2007

Integrate qualitative biological knowledge for gene regulatory network reconstruction with dynamic Bayesian networks

Song Li
Iowa State University

Integrate qualitative biological knowledge for gene regulatory network reconstruction with dynamic Bayesian networks

by

Song Li

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Science

Program of Study Committee:
Hui-Hsien Chou, Major Professor
Dimitris Margaritis
Oliver Eulenstein
Karin Dorman
Yanhai Yin

Iowa State University

Ames, Iowa

2007

DEDICATION

To my parents.

TABLE OF CONTENTS

LIST OF TABLES vi

LIST OF FIGURES vii

ACKNOWLEDGEMENTS ix

ABSTRACT x

CHAPTER 1. INTRODUCTION 1

1.1 Biological Background 1

1.1.1 Gene Expression 1

1.1.2 Gene Regulation 3

1.1.3 Microarray Experiment 4

1.2 Microarray Data Analysis 6

1.2.1 Clustering 6

1.2.2 GRN Reconstruction 8

1.2.3 Challenges 10

1.3 Overview 11

CHAPTER 2. GENE NETWORK RECONSTRUCTION 12

2.1 Review of the Literature on Gene Network Reconstruction 12

2.2 Bayesian Networks 15

2.2.1 Introduction 15

2.2.2 Dynamic Bayesian Networks 17

2.2.3 Advantages of using DBNs for GRN reconstruction 18

2.3 Review of BN structure learning 19

2.3.1 Known structure, full observability 19
2.3.2 Known structure, partial observability 20
2.3.3 Unknown structure, partial observability 20
2.3.4 Unknown structure, full observability 21

CHAPTER 3. METHODS AND PROCEDURES 26

3.1 Background 26
3.1.1 DBN structure learning 26
3.1.2 Data integration for learning gene network structure 27
3.1.3 Cross-correlation function 29
3.2 Novel model 31
3.3 Implementation 34
3.4 Parallel BN structure learning 37

CHAPTER 4. RESULTS 39

4.1 Data 39
4.2 Experiment Setting 40
4.3 Results 41
4.3.1 Accuracy 41
4.3.2 Consistency 45
4.3.3 Performance of the paralell implementation 45

CHAPTER 5. RECONSTRUCTING GENE NETWORKS USING SHORT-TIME CORRELATION 49

5.1 Background 49
5.2 Method 50
5.3 Experimental study 52
5.3.1 Data 52
5.3.2 Result 52

CHAPTER 6. VISUALIZATION 57

6.1 Introduction 57

6.2 Method 59

CHAPTER 7. CONCLUSION 62

APPENDIX A. THE DERIVATION OF VIOLATION FUNCTION 64

APPENDIX B. GENES AND REFERENCE NETWORKS OF TWO EXPERIMENTS 66

B.1 Experiment 1 (small) 66

B.2 Experiment 2 (large) 67

LIST OF TABLES

4.1	Summary of the microarray dataset published by [Spellman et al., 1998]	39
4.2	Performance comparison	43
4.3	Parameter selections	44
4.4	Contributors of the performance improvement	44
4.5	A comparison of standard deviations – The standard deviations of Random networks and Unsupervised method are justified regarding to the supervised method, the original standard deviation values are given in parentheses	46
4.6	Number of edges in 5 groups according to their occurrences	47
4.7	Running time comparison	48
5.1	minimum p-values and their associated time shift and window size (rows with bold font are those gene pairs whose minimum p-values are produced by sliding window shorter than the full length of the time series)	54

LIST OF FIGURES

1.1 Process whereby DNA encodes for the production of amino acids and proteins. 3

1.2 Regulation of Gene Expression 4

1.3 An Overview of Microarray Experiment [Duggan et al., 1999] 5

1.4 Hierarchical Clustering of Microarray Expression Data [Eisen et al., 1998] 8

2.1 From Experiments to Gene Network [Gardner and Faith, 2005] 12

2.2 Bayesian Networks - an example 16

2.3 A simple 2TBN with 8 variables 18

2.4 The high correlation between two genes' expression profiles may be explained by other genes 18

2.5 Multiple time slice can be "'rolled"' to a hyper-structure 19

2.6 Pseudo-code for hill-climbing. $nbd(G)$ is the neighborhood of G, i.e., the models that can be reached by applying a single local change operator. 23

3.1 Time shift between two series. Axis t denotes time, axis E denotes gene expression level. 30

3.2 Supervised learning for gene regulatory networks 31

3.3 Neighboring nodes. (A,B), (A,C) and (B,C) $\in R$, (A,D), (B,E) and (F,C) $\in nL(R)$ 33

3.4 System diagram. Components drawn as rectangles are our enhancements models inserted into the standard greedy search algorithm. . . . 37

3.5 Parallel BN structure learning 38

4.1 Reference Gene Network containing 25 genes and 30 interactions. 6 edges drawn in bold font: (SWI4, CDC42), (SWI4, CDC45), (SWI4, DEP1), (SWI4, PHO85), (PHO85, UME6) and (UME6, CHS6) form the training network 40
4.2 Weighted edge counts 47

5.1 Yeast cell cycle 49
5.2 Using sliding window for time-series microarray gene expression data . 50
5.3 Reference network. Every edge indicates an interaction reported by literature. 53
5.4 The network produced by standard DBN. (only takes microarray data as input), the dashed line is the biologically meaningful gene relationship with $bestPValue$ produced by partial sequence, however standard DBN method failed to predict it. 55
5.5 The network produced by our method (takes both microarray data and qualitative prior). The edge between "YPR119w" and "YOR372C" (with bold line) is missed in the standard DBN's prediction. 56

6.1 A comparison between 2-D and 3-D visualizations of the same metabolic pathway. (A) Pathway in 2-D space (source: www.kegg.org). (B) The same pathway in 3-D space. 58
6.2 UBViz's user interface 60

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Hui-Hsien Chou, for his guidance on my research through the years. Also thank other committee members who offered me insights during our discussion.

A part of the research effort reported in Chapter 5 is collaborated with Pan Du and Tian Xia, I enjoy discussing with them and appreciate the time they spent to teach me using the software they developed.

I greatly appreciate Dr. Brett Bode at Ames Laboratory of DOE who provided me graduate assistantship during the writing of this dissertation.

ABSTRACT

Reconstructing gene regulatory networks, especially the dynamic gene networks that reveal the temporal program of gene expression from microarray expression data, is essential in systems biology. To overcome the challenges posed by the noisy and under-sampled microarray data, developing data fusion methods to integrate legacy biological knowledge for gene network reconstruction is a promising direction. However, large amount of qualitative biological knowledge accumulated by previous research, albeit very valuable, has received less attention for reconstructing dynamic gene networks due to its incompatibility with the quantitative computational models.

In this dissertation, I introduce a novel method to fuse qualitative gene interaction information with quantitative microarray data under the Dynamic Bayesian Networks framework. This method extends the previous data integration methods by its capabilities of both utilizing qualitative biological knowledge by using Bayesian Networks without the involvement of human experts, and taking time-series data to produce dynamic gene networks. The experimental study shows that when compared with standard Dynamic Bayesian Networks method which only uses microarray data, our method excels by both accuracy and consistency.

CHAPTER 1. INTRODUCTION

1.1 Biological Background

The recent completion of genome sequencing projects of many species provides us with an unprecedented amount of genetic information, and thus shifted our focus from obtaining gene sequence data to identification of gene functions. The traditional methodology which separately studies basic units of information (or genes) is not sufficient since it is widely believed that biological systems are complex, where thousands of genes and their products interact in concert to enable life. Contrary to the traditional approaches, recently emerged functional genomics develops the genome-wide or system-wide experiments in hope of obtaining a global view of such a complex biological system.

An important objective of the functional genomics is to understand the controlling mechanism of the expression of these genes as well as the consequent synthesis of proteins, and ultimately encode the knowledge learned from the experiments into the format of a graph, namely gene regulatory network (GRN). To achieve this, computational and statistical tools are especially needed.

1.1.1 Gene Expression

The genetic information carried by an organism is primarily inscribed in deoxyribonucleic acid (DNA). DNA is helix-shaped molecule whose constituents are two parallel strands of nucleotides. There are four types of nucleotides in DNA denoted by letters A (for adenine), T (thymine), C (cytosine) and G (guanine). The two strands of DNA are reverse complementary, meaning that the second strand is always derivable from the first by pairing As with Ts and Cs with Gs and vice versa. Some contiguous pieces of DNA strand have been associated with

certain functions in the living organism, we name them "genes". Genes are the functional and physical unit of heredity [Bodenreider, 2004].

