1$) is proposed. First, a mutual information matrix is computed to store the variable pairs with mutual information above threshold m . Second, genetic algorithm is applied to search for an optimal DBN structure by using the mutual information matrix obtained in step 1. The learning framework is presented in Algorithm 1. The details of the framework are described and explained in the following sections.

3.2.1 Pre-processing: Discretization

Before computing the mutual information and fitness functions, gene expression data is discretized first. Choosing discrete gene expression values instead of using continuous ones is because it is more easy to capture the nonlinear relationships [20]. Some researchers chose fixed threshold to discretize gene expression data into two or three categories [12]. The fixed threshold is de-

Algorithm 1 Learning framework

Pre-processing: Discretize the each gene's expression data in to 3 categories using k-mean clustering algorithm.

- 1 Compute mutual information matrix M , select the cells in the matrix whose values are above threshold m , and update M .

- 2 Genetic Algorithm ($M, r, p, m_1, m_2, w, Max, Max - fan - in$)

M : Mutual information matrix.

r : the order of the DBN.

p : The size of population.

m_1 : The percent of population to do knowledge guided mutation.

m_2 : The percent of population to do random mutation.

w : The percents of the population to do the swap.

Max : The maximal number of iterations.

$Max - fan - in$: The maximal number of parents for a node.

– Initial the *population* using M and evaluate the *population*.

– While $iterations \leq Max$

Select:

Choose $(1 - w) \times p$ individuals from *population* probabilisticly and add it to *Population_{new}*.

Crossover: Choose $0.5 \times w \times p$ pairs of individual from *population* to do the swap and add it to *Population_{new}*.

Knowledge Guided Mutation:

Choose $m_1 \times p$ individuals from *population_{new}* randomly to do knowledge guided mutation.

Random Mutation:

Choose $m_2 \times p$ individuals from *population_{new}* randomly to do random mutation.

Update:

$Population \leftarrow Population_{new}$

Evaluate:

Compute fitness function (MDL score, ML score) for *Population*

– Return the individual in *Population* with the lowest MDL score or highest ML score.

terminated according to the overall gene expression data of all the genes in the dataset. Using the fixed threshold to discretize all the genes is not delicate enough because the scopes of gene expression value vary for different genes. Discretizing every gene's data based on its own expression profile is more reasonable. In the thesis, a discretization method using k-mean algorithm [10] to group every gene's expression values into three clusters is proposed. Given one gene's expression profile composed by n values, the initial means of the three clusters are the lowest, average and highest values of the n values. Every value in the profile is then assigned to the closest cluster and the means of the three clusters are updated. This process is repeated until the the means of the three clusters are stable. Using clustering to discretize gene expression data was also adopted in Pe'er *et al.* [39]. Figure 3.2 shows the discretization results of 6 genes' expression profiles in spellman's dataset. The horizon axis is the time points and the vertical axis is the gene expression values. The profile in dashed line is the original gene expression values and the profile in solid line is the discrete values as 1, 2, 3. Figure 3.2 shows that the shapes of the discrete profiles are similar to the original ones, and most of the turning points are kept.

3.2.2 Step 1: Mutual Information Matrix

Mutual Information [3] is a natural way to measure the dependence between two variables and was employed in gene expression analysis [3,12]. It is defined by as [13],

$$I(X; Y) = \sum_{x, y} \hat{P}(x, y) \log \frac{\hat{P}(x, y)}{\hat{P}(x) \hat{P}(y)}, \quad (3.1)$$

where X, Y are two variables, x, y are particular values that X , and Y take, and \hat{P} denotes the observed frequencies in the dataset. The higher the mutual

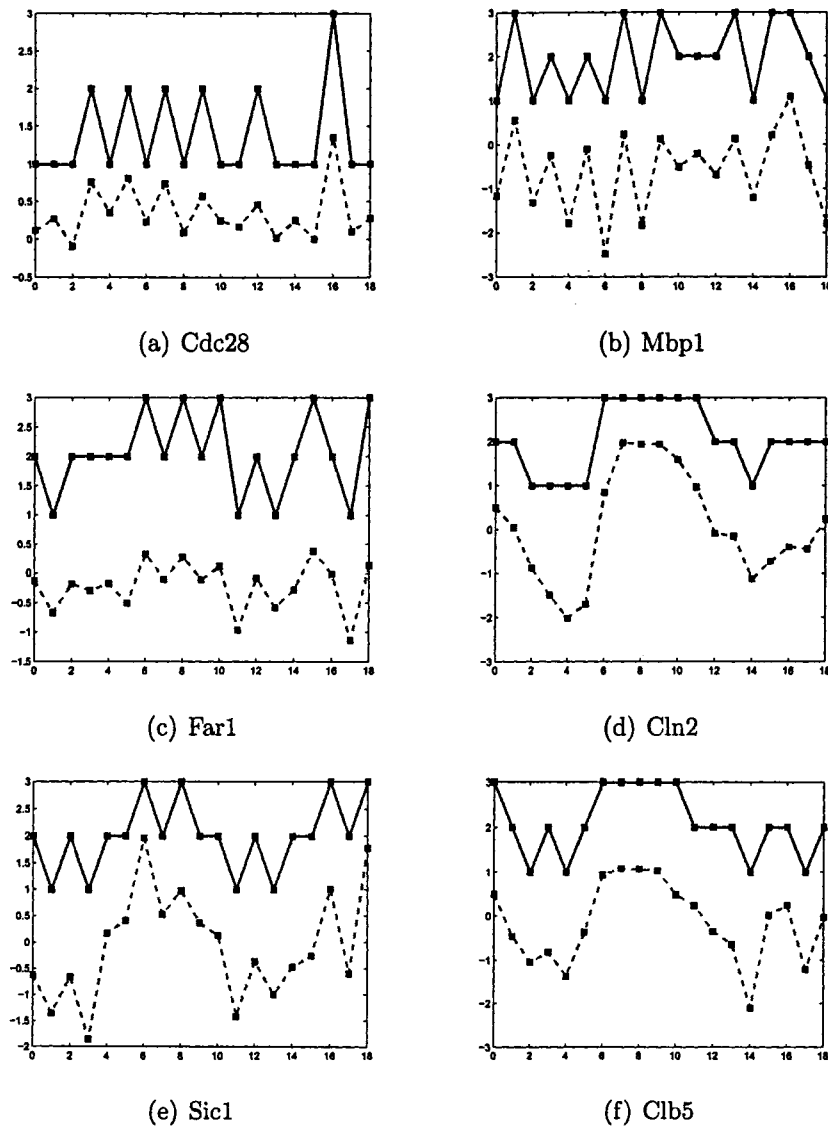


Figure 3.2: discrete profiles of six yeast genes using k-mean algorithm.

information, the stronger the dependence between X and Y .

To learn a r^{th} -order DBN of N variables evolving in time series, a matrix M with the dimension of $N \times N \times r$ is computed first. The cell $M(i, j, l)$, where $1 \leq i, j \leq N, 1 \leq l \leq r$, is the mutual information between variable X_j and X_i with time lag l (X_j precedes X_i). When computing the mutual information $M(i, j, l)$, the data of X_i needs to be shifted back l time units to align with the data of X_j [18]. The value of threshold m is set to select the candidate pairs with a mutual information above the threshold. In the proposed method, m is chosen as the mean of the overall mutual information. The matrix M is then transformed to record the candidate pairs with particular time lags using the following Equation,

$$M(i, j, l) = \begin{cases} -1 & \text{if } M(i, j, l) < m \\ l & \text{if } M(i, j, l) \geq m \end{cases} \quad (3.2)$$

$M(i, j, l) = l$ represents X_j and X_i with time lag l has a high mutual information; while $M(i, j, l) = -1$ represents the mutual information is low. When searching DBN structures, if $M(i, j, l) = l$, it is more likely that there will be an arrow from variables $X_j[t - l]$ to variable $X_i[t]$. That is to say, variable $X_j[t - l]$ is a *good candidate parent* for variable $X_i[t]$. The same idea was also employed in *Sparse candidate* algorithm [13].

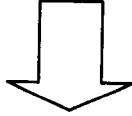
Figure 3.3 is an example of computing mutual information matrix for a 2-node 3^{rd} -order Markov DBN. A matrix of dimension of $2 \times 2 \times 3$ is first computed to store the mutual information between variable A and B with time lags 1, 2, 3 (the matrix on the top). The average value of the overall threshold

is

$$\frac{0.8 + 0.3 + 0.1 + 0.5 + 0.4 + 0.3 + 0.7 + 0.4 + 0.6 + 0.2 + 0.9 + 0.4}{2 * 2 * 3} \approx 0.5. \quad (3.3)$$

Then, 0.5 is used as the threshold to transform the mutual information matrix to store good candidate parents for every node. In the matrix at the bottom, in the intersection of column A and row B, the numbers 1, -1, 3 represents it is more possible that there is an arrow from gene A to gene B with time lag 1 or time lag 3.

Gene	A			B		
Time Lag	1	2	3	1	2	3
A	0.8	0.3	0.1	0.5	0.4	0.3
B	0.7	0.4	0.6	0.2	0.9	0.4


Threshold=0.5

Gene	A			B		
A	1	-1	-1	1	-1	-1
B	1	-1	3	-1	2	-1

Figure 3.3: Mutual information matrix for a 2-node 3rd-order Markov DBN.

Zou *et al.* [55] proposed a method to find good candidate parents for a gene with a particular time lag. In their method, given a time series gene expression dataset, in order to determine if gene A is a possible parent of gene B, a threshold is chosen to determine the time point of the initial significant change in the expression profiles of gene A and B. If gene A's significant initial change is before gene B's, then gene A is chosen as a good candidate parent of gene B. The time lag between A and B is the time difference of their significant

initial changes. The problem of this method is that it is very sensitive to the value of the threshold and noisy data. Using mutual information to choose the good candidate parents is more reliable because it is based on the dependence of a segment of gene expression profiles rather than a single time point value.

3.2.3 Step 2: Genetic Algorithm

Genetic algorithm was proposed to learn the structure of (static) Bayesian networks and dynamic Bayesian networks before [50]. In the thesis, genetic algorithm is customized to interact with the mutual information matrix to search the structure of a HMDBN.

Genetic algorithm (GA) is motivated by an analogy to biological evolution. It is designed to address the problem of searching a space of candidate hypotheses to identify the best hypothesis. In GA, hypothesis is evaluated by a predefined numerical measure for the problem, called *fitness function*. The search for the best hypothesis begins with a *population*, a collection of initial hypotheses. Individuals in the current generation give rise to the next generation by operations such as random mutation or crossover, which are inspired by biological evolution. In each generation, the hypotheses are evaluated by the fitness function and a part of hypotheses are selected probabilistically as seeds for producing the next generation. When selecting hypotheses, the one with higher fitness function will be chosen to produce offsprings with a higher probability. In this way, the new generation will become better and better [31]. To design a GA, three main parts have to be considered, including representation of hypothesis, designing of genetic operators, and selection of fitness function.

In the proposed GA of searching for an optimal structure of a HMDBN (Figure 1), individuals in a generation of population are a set of possible network structures. The structure of a HMDBN is represented as a matrix in the GA. To evaluate a structure given the dataset, score functions for the first-order Markov DBN, such as ML score and MDL score, can be extended to evaluate the structure of a HMDBN. Therefore, these score functions can be used as the fitness functions of GA. The population is initialized using the mutual information matrix produced in Step 1. Three operators, crossover, knowledge guided mutation and random mutation are designed. The details of the proposed genetic algorithm are described as below.

Representation

In the customized genetic algorithm, the structure of a r^{th} -order Markov DBN of N variables evolving in a time series is represented as a matrix I with dimension $N \times N$. Each cell $I(i, j)$, in the matrix I is defined as,

$$I(i, j) = \begin{cases} -1 & \text{if there is no arrow between} \\ & \text{variables in } \mathbf{X}_j \text{ and } \mathbf{X}_i \\ l(1 \leq l \leq r) & \text{if there is an arrow from} \\ & \text{variable } X_j[t-l] \text{ to } X_i[t]. \end{cases} \quad (3.4)$$

Table 3.1 is an example of using a 3×3 matrix I to preprint the 3-node 2^{nd} -order Markov DBN in Figure 3.1(a). In Table 3.1, each row represents a *local structure*, the structure of a node and its parents. The first row represents node A has a parent node B from previous adjacent time slice. The second row represent node B has no parents. The third row represents node C has

Gene	A	B	C
A	-1	1	-1
B	-1	-1	-1
C	2	1	-1

Table 3.1: Representation of a network structure in genetic algorithm

two parents, node A and B , from two time slices and one time slice before respectively. Note that this representation is based on the assumption that one gene only can be the parent of another gene with one particular time lag.

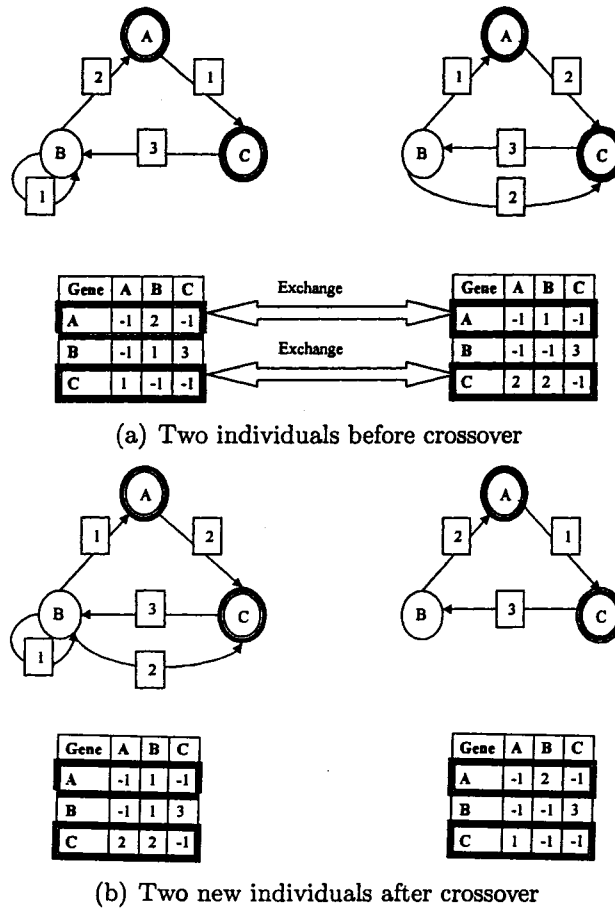
Population initialization

In order to limit the search space, the population are initialized using mutual information matrix as prior knowledge. For a r^{th} -order Markov DBN, assuming we have a matrix I representing an individual in the population and a mutual information matrix M , the cell $I(i, j)$ is initialized by randomly choosing a value from the set of values $\{M(i, j, l) \mid 1 \leq l \leq r, M(i, j, l) > -1\} \cup \{-1\}$. This initialization seeds the population to a set of structures in which all the variables pairs connected by an arrow have a high mutual information.

Operators in GA

The three operators used in the genetic algorithm are *crossover*, *knowledge guided* mutation and *random* mutation.

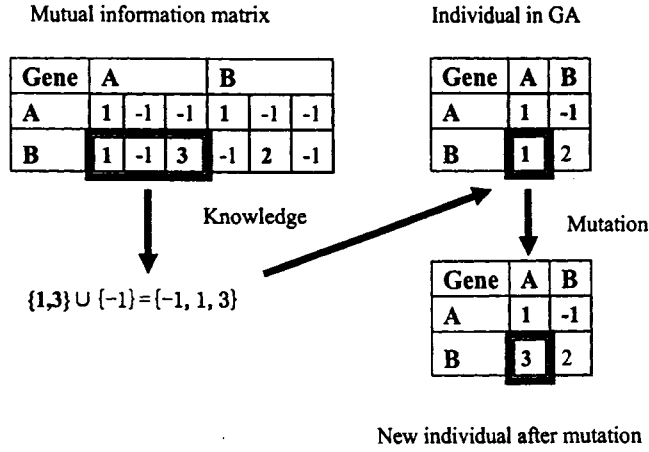
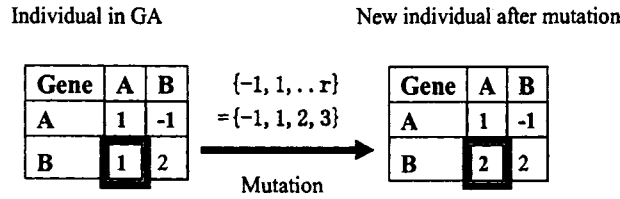
The crossover operator randomly swaps several pairs of parallel rows between two individuals in the population. Given two $a \times b$ matrixes I_1 and I_2 from a population, $\langle I_1(i), I_2(i) \rangle$, $(1 \leq i \leq a)$ is a pair of parallel rows. As

Figure 3.4: Crossover for 3-node 3rd-order Markov DBNs.

described before, every row in an individual matrix represents a local structure of a node and its parents. Using a row as the swapping unit is based on the property of ML/MDL scores that the score of the whole structure can be decomposed as the sum of local structure scores. In Figure 3.4, an example of crossover between two structures of a 3rd-order Markov DBN is shown. The local structures of node *A* and node *C* are exchanged between the two structures, and two new individuals are generated.

Knowledge guided mutation operator mutates the value in cell $I(i, j)$ by randomly choosing a value from the set of values $\{M(i, j, l) \mid 1 \leq l \leq r, M(i, j, l) > -1\} \cup \{-1\}$ with uniform distribution. Knowledge guided mutation switches the gene regulation pair to other time lags which also appear in the mutual information matrix or turns off the gene regulation relationship between two genes. In Figure 3.5, an example of knowledge guided mutation for a two-node 3rd-order Markov DBN ($r = 3$) is shown. Before mutating the value of the cell of column *A* and row *B*, a set of possible mutation values, $\{-1, 1, 3\}$, are obtained from the mutual information matrix. Then, the value of the cell in the individual is changed from 1 to 3. The value 3 is randomly picked from the set of values, $\{-1, 1, 3\}$.

Because the population initialization and knowledge guided mutation are both based on the mutual information matrix, the search space could be over-restricted by the dependence on high mutual information single arrow structures. In order to overcome the over-restriction, random mutation operator is designed. Random mutation, without referring to the mutual information matrix, mutates $I(i, j)$ randomly to a value in the set of values $\{-1, 1, 2, \dots, r\}$ with uniform distribution. In Figure 3.6, an example of random mutation for

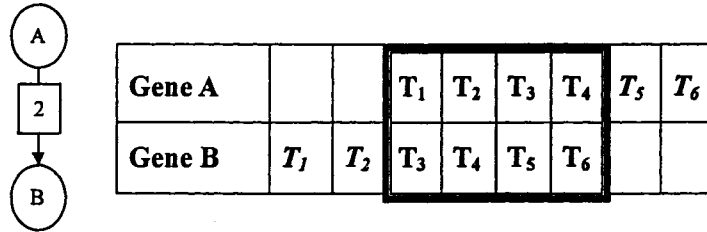
Figure 3.5: Knowledge guided mutation for a 2-node 3rd-order Markov DBN.Figure 3.6: Random mutation a 2-node 3rd-order Markov DBN.

a two-node 3rd-order Markov DBN ($r = 3$) is shown. The set of possible mutation values is $\{-1, 1, 2, 3\}$. The value of the cell in the individual is changed from 1 to 2 by randomly choosing a value from $\{-1, 1, 2, 3\}$. Without using the mutual information matrix in Figure 3.5, value 2 becomes a possible time lag.

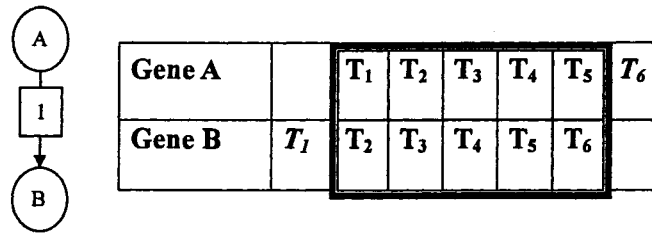
3.3 The Problem of Varied Size of Training Data

When extending the score functions of first-order Markov DBN, such as ML score and MDL score, to evaluate the structure of a higher-order Markov DBN,

a particular problem is encountered. This problem will be introduced through a simple example as blow.



(a) Structure 1, Size=4.



(b) Structure 2, Size=5.

Figure 3.7: An example of training data size variation for different HMDBN structures.

In Figure 3.7, there are two possible structures of a two-node 2^{nd} - order Markov DBN. Figure 3.7(a) is the structure that A is B 's parent with time lag 2. Figure 3.7(b) is the structure that A is B 's parent with time lag 1. To choose a better structure from the two structures, the scores of the two structure need to be compared. When scoring the structure in Figure 3.7(a), the data of gene B need to be shifted two time units back to align with the data of gene A , and the available data points can be used is $6 - 2 = 4$. When scoring the structure in Figure 3.7(b), the data of gene B need to be shifted one time unit back to align with the data of gene A , and the data points can be used is $6 - 1 = 5$. The scores of the two structures are computed based on different amount of training data, which makes the score comparison for the

two structures unreliable. In general, when evaluating different structures of a HMDBN ($r > 1$), because of different time lags, the size of training data varies for different structures. This problem does not exist when learning first-order Markov DBN because the time lag between all variables is fixed to be 1.