The process by which a gene's DNA sequence is converted into functional materials of a cell is usually referred to as gene expression. In most cases, the functional materials produced by gene expression are proteins. Protein is made by a linear sequence of amino acids, and the type of each amino acid is defined by three consecutive bases on DNA which is known as codon. For example, three bases "CGA" comprise a codon which would code for the amino acid *arginine*. Theoretically, there are $4^3 = 64$ types of amino acid, while in reality some combinations of three bases point to the same amino acid, leaving the number of amino acid types to be 20.

Protein is considered the most basic building block of life. Its roles include constituting cell structures, regulating cellular processes, catalyzing biochemical reactions in metabolic pathways, and many other functions. A protein's functions are determined by its particular physical structure and chemical properties. These properties in turn are determined by the particular sequence of 20 possible biologically-active amino acids as well as the exact manner the amino acid chain is folded into a three-dimensional structure. The very existence of life is made possible by thousands of different proteins acting at the right times and right places in a cell.

Figure 1.1 shows the gene expression process (source: http://www.accessexcellence.org/RC-/VL/GG/protein_synthesis.html). The first step is transcription of DNA, where mRNAs (messenger RNA) are made based on the information of the gene sequence. Translation occurs after the transcription of DNA to mRNA. The translation of mRNA into protein depends on adaptor molecules that recognize both an amino acid and a triplet of nucleotides. These adaptors consist of a set of small RNA molecules known as tRNA, each about 80 nucleotides in length. The ribosome is a complex of more than 50 different proteins associated with several structural rRNA molecules. rRNA is a machinery for synthesizing proteins by translating mRNA. Each ribosome is a large protein synthesizing machine, on which tRNA molecules position themselves for reading the genetic message encoded in an mRNA molecule.

The gene expression processes also depend on other factors not depicted in Figure 1.1, which

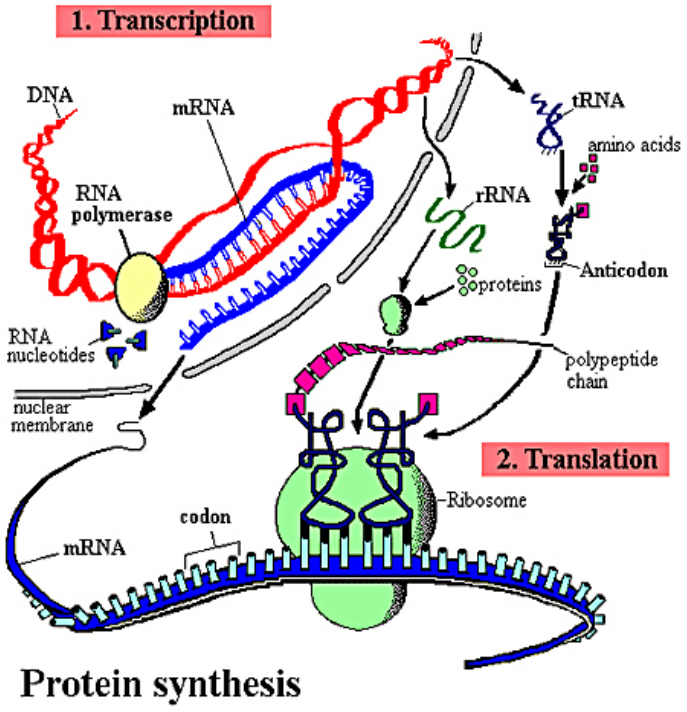

Figure 1.1 Process whereby DNA encodes for the production of amino acids and proteins.

include chromosomal activation or deactivation, control of transcription initiation, processing of RNA, RNA transport, mRNA degradation, initiation of translation and post-translational modifications.

1.1.2 Gene Regulation

A gene regulation system consists of genes, cis-elements, and regulators [Filkov, 2005]. Figure 1.2 illustrates the regulatory process of genes. In most cases the regulators are proteins, sometimes they also can be small molecules, such like RNAs and metabolites. The proteins that participate in regulatory system are usually called *transcription factors* (TFs), and sometimes they are also referred to as *trans-regulatory* elements. Cis-regulatory elements (or cis-elments in simple form), the complementary to trans-regulatory elements, are the DNA segments in the same strand of genes that control the expression of correspondent genes. The regulatory mechanism involves the binding of certain TFs to cis-elements in the cis-region of genes, and

consequently control the level of target gene's expression during transcription. A gene might be two-faced: its expression is regulated by some regulators, while its own expressed products can be regulators for other genes. The complex regulatory connections, together with an interpretation scheme form gene regulatory networks (GRNs).

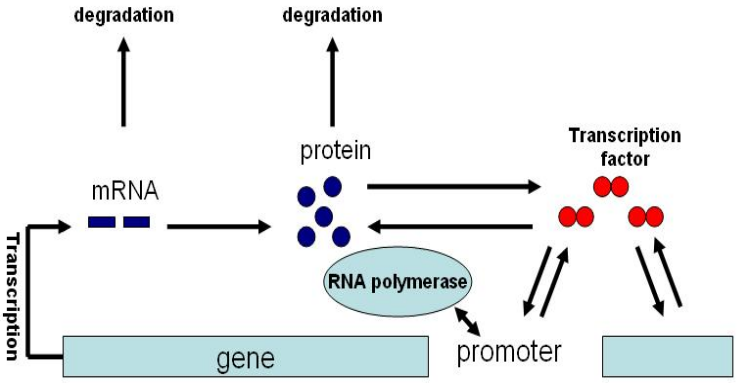

Figure 1.2 Regulation of Gene Expression

GRNs can be viewed as a cellular input-output device. A simple GRN would consist of one or more input signaling pathways, regulatory proteins that integrate the input signals, several target genes, and the RNA and proteins produced from those target genes. In addition, such networks often include dynamic feedback loops that provide for further regulation of network architecture and output.

1.1.3 Microarray Experiment

Traditional biology typically focuses on a single component in the living organism, e.g., a gene, a protein, or a reaction. Although there are remarkable achievements made by the research efforts guided by this methodology, traditional biology can hardly be adapted to the complex and integrated nature of real world biological processes, which make it difficult to explore the complex relationships among biological entities. For example, according to [Lockhart and Winzeler, 2000], both [Golub et al., 1999] and [Alizadeh et al., 2000] reported that monitoring a large number of genes is crucial for discovering the gene expression (mRNA) markers to classify several diseases. In [Golub et al., 1999] 50 genes were selected from more than 6,000 genes monitored on the arrays, the predictions based on the expression levels of

these 50 genes are by far more accurate than the prediction from any single gene. The results of both of these studies indicate that measurements with more individuals and more genes will be needed to identify robust expression markers that are predictive of clinical outcome. As a result, the high-throughput techniques that can simultaneously monitor gene activities are desired.

Microarray technology is one of such novel high-throughput tools that enable global analysis of genes and gene productions [DeRisi et al., 1997]. Among various types of microarray platform, here we use the popular DNA microarray to illustrate the general methodology. Figure 1.3 shows the process of microarray experiments.

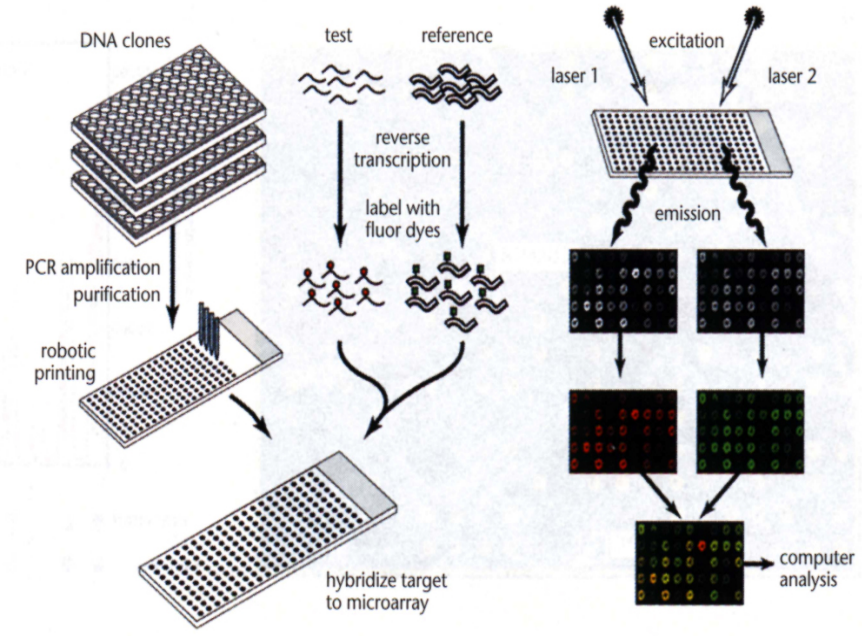

Figure 1.3 An Overview of Microarray Experiment [Duggan et al., 1999]

Microarray experiments usually require three phases. In the first phase, thousands of different "probes" are placed on the solid surface of a small chip, depending on the specific technology of the DNA microarray, the probe could be a cDNA sequence, or just a short DNA sequence which can exclusively match a part of the target mRNA sequence (called oligonucleotide). The chip is arranged as a rectangular 2-D grid of spots, and each of the probes is placed on a spot of the grid.