In order to maintain the same size of training data for different structures, one solution is to sacrifice the size of training data. For the structure in Figure 3.7(b), although there are 5 available time slices can be used as training data, in order to use the same amount of training data as the structure in Figure 3.7(a), only four time slices will be used. In general, to learn a r^{th} -order Markov DBN from a time series dataset of T time slices, only $T - r$ time slices can be used as training data. Currently, most time series dataset contains only a few dozen time slices. The decreased size makes the learning result unreliable when r becomes larger. For example, the yeast cell cycle dataset of Cho *et al.* [6] contains 17 time slices. To learn a 5^{th} -order Markov DBN, the size of available training data decreases to $17 - 5 = 12$.

Compared to sacrificing the size of training data, another solution is to simulate some time slices to maintain the same size of training data for all structures. In Figure 3.7(a), because of shifting back the data of the child node, the data of the first two time slices of the child and the last two time slices of the parent can not be used. The size of available training data decreases when time lags exist in a local structure (the structure of a node and its parents). If two time slices can be added at the end of the child's data as a complement to align with the last 2 time slices of the parent, the size of the training data will be maintained. Sampling two more time slices at the end of the time series data of the child based on the pattern of its previous

time slices is a possible solution. For some particular time series dataset, some special time slices can be chosen to simulate the time slices after the end. The yeast cell cycle dataset of Cho *et al.* [6] contains 17 time slices across nearly two complete cell cycles. The 17th time slice is the end of a cell cycle. If the 18th time slice exists, it should be the start time slice of another cell cycle; while the 1th time slice, which is a start time slice of a cell cycle, can be used to simulate the 18th time slice. Because the dataset records a period event, the data can be viewed as a loop. No matter which time slice is chosen as the start point, the end point can be found in the loop to keep the size as 17.

Figure 3.8 is an example of using time series data as a loop to maintain the

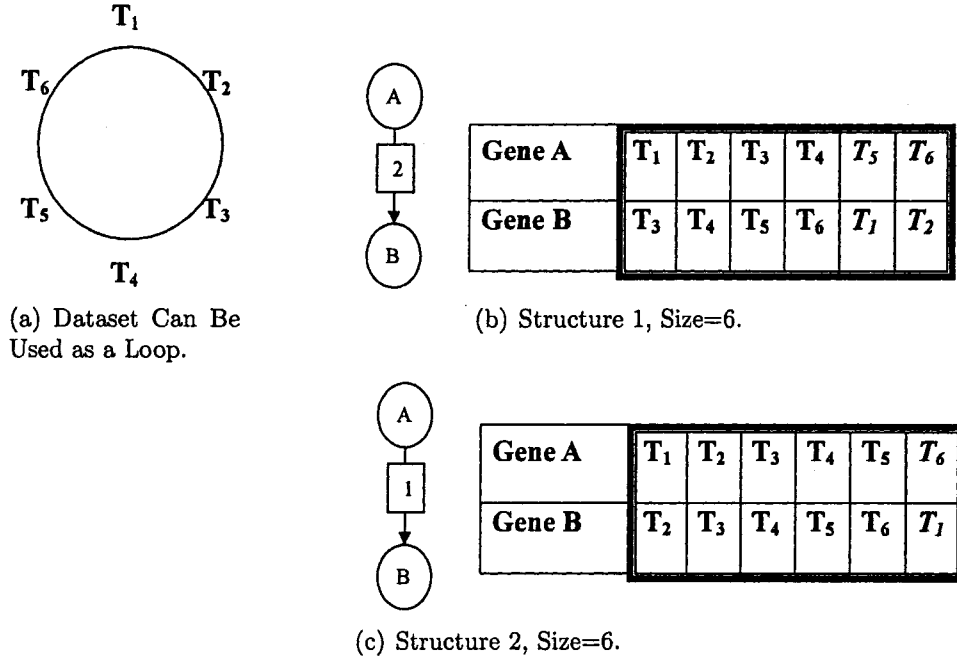


Figure 3.8: An example of using dataset as a loop.

same size of training data for different structures. A time series dataset containing 6 time slices is viewed as a loop as in Figure 3.8(a). In Figure 3.8(b), for the child node B, the start time slice of the data is T_3 and time slices T_1 and T_2 are added after time slice T_6 . The size of available training dataset for the

structure in Figure 3.8(b) is maintained to be 6. The same as Figure 3.8(b), the size of available training dataset for the structure in Figure 3.8(c) is also maintained to be 6. Compared with the example in Figure 3.7, it can be seen that using the data as loop can solve the problem of the size variation of the training data and make the score comparison for different structures reliable.

In our experiments, the two solutions, using data as a loop, and simply sacrificing the size of training dataset, are both used. For the experiment on the dataset of Chou *et al.* [6], using data as a loop are adopted. For the experiment on the dataset of Spellman *et al.* [47], simply sacrificing the amount of training data is used because the information about the cell cycle boundary is not available for this dataset.

Chapter 4

Experiments

4.1 A Nine-Gene Network

4.1.1 Dataset

To test the proposed method, the learning framework is applied to analyze a *Saccharomyces cerevisiae* gene expression dataset of Chou *et al.* [6]. In this dataset, the expression data of thousands of yeast genes were collected in time series experiments. This dataset is ideal to test the proposed method because it has a relatively large number of time slices (17 time slices) and small time intervals (10 minutes). Furthermore, the 17 time slices cross nearly two full cell cycles [6]. As mentioned in section 3.3, the dataset can be used as a loop. Figure 4.1 (taken from Chou *et al.*) shows 4 genes' expression profiles in the dataset [6]. The vertical axis is the relative expression value according to the mean of every gene's expression data. The horizontal axis is the time. One cell cycle is divided into four phases, G_1 , S , G_2 , M . In Figure 4.1, below the horizontal axis, the corresponding phases of time scopes in the experiments are presented.

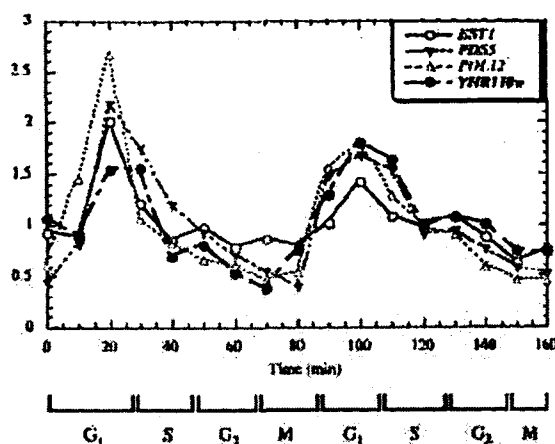


Figure 4.1: Gene expression profiles and yeast cell cycle phase information of Chou's dataset.

4.1.2 Results

Simon *et al.* [45] used genome wide location analysis to investigate nine genes (Swi4, Swi6, Swi5, Fkh2, Mcm1, Ndd1, Cln3, Ace2, Mbp1) that play a role during yeast cell cycle progression. The result revealed a nine gene regulatory network that appears to control the sequential activation of cyclins and other cell cycle regulators. It is shown in Figure 4.3(a).¹ Using Figure 4.3(a) as a comparison target, an experiment is designed to learn the HMDBN from Chou's [6] data subset containing the same nine genes.

Figure 4.3(a) (from Simon *et al.* [45]) represents a regulatory circuit in yeast cell cycle derived from genomic binding data. In Figure 4.3(a), there are four groups of genes who are active in different phases. The group of genes Swi6, Swi4 and Mbp1 are active in phase late G1; genes Mcm1, Fkh2, Ndd1 are active in phase G2/M; genes Swi5, Ace2, Mcm1 are active in phase M/G1; gene Cln3 are active in late G1 phase. Arrows represent gene regulation rela-

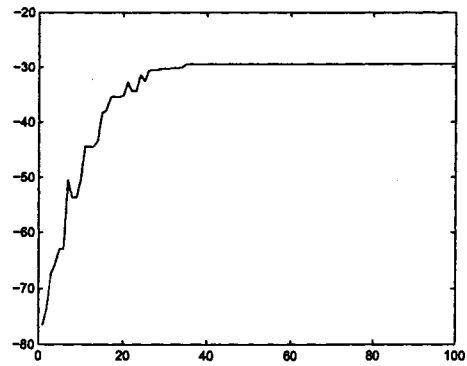
¹The figure is taken from [45].

Experiment	Fitness Function	r	Max-fan-in
1	ML Score	5	2
2	MDL Score	5	9

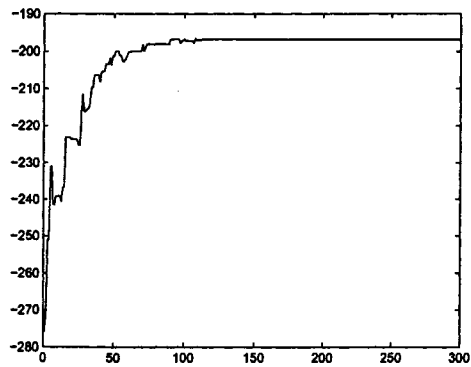
Table 4.1: Experiment Parameters.

tionships. Each group of genes regulate other genes acting in the next phase in the cell cycle. Figure 4.3(a) is used as a comparison target because it is a well-studied regulation network and it offers gene regulation relationships with corresponding phase information.

Two experiments are conducted using ML score and MDL score as fitness functions respectively. The parameters of the two experiments are shown in Table 4.1. Parameter r is the order of the DBN. A r^{th} -order Markov DBN means the maximal time lag between variables is r . Parameter *max-fan-in* is the maximal number of parents a node can have. Using ML score, the maximal number of parents has to be limited because this score prefers complex structures. Using MDL score, the maximal number of parents is not limited because this score can penalize complex structures. Figure 4.2 shows the fitness function converging plots. Figure 4.2(a) is the converging plot for ML score. The vertical axis is the highest ML score in each generation of population. The horizontal axis is the number of iterations. Figure 4.2(b) is the converging plot for MDL score. The vertical axis is the *negative* MDL score. The genetic algorithm converges around 40 iterations for the ML score and 120 iterations for MDL score. It takes more iterations to converge for MDL score because in the experiment of using MDL score, the maximal number of parents is not limited and search space is larger.



(a) ML Score Converging Plot.



(b) MDL Score Converging Plot.

Figure 4.2: Fitness function converging plots for the 9-gene network.