The second phase consists of pouring genetic material containing RNA which is obtained from the biological samples to be studied. These samples could be a pair of cancer tumor and normal organism, or cells being stimulated to starvation, high temperature, etc. For cDNA microarrays, two samples are needed, one for real test, another one is called reference sample which is used to be compared with the test sample. These two samples are labeled with different fluorochromes for the purpose of obtaining gene expression level in the next step: the test sample is labeled with the flurochrome Cy3, and the control sample is labeled with a different flurochrome Cy5. Because of the sequence similarity and complementariness the probes are likely to hybridize with their correspondent target but not the other RNA molecules. The amount of the hybridization products obtained is an indicator of how much RNA is being expressed by each one of the being studied.

The third phase consists of using a quantifying the amount of the hybridization products. By using laser scanner connected to a computer, fluorescence from array spots is detected and digitally imaged, then the intensity of of fluorescence from each spot is quantified. In cDNA microarrays, two independent images are generated for both samples (one only detects Cy3 and one only detects Cy5), the ratios of the measurement of the two samples is used in the subsequent data analysis.

Huge amount of data has been generated by microarray experiments, however, there is still a long way to go to transform the data into meaningful biological insight as data analysis methods are especially required in this process. In the next section we briefly review existing methods on microarray data analysis.

1.2 Microarray Data Analysis

1.2.1 Clustering

The large number of genes that are profiled in microarray experiment provides us with opportunities to a gain global view of the experiment results. Among various existing classes of data analysis methods, clustering is the most popular to date in this line. The key objective of the gene expression data clustering methods is to partition genes into groups based on given

expression profiles of each gene, so that the groups are homogeneous and well separated. It allows biologists to identify potentially meaningful relationships among genes within a cluster, or the relationships between clusters. Numerous literatures have reported successful uses of clustering methods on expression data in different aspect; for example, clustering was used to determine function for unknown genes [Eisen et al., 1998], to look at expression programs for different systems in the cell [Spellman et al., 1998] and for identifying sets of genes that are related to a certain type of cancer or other diseases [Alon et al., 1999].

An important technical component of clustering algorithms is the way to calculate the pairwise distance between two variables. The most commonly used distance measurements in the gene expression literature include Euclidean distance [Ewing et al., 1999], Pearson correlation [D'haeseleer et al., 1998], and mutual information [Butte and Kohane, 2000, Michaels et al., 1998]. With the distance function determined, the goal of a clustering algorithm is to achieve the minimum intra-cluster distances and the maximum inter-cluster distances.

Clustering algorithms can be divided into hierarchical and non-hierarchical methods. Hierarchical methods are named by their outcome which can be viewed as an multiple-layered hierarchy of nested clusters, and each cluster contains only a few sub-clusters. Hierarchical methods can be further categorized as agglomerative methods and divisive methods. Agglomerative methods take the bottom-up methodology, that is, start with forming small clusters with a few variables that are nearest in distance, and then recursively combine small clusters into bigger ones, until all clusters connect together. Divisive methods, on the other hand, go in reversed direction. All the variables are initially put into one cluster, divisive methods recursively divide it to smaller clusters. Figure 1.4 illustrates the general idea of Hierarchical clustering.

Although clustering is a mature technology in the area of statistics through decades of study, the recent availability of microarray data has sparked the development of multiple new methods. Many early works (e.g., [Michaels et al., 1998], [Eisen et al., 1998] and [Alon et al., 1999]) use hierarchical clustering, while more recent research focus more on non-hierarchical methods, for example, [Tamayo et al., 1999] and [Toronen et al., 1999] use SOMs-like methods,

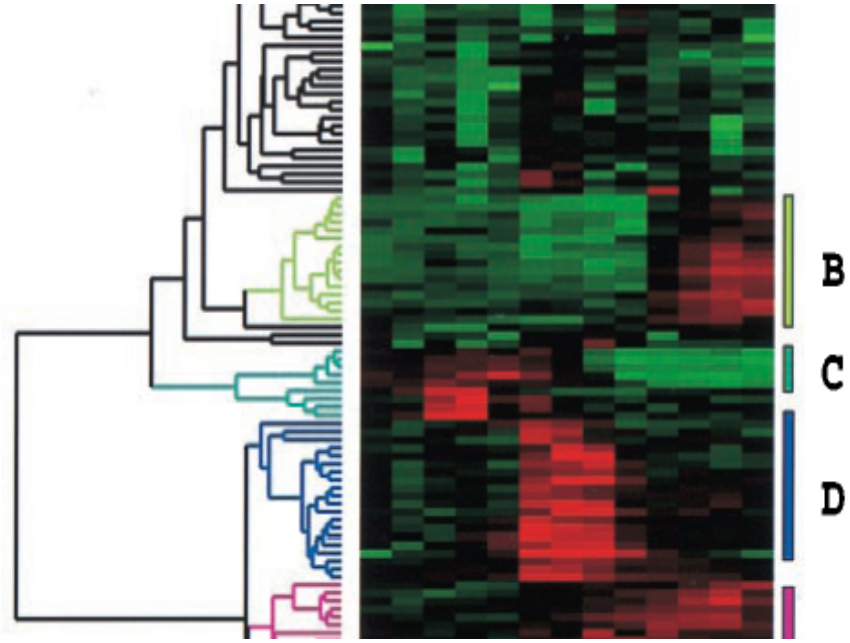

Figure 1.4 Hierarchical Clustering of Microarray Expression Data [Eisen et al., 1998]

[Ben-Dor et al., 2000] proposed a method with features with both agglomerative hierarchical clustering and K-means, and [Sasik et al., 2001] uses a Monte-Carlo like clustering method.

Non-hierarchical clustering methods normally cluster variables in an iterative process thus certain quantitative objective is optimized. Examples of non-hierarchical include K-means ([MacQueen, 1967]), Self-Organized Map (SOM) ([Kohonen, 1989]) and expectation-maximization (EM) ([Dempster et al., 1977]).

1.2.2 GRN Reconstruction

Clustering is a coarse-grained data analysis method of microarray expression profiles. Although it has successfully provided a global understanding of microarray experiments in previous works, there are several limitations that prevent clustering technology to be a tool for obtaining more detailed insight from the experimental results.

First, clustering is an unsupervised learning process, as a result it is difficult for clustering technologies to integrate the prior knowledge. However, using biological prior knowledge is very important for the success of transferring microarray data into real knowledge. In par-

ticular, the underlying assumption of clustering microarray data is that genes sharing similar expression profiles also share similar biological properties, this assumption may not always holds [Altman and Raychaudhuri, 2001]. Genes with similar expression patterns may not always share the same function or regulatory mechanism. Even when similar expression levels correspond to similar functions, the result of clustering cannot tell which gene inside a particular cluster has a certain function. To address these problems, further justifying the clustering result using existing biological knowledge is needed, and as reported in [Shatkay et al., 2000], is not trivial and requires a lot of effort.

Second, standard clustering algorithms group genes whose expression levels are similar across all conditions. However, a group of genes involved in the same biological process might only be co-expressed in a small subset of experimental conditions ([Carmona-Saez et al., 2006]). Furthermore, in the time-series microarray experiments, data points have a certain order and may not be independent of each others, genes may express at different times but serve complementing roles of one unifying function. Although several approaches have been developed to find genes with high similarity of a subset of expression profiles ([Cheng and Church, 2000], [Getz et al., 2000]), they are still limited to static microarray data thus cannot reveal temporal relationships among genes.

Third, clustering methods do not catch the conditional expression of genes and further discover the causal relationship between genes. Many genes can be conditionally co-expressed with different sets of genes, which may reflect the different biological roles that a gene product can play in the cell [Gasch and Eisen, 2002]. With the time-series microarray experiments, in addition to identifying genes with similar expression patterns, people also seek to infer causality among genes, that is, identifying the genes that play roles as regulators and the genes that they regulate. The coarse-grained clustering methods are not capable to give information in this detail.

1.2.3 Challenges

To address these limitations of clustering methods, a new branch of expression data analysis known as gene network reconstruction or "reverse-engineering" of GRNs has emerged.

Most early research works on microarray data analysis assume microarray experiments are mutually independent, ignoring the fact that real-world gene expression is a temporal process [Bar-Joseph, 2004], hence the expression levels at a certain time point may depend on the previous expression levels. To explore this temporal regulatory relationship among genes, time course microarray data are collected by conducting experiments across a number of time points [Spellman et al., 1998, Cho et al., 1998]. Recently, more works focus on modeling and analyzing the temporal aspects of gene expression data.

There are some inherent limitations of microarray data. First, microarray data are normally under-sampled: a typical microarray dataset contains thousands of genes, but only has at most a couple of hundred experiments. This makes microarray data statistically insufficient for reconstructing even moderate size gene networks. Second, as an high-throughput method, microarray experiments sacrifice specificity for scale in the quality to coverage trade-off, yielding too many false positives [Troyanskaya, 2005]. As a result, microarray data are typically very noisy, with the noisy data as the input, the outcome of computational methods is hardly reliable. Because of these limitations, microarray data alone is not enough to make accurate prediction of gene networks [Bar-Joseph, 2004].

On the other hand, there is a wealth of knowledge accumulated over decades by previous biology research, including protein-protein and protein-DNA interactions, transcription factor binding location data, phylogenetic profiles, ChIP-chip data, and even biological literature. Motivated by the challenges posed by microarray data, many recent research works try to integrate this additional information together with microarray data into the network reconstruction process. Several data integration methods in this field have been reported in the literature. Some are based on pairwise statistical analysis or mutual information [De Bie et al., 2005, Carmona-Saez et al., 2006, Margolin et al., 2006], while others are based on more complicate machine learning methods such as Bayesian Network [Bernard and Hartemink, 2005],

[Segal et al., 2003, Imoto et al., 2003, Li et al., 2006, Nariai et al., 2004] and Kernel Methods/SVM [Kato et al., 2005, Lanckriet et al., 2004, Yamanishi et al., 2004].

Some of the data generated by previous research is quantitative. For example, transcription factor binding location data has p-values to represent the quality of the data. It is natural to integrate the quantitative information into probabilistic learning models such as Bayesian Networks for gene network reconstruction. However, some other data such as gene ontology, domain expert's annotation and protein-protein interactions extracted from database cannot be represented by numbers. As reported in [Druzdzel and van der Gaag, 1995], a domain expert may be reluctant to provide numerical probabilities in a certain matter, rather they may simply give a statement like "more expression of gene A likely leads to less expression of gene B". these qualitative evidences, although not in a precise format, are still very valuable and worth the effort to make use of them. Unfortunately, there is not much previous research addressing the problem of automatically integrating qualitative biological knowledge for GRN structure learning.

1.3 Overview

In Chapter 2, we introduce gene network reconstruction with the focus on using Bayesian Networks and its extension Dynamic Bayesian Networks. In Chapter 3, we present our novel data integration method which can make use of qualitative biological knowledge using Dynamic Bayesian Networks, and show it is feasible to be parallelized. The experimental results and discussion of our method are included in Chapter 4. In chapter 5, we add short-time cross correlation into our method and show its potential to make better network structure prediction. In Chapter 6 we present a novel 3-D gene network visualization system.

CHAPTER 2. GENE NETWORK RECONSTRUCTION

2.1 Review of the Literature on Gene Network Reconstruction

Abstractly, inferring gene networks from biological experiments can be transformed to a reverse-engineering problem. In engineering disciplines, reverse-engineering methods are widely used to decode the design of a product such as a circuit, a software tool, etc., usually by analyzing the behaviors of the product with varied inputs. Within the context of gene network reconstruction, the biological "machine" is the product we want to study, while microarray data reflects its system outputs. Figure 2.1 shows a high level view of the problem we try to solve.

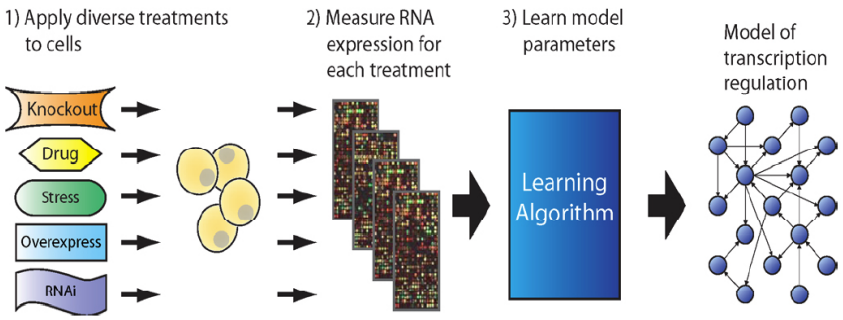

Figure 2.1 From Experiments to Gene Network [Gardner and Faith, 2005]

There are two major strategies for gene regulatory network reverse-engineering : the "physical approach" and the "influence approach" [Gardner and Faith, 2005]. The "physical approach" tries to use microarray expression data to identify the transcription factors (TFs) and their corresponding binding sites on DNA, thus the objective is to infer the true physical interactions between the TFs and the promoters of the regulated genes. The "influence approach", on the other hand, has a broader definition of gene regulation: it is not necessarily

carried by TFs and their binding site pairs, instead any "influence" among genes as reflected by expression data could be captured.

Physical strategy usually involves rate-law kinetics and ultimately models gene transcription as differential equations. It offers a relatively precise simulation of real-world transcription processes, for example, it can describe a repressive interaction by a gradually decreasing production rate, by using the differential equation we can calculate when will the mRNA concentration eventually decrease. The limitation of physical strategy is that it can only model the regulatory control based on transcription factors [Gardner and Faith, 2005].

My research follows the methodology of the "influence approach" which is not restricted to physical interactions. Previous works in this area can be further divided into four categories:

1. Association Networks

   The most straightforward way to reconstruct gene regulatory network is to test all pairwise statistical similarities of gene expression profiles; if a gene pair has a statistical similarities greater than a predefined threshold, we conclude that they are dependent, and consequently an edge is drawn between them in the resulted network. In [Gardner and Faith, 2005], this class of methods are referred to as Association Networks. A variety of similarity measurements have been adopted by previous works; for example, *Pearson Correlation* [de la Fuente et al., 2004], *Mutual Information* [Margolin et al., 2006], and the time-delayed cross-correlation coefficient of time-series microarray data [Schmitt et al., 2004].

2. Differential Equation models

   Gene network may be described as a system of differential equations. In general, let x_i denotes the expression level of gene i, the changing rate of x_i is given by a function f_i:

$$\frac{\mathrm{dx_i}}{\mathrm{dt}} = f_i(x_1, ..., x_n) \tag{2.1}$$

   where n is the number of all genes to be studied. Among many representations of f_i, previous researches suggest that linear function is the most successful concrete format

of f_i [D'haeseleer et al., 1999, Gardner et al., 2003]. Using a linear function to represent the changing rate, the differential equation is in the format of:

$$\frac{\mathrm{dx_i}}{\mathrm{dt}} = \sum_j w_{ij} x_j + p_i \tag{2.2}$$

where w_{ij} represents the influence of gene j on gene i. and p_i is an externally applied perturbation to the transcription level of gene i.

3. Boolean Network Models

In contrast to the differential equation models whose variables are continuous, Boolean network simplifies gene expression levels to binary variables. Under the Boolean network framework, a gene's expression level only has two states: being "turned on" or being "turned off". The state of a particular gene x_i at time $t+1$ is determined by a Boolean function f_i^B of all states at time t:

$$x_i(t+1) = f_i^B(x_1(t), ..., x_n(t)) \tag{2.3}$$

where $x_1(t), ..., x_n(t)$ are the states of all genes to be studied at time t.

The early works using Boolean Networks to reconstruct gene networks([Akutsu, 1999, Liang et al., 1998, Ideker et al., 2000]) tried to find a single best Boolean function for the given data, disregard the stochastic nature of biological phenomena. Later researches addressed this problem by introducing a new model called the Probabilistic Boolean Networks(PBNs) which combines the ideas of both deterministic Boolean Networks and probabilistic theory [Shmulevich et al., 2002]. Instead of associating gene i's state by a single Boolean function f_i^B, PBNs define a set of Boolean functions for a gene i, denoted as:

$$F_i = f_j^{(i)}, \quad j = 1, ..., l(i) \tag{2.4}$$

where $f_j^{(i)}$ is a possible Boolean function for gene i, $l(i)$ is the number of all possible functions in the set F_i. For each possible function, there is a probability associated with it to predict gene i's state. Thus, a PBN is defined by a list of function sets $F = F_1, ..., F_n$, and a matrix of probabilities with each element c_{ij} denoting the probability that the j^{th} function is used to define gene i's state.