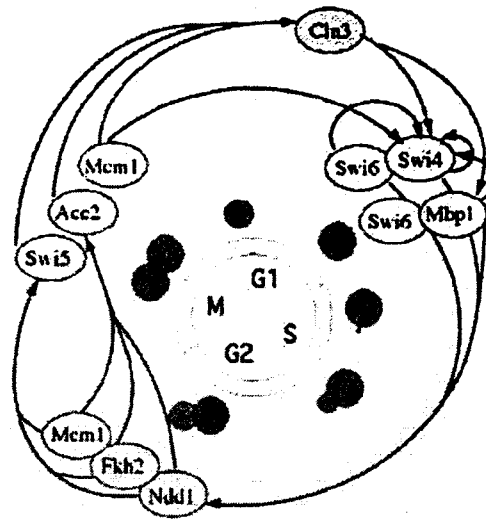
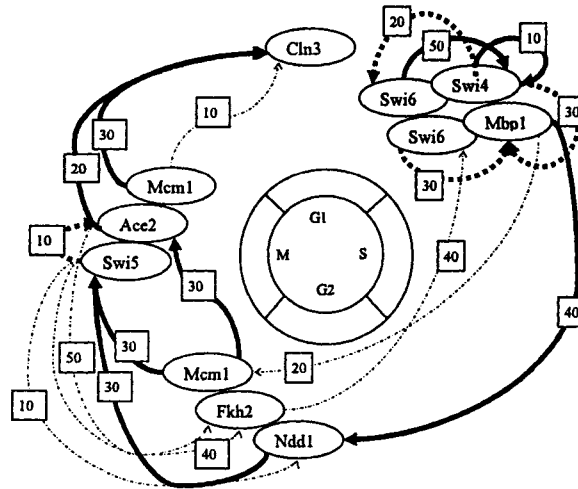
	Bold and Solid	Bold and Dashed	Dashed
10 min.	Swi4⇒Swi4	Ace2⇒Swi5	Mcm1⇒Cln3 Swi5⇒Ndd1
20 min.	Ace2⇒Cln3	Swi4⇒Swi6	Mbp1⇒Mcm1
30 min.	Ndd1⇒Swi5 Mcm1⇒Swi5 Mcm1⇒Ace2 Mcm1⇒Cln3	Swi4⇒Mbp1 Swi6⇒Mbp1	
40 min.	Mbp1⇒Ndd1		Fkh2⇒Swi6 Swi5⇒Fkh2
50 min.	Swi6⇒Swi4		Ace2⇒Fkh2
<i>Total Number</i>	<i>8</i>	<i>4</i>	<i>6</i>
<i>Support</i>	<i>Simon</i>	<i>Literatures</i>	<i>None</i>

Table 4.2: Predicted regulation pairs in Figure 4.3(b)

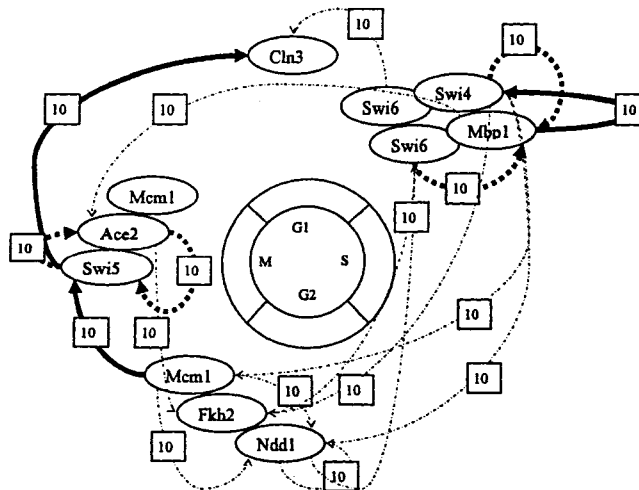
Result of ML Score

Figure 4.3(b) is the result produced by the proposed learning framework using ML score. In Figure 4.3(b), arrows represent the regulation relationships and the numbers associated with the arrows are the predicted time delay. Among the 18 arrows, 8 solid and bold arrows are the common arrows appeared in the target network (Figure 4.3(a)), 4 dashed and bold arrows are regulation relationships supported by SGD database [44], and 6 regular dashed arrows are new predicted regulation relationships (support from biological literature is not available). Taking the gene pairs connected by solid and bold arrows and dashed and bold arrows as supported relationships, the rate supported by biological evidence is $(8 + 4)/18 = 0.67$. The regulation pairs presented in Figure 4.3(b) are also summarized in the Table 4.2

Analyzing the 8 solid and bold arrows and attached time delay carefully, it is found that the predicted time delay is consistent with the yeast cell cycle phase information. In Figure 4.3(b), the arrows representing gene Mcm1 and

(a) The Network of Simon *et al.*

(b) Predicted Network by Our Learning Framework Using ML Score.



(c) Predicted Network by REVEAL using ML Score.

Figure 4.3: Results Comparison of the 9-gene network using ML score

Ndd1 co-regulating gene Swi5 together and the arrow representing gene Mcm1 regulating Ace2 are all attached with 30 minutes time delay. Compared with Figure 4.3(a), it shows that gene Mcm1 and Ndd1 are active in phase late G2/M, and gene Ace2 and Swi5 are active in phase late M/G1. Referring to the time scopes of the phases for the dataset in Figure 4.1, the time difference from late G2 phase to early G1 phase is approximately between 60 minutes to 90 minutes. The acting phase delay is $90 - 60 = 30$ minutes, which is consistent with the predicted 30 minutes time delay. In Figure 4.3(a), it shows that gene Mbp1 acting at phase late G1 regulating gene Ndd1 which is active at phase G2/M. The acting time delay cross phase late G1, S, G2. Referring to the time scope of phases for the dataset in Figure 4.1, the time difference from phase late G1 to phase late G2 is approximately between 20 minutes to 60 minutes or between 100 minutes to 140 minutes. In Figure 4.3(b), the proposed method predicts that gene Mbp1 regulates gene Ndd1 with 40 minutes time delay, which is also consistent with the phase information. Similarly, gene Ace2 regulating gene Cln3 with 20 minutes delay and Mcm1 regulating Cln3 with 30 minutes delay predicted in Figure 4.3(b) are also supported by their acting phases information in Figure 4.3(a). From the above comparison, it seems that the predicted time delay information are highly consistent with the cell cycle phase information.

Besides the solid and bold arrows, there are 4 dashed and bold arrows. The regulation pairs presented by the 4 dashed arrow do not appear in the network of Simon et al. (Figure 4.3(a)), but also can be explained. The arrow representing gene Swi5 regulating gene Ace2 is supported by the evidence that gene Ace2 and Swi5 are homologous regulators [44]. Swi4 regulating Swi6 and Swi6 regulating Mbp1 is supported by the fact that Swi4 and Swi6 compose

as a complex SBF and Swi6 and Mbp1 compose as a complex MBF [44, 45].

In Figure 4.3(a), the arrows from Cln3 to the group of genes Swi4, Swi6 and Mbp1 are post-transcriptional regulations while other arrows are transcriptional regulation. Interestingly, the predicted network by our method is nearly a full cycle except for missing arrows from gene Cln3 to the group of genes Swi4, Swi6 and Mbp1. This may indicate that the post-transcriptional regulation dependence is not hidden in gene expression data.

In order to compare the proposed HMDBN with widely adopted first-order Markov DBN, experiments using an existed algorithm, REVEAL, are also conducted. REVEAL is first proposed for learning boolean network from gene expression dataset and is adopted to learn the structure of first-order Markov DBN by Murphy *et al.* [35]. Some researchers applied REVEAL to learn gene regulatory network from discrete gene expression datasets [35, 55]. In an open source package in matlab named Bayesian Network Toolbox (BNT), REVEAL is implemented using two score functions, ML score and BIC score [33]. The experimental results below is obtained by using the REVEAL algorithm in BNT package.

An similar experiment on the same nine gene dataset is also conducted using REVEAL with ML score. In Figure 4.3(c), the result produced by REVEAL is presented. Among the 18 arrows, only 3 solid and bold arrow and 4 dashed and bold arrows. The biological evidence support rate is $3 + 4/18 = 0.39$. As mentioned above, the biological support rate for the experiment of the proposed 5th-order Markov DBN using ML score is 0.67. It seems that HMDBN can produce more supported gene regulatory pairs than

	Bold and Solid	Bold and Dashed	Dashed
10 min.	Mbp1⇒Swi4	Swi6⇒Mbp1	Cln3⇒Mcm1
20 min.	Ace2⇒Cln3 Ndd1⇒Ace2		
30 min.	Ndd1⇒Swi5		Ace2⇒Swi6
40 min.	Swi6⇒Ndd1		
50 min.			
Total Number	5	1	3
Support	Simon	Literatures	None

Table 4.3: Predicted regulation pairs in Figure 4.3(b)

first-order Markov DBN.

Result of MDL Score

Besides ML score, an experiment using MDL score is performed. The result is in Figure 4.4(b), and it is also summarized in the Table 4.3. Because the MDL score penalizes complex structure, it produces a simpler structure which contains only 9 arrows. There are 5 bold and solid arrows, 1 bold and dashed arrow, and the prediction rate supported by biological evidence is $(5 + 1)/9 = 0.67$. Compared with the result obtained by using ML score, three solid and bold arrows disappear when using MDL score. Taking gene Swi5 for example, in Figure 4.3(b), it has two parents, gene Mcm1 and Ndd1; in Figure 4.4(b), only one parent, gene Ndd1, remains. It indicates that using MDL score, the situation of several genes together co-regulating one gene are inclined to be replaced by the situation of one gene regulated by a single regulator.

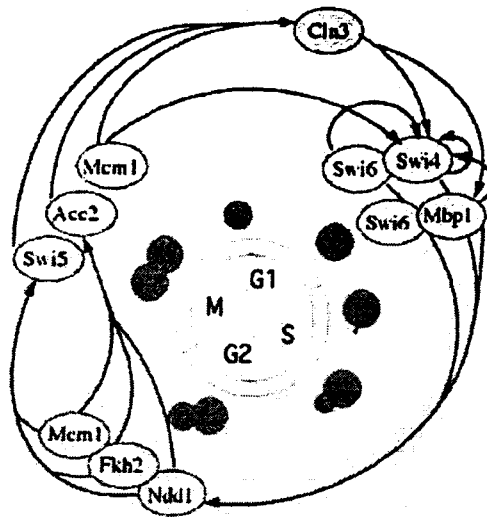
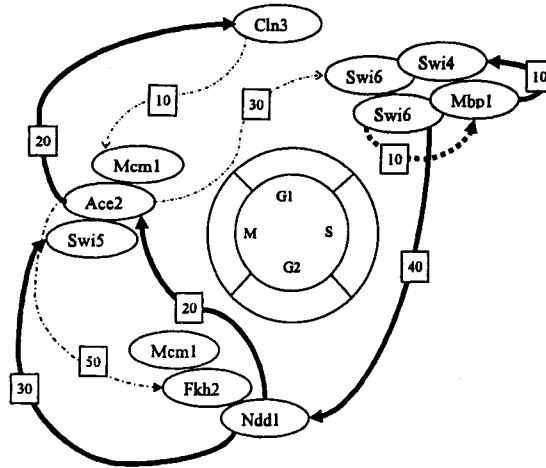
In Figure 4.4(c), the result of REVEL algorithm using BIC score is provided. BIC score is very similar to MDL score. So, using BIC score in REVEAL can be a comparison with using MDL score in the proposed method. In

Figure 4.4(c), there are two solid and bold arrows and 3 three dashed and bold arrows. The support rate by biological evidence is $(2+3)/9 = 0.56$. Compared with the result in Figure 4.4(b), it can be seen that in the result of REVEAL, the supported arrows across phase late G1 to phase G2 and across phase G2 to phase M disappear. It may indicate that first-order Markov DBN are good at finding regulatory pairs existed in the same phase but weak in finding regulatory pairs existed across phases. While higher-order Markov DBN can handle the regulatory pairs both with large time delay and small time delay.