2.2 Bayesian Networks

2.2.1 Introduction

My research focuses on using another kind of influence approach called Bayesian networks for GRN reconstruction. A Bayesian network represents the joint distribution over a set of random variables $\mathbf{X} = \{x_1, ..., x_N\}$ by directed acyclic graph (DAG). In this DAG, each node represents a random variable x_i, lack of edge between two nodes indicates that these two variables are conditionally independent. On the other hand, if a node x_i is connected, and let Pa_i denotes all of its parent nodes according to the direction of edges, there is a *Conditional Probability Distribution* (CPD) which can be represented as $\Pr(x_i|Pa_i)$.

Figure 2.2 shows a simple example of Bayesian network. Suppose one day your neighbor John calls to say the alarm in your house is ringing, but neighbor Mary doesn't call. Sometimes the alarm is set off by minor earthquakes. You want to know how likely that there was a burglar. Figure 2.2 describes the following relationships among these five variables: "Burglary", "Earthquake", "Alarm", "John Calls" and "'Mary Calls"':

- A burglar can trigger the alarm
- An earthquake can trigger the alarm
- The alarm can cause Mary to call
- The alarm can cause John to call

The knowledge is quantified as conditional probabilities. Consequently, in addition to the graph topology, another major component of Bayesian Networks is *conditional probability*

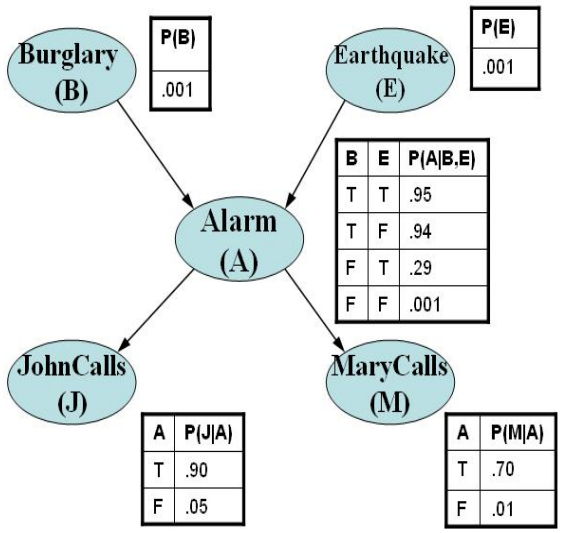

Figure 2.2 Bayesian Networks - an example

Distribution (CPD) (or *conditional probability Table* (CPT) if variables are discrete) associated with each variable. For example, the CPT in figure 2.2 tells us that if there was burglary, it will very likely set off the alarm; however, on the other hand, the alarm can be set off by earthquake with a much smaller probability.

Bayesian network is a compact representation of joint distribution, the compactness is achieved by factoring the joint distribution into local conditional distributions for each variable x_i given its parents Pa_i:

$$\Pr(x_1, ..., x_n) = \prod_i \Pr(x_i | Pa_i) \tag{2.5}$$

To describe the joint distribution over N variables, we need to store the probability of every possible event as defined by the values of all the variables. There are exponentially many such events, therefore the space complexity is $O(2^N)$. In BN representation, if the maximum number of parents is denoted as k, it is easy to see that the space complexity of Bayesian network is $O(2^k \cdot N)$. Since k is usually a much smaller number than N, the space requirement of Bayesian networks is much lower than the method which exhaustively enumerates all the possible events.

2.2.2 Dynamic Bayesian Networks

In many BN applications the conditional dependency among variables may change over time. Dynamic Bayesian Networks (DBNs) [Dean and Kanazawa, 1990] was proposed to express certain transient features of BNs, here the term "dynamic" means we are modeling a dynamic system, not that the network changes over time.

DBNs can be viewed as an extension of static BNs by introducing the concept of time slice. In each time slice t, there are a fixed number N of variables $\mathbf{X}_{ti}$, where $i = 1, ..., N$, that are only dependent on the variables in previous time slices $1, ..., t-1$. Assume t ranges from 1 to T, there are totally $T \times N$ variables in the DBN instead of N variables in its static BN counterpart. For the sake of simplicity, we assume variables in time slice t are only dependent on the variables in the immediately preceding time slice $t-1$(i.e., the system is first-order Markov), and the transition feature between time slices does not change. Hence two time slices are enough to express the whole dynamic system.

Formally, a DBN is defined to be a pair, $(B_1, B_{\rightarrow})$, where B_1 is a BN which defines the prior joint distribution, and $B_{\rightarrow}$ is a two-slice temporal BN (2TBN) which defines $P(X_t|X_{t-1})$ by means of a DAG (directed acyclic graph) as follows:

$$P(X_t|X_{t-1}) = \prod_{i=1}^{N} P(X_t^i|Pa(X_t^i))$$

where X_t^i is the i'th node at time t, and $Pa(Z_t^i)$ are the parents of Z_t^i in the graph. The nodes in the first slice of a 2TBN do not have any parameters associated with them, but each node in the second slice of the 2TBN has an associated conditional probability distribution (CPD), which defines $P(X_t^i|Pa(X_t^i))$ for all $t > 1$. Figure 2.3 illustrates a simple 2TBN.

DBN is a generalized version of Hidden Markov Model (HMM) ([Rabiner, 1989]) and Kalman Filter Model (KFM) ([Roweis and Ghahramani, 1999, Minka, 1999]). The difference between a DBN and a HMM is that a DBN represents the hidden state in terms of a set of random variables, $X_1^t, \ldots, X_{N_h}^t$, i.e., it uses a distributed representation of state. By contrast, in an HMM, the state space consists of a single random variable X_t. The difference between a

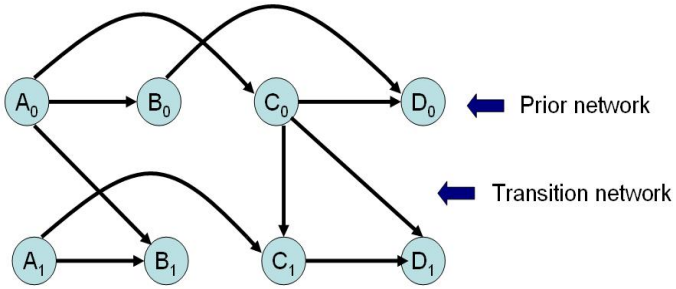

Figure 2.3 A simple 2TBN with 8 variables

DBN and a KFM is that a KFM requires all the CPDs to be linear-Gaussian, whereas a DBN allows arbitrary CPDs. In addition, HMMs and KFMs have a restricted topology, whereas a DBN allows much more general graph structures.

2.2.3 Advantages of using DBNs for GRN reconstruction

Traditional GRN reconstruction methods are based on pairwise correlations. However, as Figure 2.4 shows, pairwise correlation cannot be used to rigorously prove or disprove that two genes are related since two genes could be co-regulated by another gene or a subset of genes. BN-based methods, on the other hand, reflect the combinatorial logic of all genes in the system and can be used to find true relationships by given a larger context.

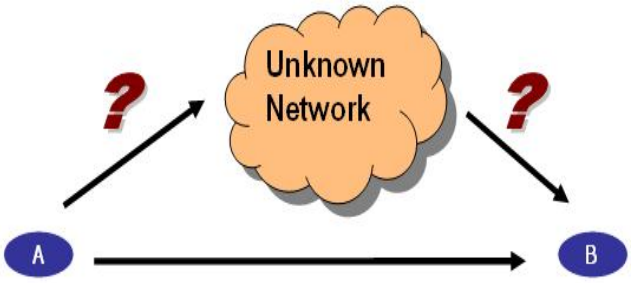

Figure 2.4 The high correlation between two genes' expression profiles may be explained by other genes

DBN is also an intuitive way for modeling gene regulatory networks. First, the directed edges in BNs have advantages since they can be interpreted as reflecting causal relationships [Murphy and Mian, 1999]. In biological data analysis, uncovering the causal relationships such as "the activity of gene A causes the activity of gene B" is more desirable than the non-causal

correlations.

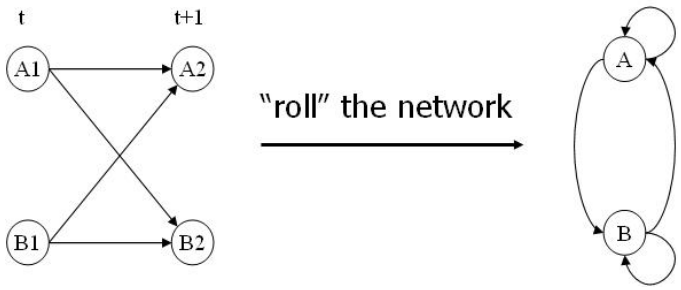

Figure 2.5 Multiple time slice can be "'rolled"' to a hyper-structure

Second, unlike static BNs that are acyclic, as Figure 2.5 shows rolled DBNs are cyclic. It is known that regulatory loops are commonly present in the biological process, thus the outcome of DBNs can better simulate gene regulatory mechanism.