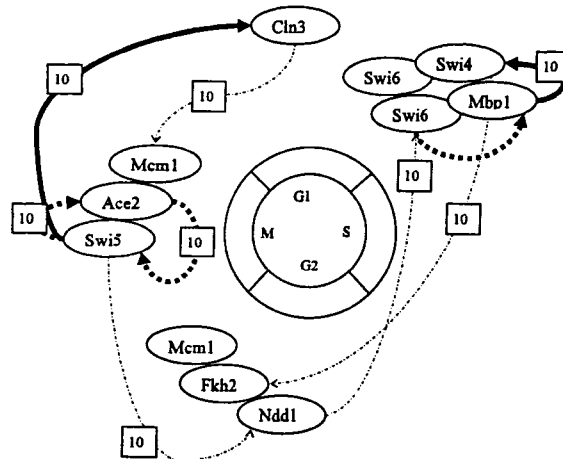
4.1.3 Parameter Selection

Using r^{th} -order Markov DBN, the parameter r has to be chosen properly. The above experimental results are obtained using 5th-order Markov DBN ($r = 5$), which means the maximal regulatory time lag between genes are 50 minutes. Observing Simon's regulatory network in Figure 4.3(a), it was found that regulatory pairs are across maximal three phases. Compared to the phase information in Figure 4.1, 50 minutes ($r = 5$) can be a proper maximal time lag to allow the HMDBN to capture all the potential gene regulatory pairs in Figure 4.3(a). As shown in Section 4.1.2, some interesting pairs with 30 and 40 minutes delay are found by the proposed method. If r is chosen to be a small value, such as 2, these pairs with large time delay will be shielded out. But if choosing a very large number for r , such as $r = 16$, then the search space will become extremely large. It is reasonable to choose r as a relative large number by using some available prior knowledge.

Besides conducting the experiment with $r = 5$, the experiments using $r = 6$ and $r = 7$ are also conducted using ML score. The results are shown in Figure 4.5(a) and Figure 4.5(b) respectively.

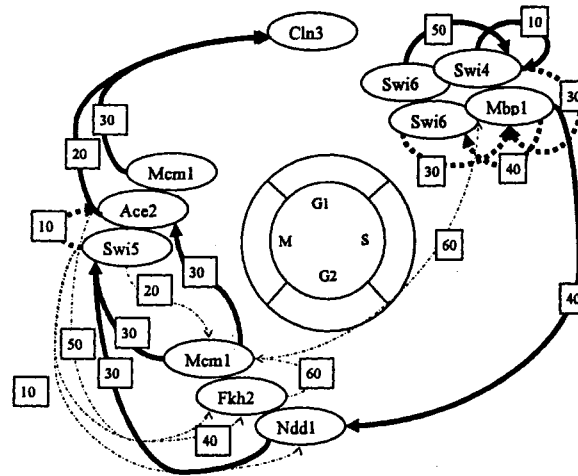
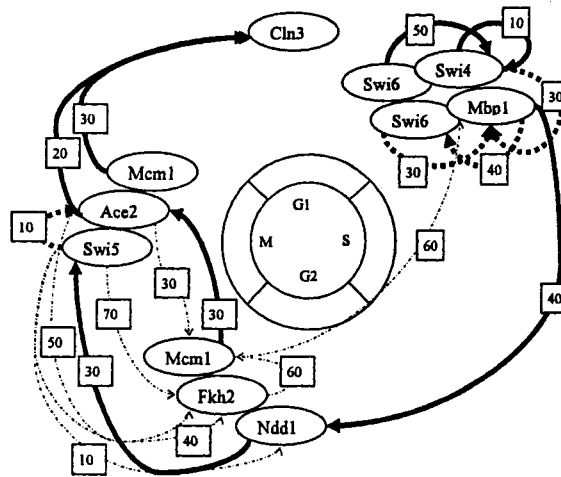
(a) The Network of Simon *et al.*

(b) Predicted Network by Our Learning Framework Using MDL Score.



(c) Predicted Network by Our Learning Framework Using ML Score.

Figure 4.4: Results Comparison of the 9-gene network using MDL score.

(a) Result of $r = 6$ (b) Result of $r = 7$ Figure 4.5: Learning results of $r > 5$.

In Figure 4.5(a), compared with the result of $r = 5$ in Figure 4.3(b), all the 8 solid and bold arrows are kept the same. One dashed and bold arrow and three regular dashed arrows are different. In Figure 4.5(b), compared with the result of $r = 5$, one solid and bold arrows changes, and other 7 are kept. From the above comparison, it shows although the result will be affected by different values of r , the result are stable when lager value for r is chosen.

4.2 A Fourteen-Gene Network

4.2.1 Dataset

Another gene expression dataset used in the experiment is collected by Spellman *et al.* [47]. Similar to chou's dataset, it is another widely used dataset for monitoring the gene expression levels of thousands of genes in yeast cell cycle. This dataset contains two short series, Cln3 (2 time points) and Clb2 (1 time points), and three medium time series, alpha (18 time points), Clu (14 time points) and Cdc15 (24 time points). Cdc15 series records the gene expression profiles in 300 minutes in the yeast cell cycle with intervals of 10 minutes or 20 minutes. In order to get a series with equal intervals, 19 time slices in Cdc15 series are used in the experiment. With a equal interval of 10 minutes, it starts from the time slice of 70 min. and ends at the time slice of 250 min.

4.2.2 Result

The experiments are designed to learn the yeast cell cycle pathway stored in the KEGG database [23]. The whole pathway in KEGG contains 45 genes. The experiments focus on recovering a partial pathway containing 14 genes in KEGG. In Figure 4.7(a), the 14 genes pathway in KEGG are presented.

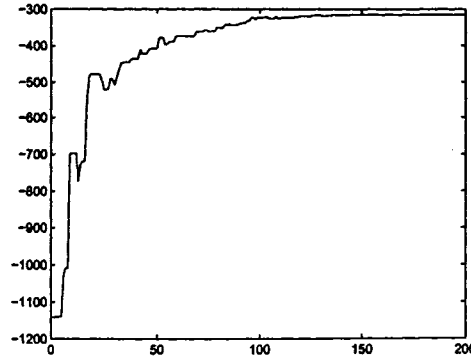


Figure 4.6: Fitness function converging plots for the 14-gene network.

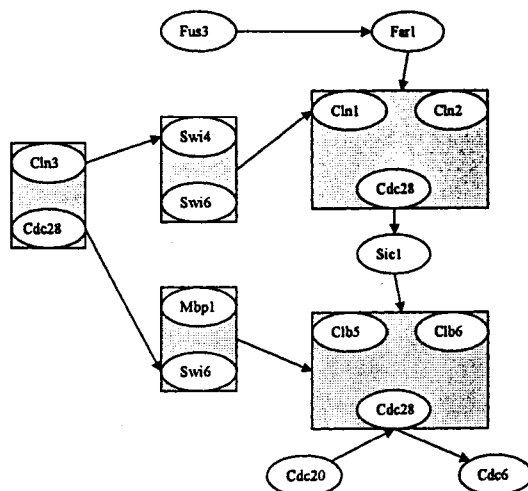
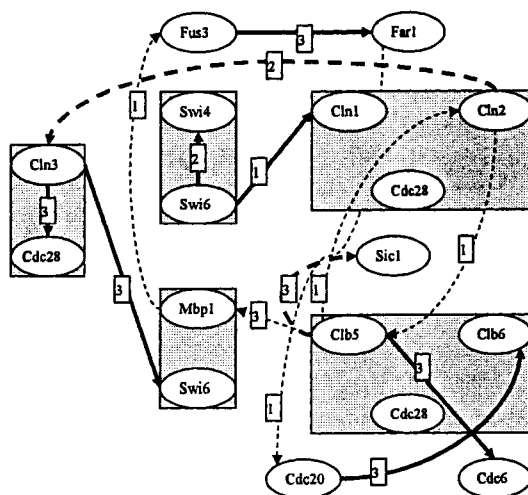
The training dataset is a subset of the gene expression dataset of Spellman *et al.* [47] containing the same 14 genes, Fus3, Far1, Cln3, Cdc28, Swi4, Swi6, Mbp1, Cln1, Cln2, Sic1, Clb5, Clb6, Cdc20 and Cdc6.

The target network in Figure 4.7(a) shows the regulatory network (pathway) of 4 single genes and three groups of genes. Because every gene and every group of genes only has one regulator, choosing MDL score in the experiment is expected to find a similar sparse structure. Observing the pathway existed in two adjacent phases, phase G1 and S, the time delay between genes are not likely to be very large, 3rd-order Markov DBN ($r = 3$) is chosen to represent the network. It means the maximal time delay between genes is allowed to be 30 minutes. When learning the dataset of the 14 genes, genetic algorithm still converge well on this larger dataset. Figure 4.6 shows the converging plot of the MDL score in genetic algorithm.

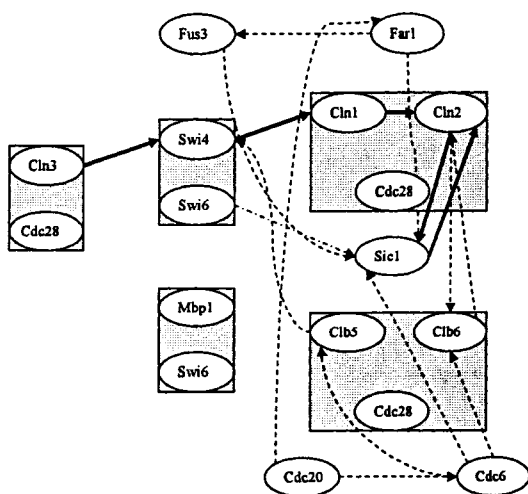
The predicted network by the proposed method is shown in Figure 4.7(b). Among the 14 predicted arrows, 7 solid and bold arrows are consistent with the target network (Figure 4.7(a)), 2 dashed and bold arrows are supported

by SGD database [44] and 5 dashed arrows lack of supporting.

In Kim *et al.* [25], an experiment result of the same 14 genes on the dataset of Spellman *et al.* [47] is also reported. Their result, which is shown in Figure 4.7(c), is obtained by using first-order Markov DBN and nonparametric regression. Among their 17 predicted arrows, 6 are consisted with the pathway in KEGG. The rate supported by KEGG pathway is $6/17 = 0.35$. In our method, the rate supported by KEGG pathway is $7/14 = 0.5$. Compared the two results, it can be seen that the supported regulatory pairs predicted by the proposed method and Kim's method are complementary to each other. It may indicated that HMDBN could capture some interesting regulatory pairs which can not be got by first-order Markov DBN. At the same time, some useful regulatory pairs with small time lags may be shielded by some incorrect pairs with larger time lags. If a correct regulator of a regulatee with one time unit delay is replaced by another regulator with a larger time delay in HMDBN, it is because the dependence is more stronger in the latter case. Although the correct one is not found, it can not be judged only based on the statistical score from gene expression data, more information need to be provided.