2.3 Review of BN structure learning

Learning process in BN can be either parameter learning or structure learning. For parameter learning, the topology of the target BN is fixed, the task is to estimate the CPTs or CPDs for every node in the network. On the other hand, if the topology is unknown, structure learning is required to learn the graph topology of the target BN before the parameters could be determined. Also the observation data used for BN learning may be either complete or incomplete, based on these varieties, [Murphy, 2002] categorized four scenarios of learning structure of BNs.

2.3.1 Known structure, full observability

The goal of learning in this case is to find the the parameter values which maximize the likelihood of the training data. The log-likelihood of the training set $D = \{D_1, \ldots, D_M\}$ is a sum of the likelihood of each node by given the observational data D_m, network structure G and its parent node $Pa(X_i)$:

$$L = \log \prod_{m=1}^{M} \Pr(D_m|G) = \sum_{i=1}^{n} \sum_{m=1}^{M} \log P(X_i|Pa(X_i), D_m) \tag{2.6}$$

Next the task is to specify how to estimate the parameters of each type of CPD given its local data $\{D_m(X_i, Pa(X_i))\}$ by various supervised learning methods. Interested readers can refer to [Murphy, 2002] for more discussion.

2.3.2 Known structure, partial observability

In the partially observable case, the log-likelihood is

$$\begin{aligned} L &= \sum_m \log P(D_m) \\ &= \sum_m \log \sum_h P(H = h, V = D_m) \end{aligned} \tag{2.7}$$

where the innermost sum is over all the assignments to the hidden nodes H, and $V = D_m$ means the visible nodes take on the values specified by case D_m. Unlike the fully observed case, the log-likelihood in this case cannot be decomposed into a sum of local terms. The output of parameter estimation would be a distribution over the parameters. The most common method for achieving this is to use Gibbs sampling [Geman and Geman, 1984, Gilks et al., 1996].

2.3.3 Unknown structure, partial observability

In the partially observable case, computing the marginal likelihood is intractable, requiring that we sum out all the latent variables Z as well as integrate out all the parameters θ:

$$P(X|G) = \sum_Z \int_\theta P(X, Z|G, \theta) P(\theta|G) \tag{2.8}$$

This score is hard to compute, and does not decompose into a product of local terms. Interested readers can refer to [Murphy, 2002] for more discussion on how to estimate the the approximated distribution of $P(X|G)$.

2.3.4 Unknown structure, full observability

In this dissertation we focus on the scenario where we want to learn the BN structure for complete observational data. Next we introduce the most popular approach based on three critical issues about the BN structure learning:

- What is the hypothesis space? This could be the space of DAGs, equivalence classes of DAGs (PDAGs), undirected graphs, trees, node orderings, etc.
- What is the evaluation (scoring) function?
- What is the search algorithm? This could be either local search (e.g., greedy hill climbing, possibly with multiple restarts) or global search (e.g., simulated annealing or genetic algorithms).

2.3.4.1 Search space

DAGs The most common approach is to search through the space of DAGs. Unfortunately, the number of DAGs on N variables is $2^{O(N^2 \log N)}$ [Robinson, 1973], [Friedman and Koller, 2000]. For example, there are 543 DAGs on 4 nodes, and $O(10^{18})$ DAGs on 10 nodes. This means that attempts to find the DAG which can best fit the data is practically impossible.

PDAGs One way to reduce the search space is to use PDAG (partially directed acyclic graph) which represents a whole class of Markov equivalent DAGs. Two DAGs are Markov equivalent if they imply the same set of conditional independencies. For example, $X \rightarrow Y \rightarrow Z$, $X \leftarrow Y \rightarrow Z$ and $X \leftarrow Y \leftarrow Z$ are Markov equivalent, since they all represent $X \perp Z|Y$. In general, two graphs are Markov equivalent if and only if they have the same structure (without considering directions of edges), and the same v-structures [Verma and Pearl, 1990]. (A v-structure consists of converging directed edges into the same node, such as $X \rightarrow Y \leftarrow Z$.) According to [Cooper and Yoo, 1999, Pearl, 2000], we cannot distinguish members of the same

Markov equivalence class if we only have observational data, therefore to search in the space of PDAGs can reduce the complexity without loosing accuracy.

2.3.4.2 Variable orderings

Another factor that may influence the search space in BN structure learning is variable ordering. [Buntine, 1991] argues that by given a total ordering $\prec$, the likelihood decomposes into a product of terms, since the parents for each node can be chosen independently.

$$\begin{aligned} P(D|\prec) &= \sum_{G\in\mathcal{G}_\prec}\prod_{i=1}^{n} score(X_i, Pa_G(X_i)|D) \\ &= \prod_i \sum_{U\in\mathcal{U}_{\prec,i}} score(X_i, U|D) \end{aligned} \tag{2.9}$$

$\mathcal{G}_\prec$ is the set of graphs consistent with the ordering $\prec$, and $\mathcal{U}_{\prec,i}$ is the set of legal parents for node i consistent with $\prec$. If we bound the number of parents by k, each summation in Equation 2.9 takes $\binom{n}{k} \leq n^k$ time to compute, so the whole equation takes $O(n^{k+1})$ time.

Given an ordering, we can the find the best DAG consistent with that ordering using greedy selection like the K2 algorithm [Cooper and Herskovits, 1992] does, or more sophisticated variable selection methods. If the ordering is unknown, we can search for it by using MCMC [Friedman and Koller, 2000]. However, the space of orderings has size $N!$, which is still huge.

2.3.4.3 Search algorithm

The hill-climbing algorithm There are two kinds of search algorithms for the best BN structure: local search and global search. For local search, first randomly select a graph as the starting point, then apply an operator which modifies this graph. The operators that move through space are usually adding, deleting or reversing a single arc; this defines the neighborhood of a graph, $nbd(G)$. As figure. 2.6 shows, the hill-climbing algorithm moves to the neighbor with the best score, and continue this process until the score cannot be improved.

```
Choose G somehow
While not converged
    For each G' in nbd(G)
        Compute score(G')
    G* := arg max_{G'} score(G')
    If score(G*) > score(G)
        then G := G*
        else converged := true
```

Figure 2.6 Pseudo-code for hill-climbing. $nbd(G)$ is the neighborhood of G, i.e., the models that can be reached by applying a single local change operator.

Obviously, the hill-climbing algorithm is too aggressive hence may easily be trapped in a local minima/maxima. Global search algorithm, on the other hand, attempt to find the global minima/maxima. In next section we briefly outline one global search algorithm called PC algorithm.

The PC algorithm For global search, one popular algorithm is called PC algorithm [Spirtes et al., 2000, Pearl and Verma, 1991, Pearl, 2000, p84] which can find the globally optimal PDAG in $O(N^{k+1}N_{train})$ time, where there are N nodes, N_{train} data cases, and each node has at most k neighbors. The algorithm works as follows: start with a fully connected undirected graph, and remove an arc between X and Y if there is some set of nodes S satisfying $X \perp Y|S$; at the end, we can orient some of the undirected edges, so that we recover all the v-structures in the PDAG.

The PC algorithm will provably recover the generating PDAG if the conditional independence (CI) tests are all correct. For continuous data, we can implement the CI test using Fisher's z test; for discrete data, we can use a χ^2 test [Spirtes et al., 2000, p95]. A major drawback of the PC algorithm is that testing if $X \perp Y|S$ for discrete random variables requires creating a table with $O(K^{|S|+2})$ entries, which requires a lot of time and samples.

2.3.4.4 Scoring function

By Bayes' rule, the MAP model is the one that maximizes

$$\Pr(G|D) = \frac{\Pr(D|G)\Pr(G)}{\Pr(D)}$$

where $P(D)$ is a constant independent of the model. Consequently:

$$P(D|G) = \int_{\theta} P(D|G,\theta)P(\theta|G)$$

Then we can combine the marginal likelihood with a structural prior to get:

$$score(G) \stackrel{\text{def}}{=} P(D|G)P(G)$$

If we assume all the parameters are independent, the marginal likelihood decomposes into a product of local terms, one per node:

$$\begin{aligned} P(D|G) &= \prod_{i=1}^{n} \int_{\theta_i} P(X_i|Pa(X_i),\theta_i)P(\theta_i) \\ &\stackrel{\text{def}}{=} \prod_{i=1}^{n} score(Pa(X_i), X_i) \end{aligned}$$

Under the assumptions of global and local parameter independence each of these integrals can be performed in closed form, so the marginal likelihood can be computed very efficiently. If the priors are not conjugate, one can try to approximate the marginal likelihood. For example, [Heckerman, 1998] shows that the approximation to the parameter posterior has the form

$$\log \Pr(D|G) \approx \log \Pr(D|G,\hat{\theta}_G) - \frac{d}{2}\log M$$

where M is the number of samples, $\hat{\theta}_G$ is the ML estimate of the parameters and d is the dimension (number of free parameters) of the model. This is called the Bayesian Information Criterion (BIC), and is equivalent to the Minimum Description Length (MDL) approach. The first term is just the likelihood and the second term is a penalty for model complexity.