(a) Pathway in Kegg *et. al.*

(b) Predicted Network by our method



(c) Predicted network in Kim

Figure 4.7: Results Comparison of the 14-gene network.

Chapter 5

Conclusion and Future Work

5.1 Contributions

In this thesis, a new type of DBN, higher-order Markov dynamic Bayesian network (HMDBN), is proposed to model multiple time units delayed gene regulatory network. Most of applications of DBN in gene regulation analysis only use first-order Markov DBN to model gene regulatory networks with a fixed length of time delay. Although in the work of Zou *et al.* [55], multiple time units delay is incorporated into DBN, it ignores the situation that several regulators regulating a regualtee with different time lags. To the best of our knowledge, this thesis presents the first application that considers all the possible combinations of the candidates regulators in all possible time lags using HMDBN.

In order to address the learning problem of large search space of HMDBN compared to the limited size of time series gene expression data, a two step heuristic learning framework is designed. First, a mutual information matrix is computed. Second, a genetic algorithm is applied to search for an optimal

structure using mutual information matrix as prior knowledge. A particular problem rose in learning HMDBN is that the size of applicable training data varies when scoring different network structures. The possible solutions to this problem are discussed in the thesis, and applied in the experiments.

In order to verify the effectiveness of the proposed method, it is applied to learn gene regulatory networks from two different gene expression datasets of yeast cell cycle. The results show that the proposed method can produce meaningful gene regulatory network which is strongly supported by biological evidence and highly consistent with yeast cell cycle phase information. Compared with the results obtained by first-order Markov DBN, it reveals that higher-order Markov DBN can find more supported gene regulatory pairs from some dataset.

5.2 Future Work

In the proposed learning framework, two score functions, ML score and MDL score are used. The different results of the two score functions show that score function is a key factor to affect the learning result. MDL score prefers simple network structures, and typically the structure that each node only has one parent. ML score prefers complex structures, and the suitable number of parents must be limited. Although both scores can find useful structures, they are both not perfect. In the future, more available score functions are going to be investigated.

Learning gene regulatory networks from gene expression data is only based

on the statistical patterns hidden in the gene expression dataset. Some researchers argue that gene expression data does not contain enough information to construct gene regulatory network, especially for large gene regulatory networks [19,26]. This argument is reasonable. In our experiments, it is found that although the regulatory pairs which are supported by the biological evidence has a high statistical score, sometimes, they are not the pairs with the highest scores. Sometimes, only picking the structure with the highest score will lead to a mis-understanding because the training dataset is not large enough and the learning result is not statistically reliable enough. More information need to be provided to choose the real structures from a pool of high score structures. Furthermore, the direct regulation and indirect regulation is hard to be distinguished only from gene expression data. One of possible solutions to this problem is to providing more information other than gene expression data, such as prior knowledge from literatures and protein interactions. The proposed learning framework has the potential to incorporate different kinds of prior knowledge. In this thesis, mutual information matrix is used as the prior knowledge. In the future, the possibilities of adding other kinds of prior knowledges into the proposed learning framework is going to be explored.

In the proposed method, it is assumed that one gene can not be the regulator of another gene with more than one different time lags. This assumption may limit the situation that one regulators regulating its regulatee in several different pathways. Solutions to this limitation need to be investigated in the future.

Bibliography

- [1] B. Alberts, D. Bray, J. Lewis, M. Ra, and K. Roberts. *Molecular Biology Of The Cell*. Garland Publishing, Inc., 1994.
- [2] P. Baldi and D. Long. A bayesian framework for the analysis of microarray expression data: regularized t-test and statistical inferences of gene changes. *Bioinformatics*, 17:509–510, 6 2001.
- [3] A. Butte and I. Kohane. Mutual information relevance networks: functional genomic clustering. *Pac Symp Biocomput*, 418–29, 2000.
- [4] T. Chen, H.I. He, and G.M. Church. Modelling gene expression with differential equations. *Proc. Symp. biocomput.*, 4:29–40, 1999.
- [5] D. Chickering, D. Geiger, and D. Heckerman. Learning bayesian networks is np-hard. *Learning Bayesian Networks: The Combination of Knowledge and Statistical Data*, 1997.
- [6] R.J. Chou, M.J. Campbell, E.A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, and T.G. Wolfsberg. A genome-wide transcriptional analysis of the mitotic cell cycle. *Mol. Cell*, 2:65–73, 1998.
- [7] G.F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:247–309, 1992.
- [8] P. D'haeseleer, S. Liang, R. Somogui, and D. Bostein. Genetic network inference : from co-expression clustering to reverse engineering. *Bioinformatics*, (16):707–726, 2000.
- [9] M.S. Dasika, A. Gupta, and C.D. Marans. A mixed integer linear programming (milp) framework for inferring time delay in gene regulatory networks. *Pac Symp Biocomput*, pages 474–85, 2004.
- [10] M.H. Dunham. *Data Mining Introductory and Advanced Topics*. Prentice Hall, pages 140–142, 2003.
- [11] M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. Natl Acad. Sci. USA*, 95:14863–14868, 1998.

- [12] N. Friedman, M. Linial, I. Nachma, and D. Pe'er. Using bayesian networks to analyze expression data. *Proc. Fourth Annual Inter. Conf. on Computational Molecular Biology (RECOMB)*, 2000.
- [13] N Friedman, K. Murphy, and S. Russell. Learning the structure of dynamic probabilistic networks. In *Fourteenth Conf. on Uncertainty in Artificial Intelligence (UAI)*., 1999.
- [14] N. Friedman, D. Pe'er, and I. Nachman. Learning bayesian network structure from massive datasets: The "sparsecandidate" algorithm. In *In Proc. Fifteenth Conf. on Uncertainty in Artificial Intelligence (UAI)*, 1999.
- [15] Z. Ghahramani. Learning dynamic bayesian networks. *Lecture Notes in Computer Science*, 1378:168–189, 1997.
- [16] A.J. Hartemink, D.K. Gifford, T.S. Jaakkola, and R.A. Young. Combining location and expression data for principled discovery of genetic regulatory network models. *Pac Symp Biocomput*, pages 437–49., 2002.
- [17] D. Heckerman. A tutorial on learning with bayesian networks. Technical report, Microsoft Research, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, 1996.
- [18] D. Husmeier. Dbmcmc package for matlab. Technical report, URL: <http://www.bioss.sari.ac.uk/dirk/software/DBmcmc/>, 2003.
- [19] D. Husmeier. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with *Bioinformatics*, 19:2271–2282, 2003.
- [20] D. Husmeier, R. Dybowski, and S. Roberts. *Probabilistic modeling in bioinformatics and medical informatics*. Springer, 2005.
- [21] S.H. Imoto, T. Goto, and S. Miyano. Estimation of genetic networks and functional structures between genes by using bayesian networks and *Pac Symp Biocomput*, pages 175–86, 2002.
- [22] S.H. Imoto, T. Goto, K. Tashiro, S. Kuhara, and S. Miyano. Combining microarrays and biological knowledge for estimating gene networks via bayesian networks. *Bioinformatics and Computational Biology*., 2(1):77–98, Mar 2004.
- [23] KEGG. Kegg pathway database. <http://www.genome.jp/kegg/pathway.html>.
- [24] S.Y. Kim, S. Imoto, and S. Miyano. Inferring gene networks from time series microarray data using dynamic bayesian networks. *Briefings in Bioinformatics*, 4(3):228–235, Sep 2003.

- [25] S.Y. Kim, S. Imoto, and S. Miyano. Dynamic bayesian network and nonparametric regression for nonlinear modeling of. *BioSystems*, 75:57–65, 2004.
- [26] S. Knudsen. *Guide to analysis of DNA microarray data*. Wiley-Liss, 2004.
- [27] D. L. Hartl and E. W. Jones. *Genetics: principles and analysis*. Jones and Bartlett Publishers, 1998.
- [28] W. Lam and F. Bacchus. Learning bayesian belief networks: an approach based on the mdl principle. *Computational Intelligence.*, 10(4):269–293, 1994.
- [29] X. Li, S. Rao, W. Jiang, and C. Li. Discovery of time-delayed gene regulatory networks based on temporal gene expression profiling. *BMC Bioinformatics*, 7(1):7–26, 2006.
- [30] S. Liang, S. Fuhrmann, and R. Somogyi. Reveal, a general reverse engineering algorithm for inference of genetic network architectures. *Proc. of Pacific Symposium on Biocomputing.*, (3), 1998.
- [31] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.
- [32] U.R. Muller and D.V. Nicolau, editors. *Microarray Technology and its applications*. Springer, 2005.
- [33] K. Murphy. The bayes net toolbox for matlab. In *Computing Science and Statistics: Proceedings of Interface*, 2001.
- [34] K. Murphy. A brief introduction to graphical models and bayesian networks. 2001.
- [35] K. Murphy and S. Mian. Modelling gene expression data using dynamic bayesian networks. Technical report, U.C.Berkeley, 1999.
- [36] R.E. Neapolitan. *Learning Bayesian network*. Prentice Hall, first edition, 2003.
- [37] 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.
- [38] D. Pe’er. *From Gene Expression to Molecular Pathways*. PhD thesis, The Hebrew University, Nov 2003.
- [39] D. Pe’er, A. Regev, G. Elidan, and N. Friedman. Inferring subnetworks from perturbed expression profiles. *Bioinformatics*, 17(1):215–224, 2001.

- [40] B.E. Perrin, L. Ralaivola, and A. Mazurie. Gene networks inference using dynamic bayesian networks. *Bioinformatics*, 19(2):138–148, 2003.
- [41] J. Qian, M. Dolled-Filhart, J. Lin, H.Y. Yu, and M. Gerstein. Beyond synexpression relationships: local clustering of time-shifted and inverted gene expression profiles identifies new, biologically relevant interactions. *J Mol Biol.*, 314(5):1053–66, 2001.
- [42] S.J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition, December 2002.
- [43] E. Segal, M. Shapira, A. Regev, D Pe’er, D. Botstein, D. Koller, and N. Friedman. Module networks: Identifying regulatory modules and their condition specific regulators from gene expression data. *Nature Genetics*, 34(2):166–76, June 2003.
- [44] SGD. Saccharomyces genome database. <http://www.yeastgenome.org>.
- [45] I. Simon, J. Barnett, N. Hannett, C.T. Harbison, N.J Ranaldi, T.L. Volkert, and J.J. Wyrick. Serial regulation of transcriptional regulators in the yeast cell cycle. *Cell*, 106:697–708, Sep 2001.
- [46] V.A. Smith, E.D. Jarvis, and A.J. Hartemink. Estimation functional network inference using of complex biological systems. *Bioinformatics*, 18(1):216–224, 2002.
- [47] P.T. Spellman, G. Sherlock, M.Q. Zheng, V.R. Iyer, K. Anders, M.B. Eisen, P.O. Brown, D. Bostein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by micorarray hybridization. *Molecular biology of the cell*, 9:3272–3297, Dec 1998.
- [48] Y. Tamada, S.Y. Kim, and H. Bannai. Estimating gene networks from gene expression data by combining bayesian network model with promoter element *Bioinformatics*, 19(2):227–236, 2003.
- [49] T. Tian and K. Burrage. Stochastic neural network models for gene regulatory networks. *Evolutionary Computation*, pages 8–12, Dec 2003.
- [50] A. Tucker and X. Liu. Extending evolutionary programming methods to the learning of dynamic bayesian networks. *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 13–17, July 1999.
- [51] ML. Whitfield, G.S. Sherlock, AJ. Saldanha, JI. Murray, CA. Ball, JC. Alexander, KE. Matese, CM. Perou, MM. Hurt, PO. Brown, and D. Botstein. Identification of genes periodically expressed in the human cell cycle and their expression in tumors. *Molecular biology of cell*, 13:1977–2000, 2002.

- [52] Wikipedia. the free encyclopedia url:
<http://en.wikipedia.org/wiki/bioinformatics>. 2005.
- [53] David W.Mount. *Bioinformatics: Sequence and Genome Analysis*. Cold Spring Harbor Laboratory Press, second edition, 2004.
- [54] D.E. Zak, F.J. Doyle, and J.S. Schwaber. Local identifiability: when can genetic networks be identified from microarray data? *Proceedings of the third international conference on systems biology.*, pages 236–237, 2002.
- [55] M. Zou and S.D. Conzen. A new dynamic bayesian network (dbn) approach for identifying gene regulatory networks from time course *Bioinformatics*, 21(1):71–79, 2005.

Appendix A Source Code

K-mean Discretization

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%function of k_means
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function output=k_means(vector)
global n;
n=length(vector);
global k;
k=3;
global a;
a=vector';
add =zeros(length(a),1);
a=[a,add];
global mean0;
mean0=zeros(k,1);

mean1=zeros(k,1);
global dis;
dis=zeros(1,2);
same=0;

% initial by the large , small and mean

% mean

mean0(1,1)= min(vector);
mean0(2,1)=mean(vector);
mean0(3,1)=max(vector);

% k mean
while (1)
```



```

    assign;

    % update k mean
    for i=1:k
        count=0;
        sum=0;
        for j=1:n
            if a(j,2)==double(i)

                count=count+1;
                sum=sum+a(j,1);

            end

        end

        % in case there are some empty class
        if(count>0)
            mean1(i)=sum/count;
        end
    end

    % check if mean remain the same
    if mean1==mean0 break;
    else mean0=mean1;
    end

    output= a(:,2)';

    % end of k-mean

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%function assign
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function assign
global a;
global mean0;
global dis;

```

```

global k
global n;

for i=1:n
    v=a(i);

    for ii=1:k
        dis(ii,1)=ii;
    end

    for j=1:k
        m=mean0(j);
        dis(j,2)= abs(v-m);
    end

    dis=sortrows(dis,2);
    a(i,2)=dis(1,1);
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%function to discretize gene expression data into three values
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function D1=discrete3(D)

ng=length(D(:,1));
time=length(D(1,:));

for i=1:ng
    vector=D(i,:);
    D1(i,:)=k(vector);
    % output some graph

    a=[0:time-1];
    if i<10
        figure,
        plot(a,vector,'--gs','LineWidth',2,...
            'MarkerEdgeColor','k',...
            'MarkerFaceColor','g',...
            'MarkerSize',5);
    end
end

```

```

        hold on

        plot(a,100*D1(i,:), '--rs', 'LineWidth', 2, ...
            'MarkerEdgeColor', 'k', ...
            'MarkerFaceColor', 'r', ...
            'MarkerSize', 5);
        %title(rowheaders(i));

        hold off
    end
    % output some graph

end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% test discrete3 function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
data;
rowheaders;
output=discrete3(data);
output
save('9regulator.txt','output','-ASCII');
% a=load('dis39regu.txt')

```

Score Functions

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% score function for one individual in the population
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [BICscore, LLscore,MDLscore]
= score(individual, data,ntimepoint,n)
% for one gene
for i=1:n
    onegenerepre=individual(i,:);

    D=[];
    maxlag=max(onegenerepre);
    for k=1:length(onegenerepre)
        infactlag=onegenerepre(k);
    end
end

```

```

        if infactlag>-1
            if infactlag==maxlag
                D=[D;data(k,:)];
            else
                beforeshift=data(k,:);

                aftershift=[beforeshift(1+maxlag-infactlag:ntimepoint)
                beforeshift(1:maxlag-infactlag)];

                D=[D;aftershift];
            end
        end

    end

    end

    % this D contains the target gene itself as a regulator
    D;
    B = data(i,:);
    B=[B(1+maxlag:ntimepoint) B(1:maxlag)];

    D = [D;B];
    % D=addlength(D,ntimepoint);
    counts = compute_counts(D, 3*ones(1,length(D(:,1))));
    CPT = mk_stochastic(counts);

    [bic_score_onegene(i), ll_score_onegene(i),mdl_score_onegene(i)]
    = bic_score_family(counts, CPT, ntimepoint);

end

    % sum the score for every gene given it's parents
    BICscore=sum(bic_score_onegene);
    LLscore=sum(ll_score_onegene);
    MDLscore=sum(mdl_score_onegene);

end

%MDL,ML, BIC Score function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [S, LL,MDL] = bic_score(counts, CPT, ncases)
% modified from the original one from BNT package
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% BIC_SCORE Bayesian Information Criterion score for

```

```

% a single family
% [S, LL] = bic_score(counts, CPT, ncases)
%
% S is a large sample approximation to
% the log marginal likelihood,
% which can be computed using dirichlet_score.
%
%  $S = \log \left[ \prod_j \prod_k \theta_{ijk}^{N_{ijk}} \right]$ 
%      - 0.5*d*log(ncases)
% where counts encode  $N_{ijk}$ ,
%  $\theta_{ijk}$  is the MLE computed from counts,
% and d is the num of free parameters.

%CPT = mk_stochastic(counts);
tiny = exp(-700);
LL = sum(log2(CPT(:) + tiny) .* counts(:));
% CPT(i) = 0 iff counts(i) = 0 so it is okay to add tiny

ns = mysize(counts);
ns_ps = ns(1:end-1);
ns_self = ns(end);
nparams = prod([ns_ps (ns_self-1)]);
% sum-to-1 constraint reduces the effective num. vals of the node by 1

S = LL - 0.5*nparams*log2(ncases);

% MDL score
d=1;
% CPT+tiny
% new=1/(CPT+tiny)
% for the small data set
%n=9;
% for the large data set
n=14;
DLdata = sum(log2(1./(CPT(:) + tiny)) .* counts(:));
DLmodel=length(ns_ps)*log2(n)+d*nparams;
MDL=DLdata+DLmodel;
MDL=-MDL;

```

Mutual Information Matrix

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% generate mutual information matrix

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
clear all
maxtimelag=7;

data= load('9regulator.txt');

numberofgene=length(data(:,1));
time=length(data(1,:));
mutualInfo=cell(numberofgene,numberofgene);
t = cputime;
for n1=1:numberofgene
    for n2=1:numberofgene

i=0;
for dt=1:maxtimelag
    seq1=data(n1,:)
    seq2=data(n2,:)
    i=i+1;
    % n2 regulate n1

%    MutualInfo(seq1,seq2,dt)
    if dt>0
        seq1=[seq1(1+dt:time) seq1(1:1+dt-1)]
        end
        output='.....'
        % seq2 is regulator

        mutualInfo{n1,n2}(i)=MutualInfo(seq2,seq1,0);
    end

%
    end % end n1

end %end n2
e = cputime-t
mutualInfo;

savematrix=cell2mat(mutualInfo);

save('mutualinfomatrix.txt','savematrix','-ASCII');
savematrix=load('mutualinfomatrix.txt');

mutualInfo2

```

```

=mat2cell(savematrix, ones(1,numberofgene),
(maxtimelag)*ones(1,numberofgene))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Transform mutual information matrix above threshold
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% above average mutual
clc
% clear all
threshold=0.2043;

data= load('9regulator.txt');

rowheaders
numberofgene=length(data(:,1));
savematrix=load('mutualinfomatrix.txt');
matrix=savematrix;
mutualInfo2
=mat2cell(savematrix, ones(1,numberofgene),
(maxtimelag)*ones(1,numberofgene));

% let the mutual information less than time lag 4

for i=1: numberofgene
    for j=1: numberofgene

        temp=mutualInfo2{i,j};
        new=[];
        for n=1:maxtimelag
            value=temp(n);
            % use this condition to control maxtimelag

            if value>threshold && n<=maxtimelag
                block=[n;value];
                new=[new block];
            end
        end
        if length(new)==0
            mutualInfo3{i,j}=0;
        else
            mutualInfo3{i,j}=new;
        end
    end
end

```

```

        end
    end

    for i=1:numberofgene
        for j=1:numberofgene
            strcat(rowheaders(j), ' regulate: ', rowheaders(i))

            mutualInfo3{i,j}
        end
    end

    end

    save('thresholdmutual.mat','mutualInfo3')

```

Genetic Algorithm

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Crossover in genetic algorithm
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Swap(nofswap)
global sizeofpopulation;
global population;
global newpopulation;
% global populationold;
% populationold=population;
global numberofgene;
global remain;
global maxfanin;
global maxtimelag;
maxtimelag=7;
crossarray=randperm(sizeofpopulation);
crossarray=crossarray(1:nofswap);

% uniform crossover
% for every pair
remainadd=1;
for i=1:length(crossarray)/2
    % uniform crossover mask
    a = 0; b = 1;
    mask = a + (b-a) *round( rand(1,numberofgene));
    % crossarray(i)
    % crossarray(nofswap+1-i)

```



```

% swap according to mask
for j=1:length(mask)
    if mask(j)==1

        temp=population{crossarray(i)}(j,:);
        population{crossarray(i)}(j,:)=
population{crossarray(nofswap+1-i)}(j,:);
        population{crossarray(nofswap+1-i)}(j,:)=temp;

    end

end

% put it into newpopulation
newpopulation(remain+remainadd)
=population(crossarray(i));
    remainadd=remainadd+1;
    newpopulation(remain+remainadd)
=population(crossarray(nofswap+1-i));
    remainadd=remainadd+1;

end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% knowledge guided mutation in genetic algorithm
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function lagmutate()
global thrmutualInfo1;
global mutatenumbr;
global population;
global newpopulation;
global numberofgene;
global sizeofpopulation;
global maxfanin;
% Loop for individual in the population
numberofpair=numberofgene*numberofgene;
mutatearray=randperm(sizeofpopulation);
mutatearray
=mutatearray(sizeofpopulation-mutatenumbr:sizeofpopulation);
% define the percent to mutate in one individual