The BIC score also decomposes into a product of local terms:

$$\begin{aligned} \text{BIC-score}(G) &= \sum_i \sum_m \log P(X_i | Pa(X_i), \hat{\theta}_i, D_m) - \frac{d_i}{2} \log M \\ &= \sum_i \sum_{jk} N_{ijk} \log \theta_{ijk} - \frac{d_i}{2} \log M \end{aligned} \tag{2.10}$$

where $d_i = q_i(r_i - 1)$ is the number of parameters in X_i's CPT.

CHAPTER 3. METHODS AND PROCEDURES

3.1 Background

3.1.1 DBN structure learning

With the observational data in hand (in our case the microarray data), our goal is to find a DBN structure which most likely generates this data. Constructing BNs without the prior knowledge of the network structure is generally a hard problem which has stimulated considerable research effort, beside a brief review of related literature in the Chapter 2 of this dissertation, much details on this topic can also be found in [Friedman et al., 1998, Heckerman, 1995]. Here we rephrase the commonly used structure learning approach. Our task is to find a network structure G which maximizes the likelihood by given the observational data D, we denote it as scoring function $\Pr(G|D)$, using Bayes' Theorem, we have:

$$\log \Pr(G|D) = \log \Pr(D|G) + \log \Pr(G) + c \tag{3.1}$$

where $c = \Pr(D)$ is a constant and can be ignored, $\Pr(D|G)$ is the likelihood that G generates D, and $\Pr(G)$ is commonly referred to as structural prior.

The term $\log \Pr(D|G)$ can be further approximated by Bayesian Information Criterion(BIC) [Schwarz, 1978], which is defined as:

$$\log \Pr(D|G) \approx \log \Pr(D|G, \hat{\Theta}_G) - \frac{d}{2} \log N \tag{3.2}$$

where $\hat{\Theta}_G$ denotes the estimated parameters by Maximum Likelihood (ML) method. Since more complex network structures tend to have greater $\log \Pr(D|G)$ values, the term $\frac{d}{2} \log N$ is

used to penalize the over-complex structures. Here N is the number of samples, and d denotes the dimension of the model. In the rest of this chapter, we refer the likelihood $\Pr(G|D)$ computed by equations (3.1) and (3.2) as BIC score.

3.1.2 Data integration for learning gene network structure

3.1.2.1 Integrate quantitative biological knowledge

If there is no prior knowledge on any structure G, the second term $\Pr(G)$ in equation (3.1) is taken unbiased, that is, every G has the same likelihood thus the score in this situation is equivalent to $\Pr(D|G)$. On the other hand, if the network structure prior is present, $\Pr(G)$ can be used to encode it and thus affect the structure learning process. There are different ways to incorporate the structure prior in terms of $\Pr(G)$. If the prior knowledge has a quantitative form, it can be naturally expressed in $\Pr(G)$; for example, the work reported in [Bernard and Hartemink, 2005] make $\Pr(G)$ to be a continuous function of p-values of transcription factor binding locations, their method can be briefly described as follows.

The evidence of regulatory relationship between a transcription factor and genes is reported as a p-value, the smaller the p-value, the more likely the edge is to exist in the true structure. First let us denote the true gene network structure as S, and define the p-value for the location data corresponding to edge E_i in terms of random variable P_i defined on the interval [0,1]. In this interval, P_i has been previously assumed to be exponentially distributed ([Segal et al., 2001]) if the edge E_i is present in S, and uniformly distributed if the edge E_i is absent from S (by the definition of a p-value). Formally, we have

$$\Pr_{\lambda}(P_i = p|E_i \in S) = \frac{\lambda e^{-\lambda p}}{1 - e^{-\lambda}} \tag{3.3}$$

where λ is the parameter controlling the scale of the truncated exponential distribution. Let β denote $Pr(E_i \in S)$, by using bayes rule, the probability that edge E_i is present after observing the corresponding p-value is:

$$\Pr_{\lambda}(E_i \in S|P_i = p) = \frac{\lambda e^{-\lambda p}\beta}{\lambda e^{-\lambda p}\beta + (1 - e^{-\lambda})(1 - \beta)} \tag{3.4}$$

Hence we can compute the cut-off p-value p^* by solving the equation $\Pr_\lambda(E_i \in S|P_i = p^*) = \Pr_\lambda(E_i \notin S|P_i = p^*)$. The result is:

$$p^* = \frac{-1}{\lambda} \log(\frac{(1-e^{-\lambda})(1-\beta)}{\lambda\beta}) \tag{3.5}$$

For any fixed value of λ, if the p-value is below the critical p-value p^*, then the corresponding edge is more like to present.

Obviously, this method depends on the availability of p-values, which can not be provided by the qualitative evidences as we have discussed above.

3.1.2.2 Integrate qualitative biological knowledge

Some other biological evidences are not quantitative in nature, to make use of them, one straightforward way is to manually convert them to numbers beforehand. Some examples using manual quantification include [Imoto et al., 2003] and [Nariai et al., 2004]). Under the framework of BN, we can let domain experts assign the values of $\Pr(G)$ for several favored structures based on their knowledge, and leave the other structures with $\Pr(G) = 0$.

The applicability of this approach is limited by the availability of domain experts, and the credibility of their judgment. In fact, even domain experts may not be able to quantify their own knowledge [Druzdzel and van der Gaag, 1995]. One previous research effort that seeks automatically integrate qualitative biological data is reported in [Kato et al., 2005], their major idea is summarized below.

To simplify the notation we assume there are one major adjacency matrix $\mathbf{Q}$ and one auxiliary adjacency matrix $\mathbf{P}$. $\mathbf{Q}$ can be further divided as:

$$Q = \begin{pmatrix} K_I & Q_{vh} \\ Q_{vh}^T & Q_{hh} \end{pmatrix}$$

where K_I is determined by qualitative evidences (or training network in terms of machine learning) using the diffusion kernel [Kondor and Lafferty, 2002]. $\mathbf{P}$ is derived from biological data, e.g. microarray data. Our goal is to predict the sub-matrix Q_{vh} and Q_{hh} which represents

the unknown part of network.

Next the Kullback-Leibler (KL) distance between **P** and **Q** is calculated as follows:

$$D[Q,P] = \frac{1}{2} trP^{-1}Q + \frac{1}{2} log \det P - \frac{1}{2} log \det Q - \frac{1}{2} l \tag{3.6}$$

where l is the number of nodes. Then the task of estimating Q_{vh} and Q_{hh} becomes to an optimization problem:

$$min_{Q_{vh},Q_{hh}} D[Q,P] \tag{3.7}$$

[Kato et al., 2005] offered an approximate solution to solve this problem based on expectation-maximization (EM) [Dempster et al., 1977].

The method introduced by [Kato et al., 2005] is kernel-based and inherently does not consider the direction of the edges in the constructed network. [Wittig and Jameson, 2000] proposed a method based on BN and addressed the data integration problem by introducing the concept of "violation function". Intuitively, if given a set of qualitative assertions such as "if variable A increases, then variable B increases", the violation function sums up the occurrences of all violations in the observational data against these qualitative assertions, and assigns it to $\Pr(G)$ to penalize the structures with more violations. Interested readers can refer to Appendix A of this dissertation for a summary of rigorous definition of violation function.

Our method adopts the methodology of violation function which seeks the automatic and objective quantitative representation of qualitative evidences from data. The work in [Wittig and Jameson, 2000] simply counts all the violations of a predefined monotonic trend, therefore it can only work for static BNs. We extend it by treating variables as time-series signals that may fluctuate over time, hence our method is suitable to be integrated into DBN framework.