```

```

% lower=round(0.2*numberofpair)
% upper=round(0.9*numberofpair)

for i=1: mutatenumber

    tempindi=newpopulation{mutatearray(i)};

    % random generate a number of pair between lower and upper
    %   rannpair=lower + (upper-lower) *round( rand(1));
    count=0;
    % loop for every gene pair mutaiton, mutate the time lag.
    for ii=1: numberofgene
        nofp=0;

        for cc=1:numberofgene
            if tempindi(ii,cc)>-1
                nofp=nofp+1;
            end
        end

        for jj=1:numberofgene

            outout='-----before %%@
change-----';

            tempthrmutual=thrmutualInfo1{ii,jj};
            le=length(tempthrmutual(1,:));
            if length(tempthrmutual(:,1))==2
                Message='tempthrmutual is no zero';
                flag=round(1+6*rand(1));
                % change random
                if flag==1 && tempindi(ii,jj)>-1
                    ranindex=round(1+(le-1)*rand(1));
                    mutatetimelag=tempthrmutual(1,ranindex);
                    count=count+1;
                    tempindi(ii,jj)=mutatetimelag;
                    output='after change';
                    tempindi(ii,jj);

                else

```

```

        if flag==1 && tempindi(ii,jj)==-1

            if nofp<maxfanin
                ranindex=round(1+(le-1)*rand(1));
                mutatetimelag
=tempthrmutual(1,ranindex);
                count=count+1;
                tempindi(ii,jj)=mutatetimelag;
                output='after change';
                tempindi(ii,jj);
                nofp=nofp+1;
            end

        end

    end

end

    end

end

    end
    % end loop for every gene pair mutaiton, mutate the time lag.

    newpopulation{mutatearray(i)}= tempindi;

end
% end Loop for individual in the population

end
% end of function

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% random mutation in genetic algorithm
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function randommutate()

```

```

global thrmutualInfo1;
global mutatenumbr;
global population;
global newpopulation;
global numberofgene;
global sizeofpopulation;
global maxfanin;
global maxtimelag;
% Loop for individual in the population
numberofpair=numberofgene*numberofgene;
mutatearray=randperm(sizeofpopulation);
mutatearray
=mutatearray(sizeofpopulation-mutatenumbr:sizeofpopulation);

for i=1: mutatenumbr

    tempindi=newpopulation{mutatearray(i)};

    % random generate a number of pair between lower and upper
    %   rannpair=lower + (upper-lower) *round( rand(1));
    count=0;
    % loop for every gene pair mutaiton, mutate the time lag.
    for ii=1: numberofgene
        nofp=0;

        for cc=1:numberofgene
            if tempindi(ii,cc)>-1
                nofp=nofp+1;
            end
        end

        for jj=1:numberofgene

            outout='-----before %%0
change-----';

            flag=round(1+6*rand(1));
            % change random
            if flag==1 && tempindi(ii,jj)>-1
                mutatetimelag

```

```

=round(1+(maxtimelag-1)*rand(1));
    count=count+1;
    tempindi(ii,jj)=mutatetimelag;
    output='after change';
    tempindi(ii,jj);

    else
        if flag==1 && tempindi(ii,jj)==-1

            if nofp<maxfanin
                mutatetimelag
=round(1+(maxtimelag-1)*rand(1));
                count=count+1;
                tempindi(ii,jj)=mutatetimelag;
                output='after change';
                tempindi(ii,jj);
                nofp=nofp+1;
            end

        end

    end

end

end

    end
end
% end loop for every gene pair mutaiton, mutate the time lag.

newpopulation{mutatearray(i)}= tempindi;

end

% end Loop for individual in the population

end

```

```

% end of function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The framework of proposed genetic algorithm
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% the genetic algorithm

% genetic algorithm to initial the population
clear all
clc
% this part is the variable used in genetic algorithm
data= load('9regulator.txt');
global numberofgene
numberofgene=length(data(:,1));

thrmutualInfo=load('thresholdmutual')
global sizeofpopulation;
sizeofpopulation=500;
% the first colum is presentation,
% the second colum is score
global population;
population=cell(sizeofpopulation,1) ;
global newpopulation
newpopulation=cell(sizeofpopulation,1);
IndexScore=[];
global maxfanin;
maxfanin=2;

ntime=length(data(1,:))
Maxscore=0;
Maxnumberofinteration=50;
threshold=150;
factnumberofiteration=0;
replacenumber=round(sizeofpopulation*0.7);
if rem(replacenumber,2)~=0
    replacenumber=replacenumber-1;
end
replacenumber
global remain
remain=sizeofpopulation-replacenumber
global mutatenumbr;
mutatenumbr=round(sizeofpopulation*0.3)

for noofp=1:sizeofpopulation
representation=-1* ones(numberofgene,numberofgene);

```

```

global thrmutualInfo1;
thrmutualInfo1=thrmutualInfo(1).mutualInfo3;
% generate the structure for gene one by one
for i=1:numberofgene

    % copy the non=empty cell to another
% temp array for every gene
    temp=[];
    count=1;
    for j=1:numberofgene
        if thrmutualInfo1{i,j}~=0

            temp{count}=thrmutualInfo1{i,j};

            temp{count}
=[temp{count};j*ones(1,length(temp{count}(1,:)))];
            count=count+1;
        end
    end
    temp;
    le=length(temp) ;
% incase the maxnumber is smaller than 4.
tempmax=min(le,maxfanin);

% generate a ramdon number between 1 to tempmax.[1,tempmax].
%
    nofrandomfanin=round(1+(tempmax-1)*rand);
% generate a purmutation between 1 to le of nofrandomfanin
    tempperm = randperm(le);
    parentsindex=tempperm(1:nofrandomfanin);
% the index is the potential parents in the temp,
% for each gene in the
% temp, you have to choose a random index for the time lag
    indextimelag=[];
    for in=1:length(parentsindex)
        tempcell=temp{parentsindex(in)};
        noftimelag=length(tempcell(1,:));
        indextimelag=[indextimelag round(1+(noftimelag-1)*rand)];
    end
    indextimelag;
% for now, you have the information of which
% is the gene in temp and the

```

```

% time lag in temp , you need to change it to 4 digits

for ind=1:length(parentsindex)
    finaltimelag
    =temp{parentsindex(ind)}(1,indextimelag(ind));
    finalgeneindex
    =temp{parentsindex(ind)}(3,indextimelag(ind));
    % finalgeneindex
    %=temp{parentsindex(ind)}(3,indextimelag(ind)) is the
    % mutual information

    % choose the gene
%    representation{i,finalgeneindex}(1)=1;

    % choose the timelag, using the case representation
    switch finaltimelag

        %case{0}
        %representation(i,finalgeneindex)=0;
        case{1}
            representation(i,finalgeneindex)=1;
        case{2}
            representation(i,finalgeneindex)=2;
        case{3}
            representation(i,finalgeneindex)=3;
        case{4}
            representation(i,finalgeneindex)=4;
        case{5}
            representation(i,finalgeneindex)=5;
        case{6}
            representation(i,finalgeneindex)=6;
        case{7}
            representation(i,finalgeneindex)=7;

    end

end

end

end % end of for for each gene

population{noofp}=representation;

```



```

% initial the score of the popuation.

%BICscore
% this part can be change to many other score
[tempBIC, tempLL, tempMDL]
=score(representation, data, ntime, numberofgene);
IndexScore(noofp,2)= tempLL;
IndexScore(noofp,1)=noofp;
% save('population5.mat','population')
end % ebd of for for each population
population;
IndexScore;
% end of initial population


% ranking the population
[IndexScore(:,2), Indexindex]
=sort(IndexScore(:,2), 'descend');
IndexScore(:,1)=IndexScore(Indexindex,1);
IndexScore;
Maxscore=IndexScore(1,2);
% caculate the selection probability, change it to
% 1-probability/(sizeofpopualation-1)
scoresum=sum(IndexScore(:,2));
for i=1:length(IndexScore(:,1))
    IndexScore(i,3)
    =(1-IndexScore(i,2)/scoresum)/(sizeofpopulation-1);
end
    IndexScore;

Maxscorearray(1)=Maxscore;


%while loop of the genetic algorithm
Nofinteration=0;
while Nofinteration<Maxnumberofinteration
    factnumberofiteration= factnumberofiteration+1;
    % probabilistic select population(replacerate)
    %to add to new population
    % in fact, I simplify it to remain the highest part.
    newpopulation(1:remain)

```

```

=population(IndexScore(1:remain,1));

% select population to crossover using uniform
%distribution(I simplify it, infact,
%should probabilisticly choose)
Swap(replacenumber)
% select population to mutate

lagmutate

randommutate

% replace the population
population=newpopulation;
newpopulation=cell(sizeofpopulation,1);

% compute new IndexScore for new population
IndexScore=[];
for ttt=1:sizeofpopulation
representation=population{ttt};
[tempBIC, tempLL,tempMDL]
=score(representation, data,ntime,numberofgene);
IndexScore(ttt,2)= tempLL;
IndexScore(ttt,1)=ttt;
IndexScore;
end

% ranking the population
[IndexScore(:,2),Indexindex]
=sort(IndexScore(:,2),'descend');
IndexScore(:,1)=IndexScore(Indexindex,1);
IndexScore;
newMaxscore=IndexScore(1,2)
difference=Maxscore-newMaxscore;
% if abs(difference)<0.0001
%     output='no diffence at'
%     factnumberofiteration
%     break;
% else difference<0

```

```

%      Maxscore=newMaxscore
% end

if difference<0
    Maxscore=newMaxscore
end

if Maxscore>threshold
    output='below threshold'
    factnumberofiteration
    break;
end
factnumberofiteration
Maxscorearray(factnumberofiteration)=newMaxscore;
Nofinteration=Nofinteration+1;
end
% end of while loop
Maxscore

a=(1:factnumberofiteration)
plot(a,Maxscorearray)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Output in text form
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
rowheaders
result=population{1};

diary resultLL.out

    sep='-----',
    Maxscore

diary off

for i=1:numberofgene
    message='';
    for j=1:numberofgene

        temp=result(i,j);
        if temp~-1

```

```

        message= strcat(rowheaders(j),'at time lag ');
        message= strcat(message,int2str(temp)) ;
        message= strcat(message, 'REGULATE :',rowheaders{i});
        diary resultLL.out

        message
        diary off
    end

end
diary resultLL.out

    ses='*****'
    diary off
end

```

VITA AUCTORIS

NAME: Zhengzheng Xing
PLACE OF BIRTH: Beijing, China
YEAR OF BIRTH: 1981
EDUCATION: Beijing Institute of Technology, Beijing, China
2000-2004
University of Windsor, Windsor, Ontario
2004-2006