3.1.3 Cross-correlation function

It is possible that the influences among genes may take effect after a certain amount of time. Consider gene A and gene B's expression intensity in Figure 3.1. Although there is not

much similarity of their expression intensity at each time point, if we shift one of them along the time axis, we can make two contours almost perfectly match, therefore we can say that these two genes are strongly correlated with a certain time lag. Moreover, the direction of this time shift can reveal the causal relationship in addition to the correlation. For example, because we need to move gene B toward left to make the best matching, data shown in figure 3.1 is a good evidence that gene A regulates gene B, but not vice versa.

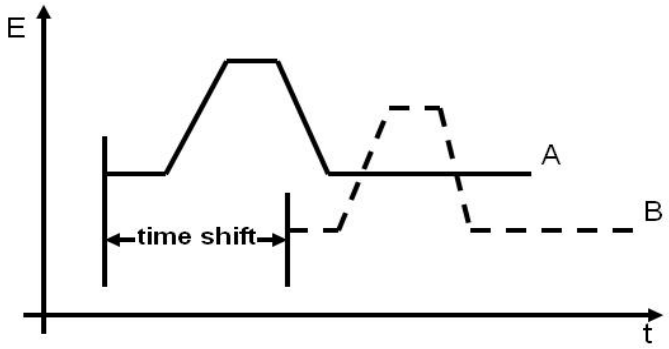

Figure 3.1 Time shift between two series. Axis t denotes time, axis E denotes gene expression level.

To numerically capture the causal relationships involving time shift, we propose using cross-correlation function(CCF) to assess the correlation of two gene expression profiles in time-series microarray data. CCF is a standard time-series analysis tool which is widely used in digital signal processing [Jenkins and Watts, 1969]. It has also been successfully applied in time-series microarray data analysis [Schmitt et al., 2004]. Taking the time lag between two time series as input, a CCF quantifies the similarity between these two time series. Formally, assume the experiments were taken at a series of time points t_i, where $i = 1, ..., n$, and the intervals between two successive time points are fixed, we define this unique interval as $\Delta t = t_i - t_{i-1}$. The definition of the CCF of two time series X and Y is:

$$CCF(X, Y, \tau_k) = \frac{\frac{1}{N}\sum_{i=1}^{N-k}(x_i - \bar{x})(y_{i+k} - \bar{y})}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2} \cdot \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2}} \tag{3.8}$$

where x_i and y_i are the i^{th} element of series X and Y, $\tau_k = k \cdot \Delta t$ is the time lag, $\bar{x}$ and $\bar{y}$ are the means of X and Y, and N is the total number of time points in the time series. In practice, tl should be a small number comparing to the length of the time series N, since

standard CCF sums up shifted difference $x_i - \bar{x}$ and $y_{i+k} - \bar{y}$ for $N - k$ times, a small $N - k$ value makes the CCF value small and statistically insufficient.

In our specific application domain, all of the time series are precisely synchronized since a single microarray experiment measures every gene's expression level simultaneously. As a result, the concrete value of time interval Δt is not critical in the correlation analysis. For the sake of simplicity we ignore Δt, thus in our computation τ_k is replaced by the indexing number k.

3.2 Novel model

In this research we are specifically interested in integrating qualitative information in probabilistic models to learn gene network. In particular, we formalize our method as a supervised learner as shown in Figure 3.2. In machine learning, unsupervised learning is a method where a model is fit to observations, there is no *a priori* outputs. On the other hand, Supervised learning tries to create a model from training data. The training data consist of pairs of input objects, and desired outputs. The output of the function can be a continuous value (called regression), or can predict a class label of the input object (called classification). The task of the supervised learner is to predict the value of the function for any valid input object after having seen a number of training examples.

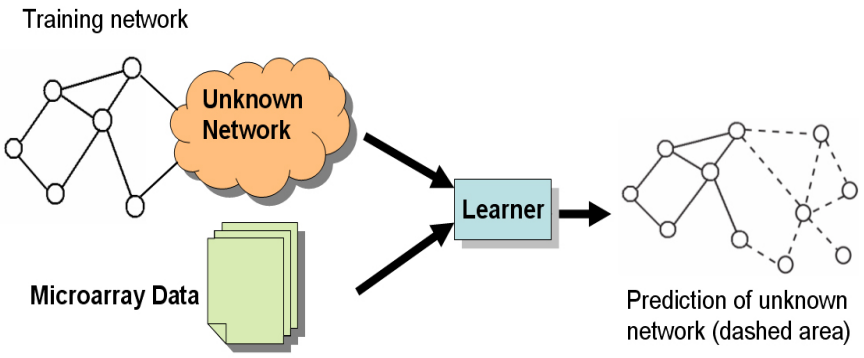

Figure 3.2 Supervised learning for gene regulatory networks

[0]Our novel model together with the experimental results are also reported in [Li, 2007] "Integrate qualitative biological knowledge to build gene networks by parallel dynamic bayesian network structure learning. In Proceedings of IEEE 7th International Symposium on Bioinformatics & Bioengineering (BIBE 2007)".

The gene interactions from the previous biological knowledge constitute the training network, which is considered correct. Combining it with the microarray data, we seek to achieve a better prediction of the rest of the network. To our knowledge, this work is the first attempt to integrate qualitative evidence by DBNs.

First we make two additional operations based on the standard CCF so that it can be used in the DBNs structure learning. First, we only consider the magnitude of the correlation among genes to score a DBN structure since the polarity of the causal effects is not a factor in structure scoring. That is, if there exists an edge $A \rightarrow B$ in a BN structure, it could reflect either positive or negative regulatory pattern from gene A to gene B. Second, different values of time lag parameter k lead to different CCF results, but it is the best value which represents the similarity of two series. Since which k value produces the best CCF value is not known *a priori*, we need to perform a search process for a best k value within a range. Combining these two new operations, we define a function $bestCCF$ as:

$$bestCCF(X, Y, tl) = \max(abs(CCF(X, Y, k))), \quad k = 0, ..., tl \tag{3.9}$$

where tl is a predefined maximum time lag, $bestCCF$ tries the time lags from zero to tl and output the best CCF value.

The qualitative evidences used by our method can be viewed as a set of edges, so we denote this set as R. We further define another set $nL(R)$ which contains all the edges outside R that connect to any node in R. For example, consider the 6-node structure in figure 3.3, and suppose edges drawn with solid lines agree with the qualitative prior knowledge set R, that is, edges (A,B), (A,C) and (B,C) $\in R$. The nodes in R, consequently, are A, B, C. Next consider edges (A,D), (B,E) and (F,C) drawn with dashed lines, they are not in R but each of them connects to a node in R, therefore, $nL(R) = \{(A, D), (B, E), (F, C)\}$.

With these definitions, we propose a scoring function for an individual edge (X,Y) as follow:

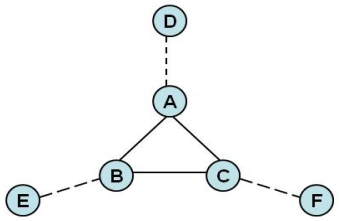

Figure 3.3 Neighboring nodes. (A,B), (A,C) and (B,C) $\in R$, (A,D), (B,E) and (F,C) $\in nL(R)$

$$edgeScore(X,Y) = \begin{cases} \alpha \cdot bestCCF(X,Y,tl), & (X,Y) \in R \\ \beta \cdot bestCCF(X,Y,tl), & (X,Y) \in nL(R) \\ -\gamma \cdot bestCCF(X,Y,tl), & otherwise \end{cases} \tag{3.10}$$

α, β and γ are positive constants that define the weight of the $bestCCF(X,Y)$ in our new scoring scheme. The idea of *edgeScore* is to award or penalize an edge (X,Y) according to the prior knowledge R and $nL(R)$. If (X,Y) is part of the prior knowledge set R, its existence is directly supported by the prior knowledge. Since it is well known that gene networks are "small world" networks, thus a node is more likely connecting to some nodes that are already connected. Therefore, if (X,Y) connects to R, it also can be considered supported by the prior knowledge indirectly. On the other hand, if the first two cases are not met, then (X,Y) is not supported by the prior knowledge and a negative score is given to penalize certain structure. Here we do not force our BN structure learning method to include the edges in R into its output based on the assumption that the qualitative prior knowledge, though considered more reliable than the microarray data, still cannot be completely trusted. As a result, we quantify the belief that edge (X,Y) should exist by $bestCCF$. If the value of $bestCCF$ is small but $(X,Y) \in R$ or $(X,Y) \in nL(R)$, then the prior knowledge contradicts with the microarray data, a positive award score still will be given in this case but the bias is not significant; for the same reason, if the value of $bestCCF$ is small, the absence of (X,Y) in R will not lead to a great penalty of the structure.

The structure prior $\Pr(G)$ is the summation of the *edgeScore* for all edges:

$$\Pr(G) = \sum_{(X,Y)\in G} edgeScore(X, Y, tl) \tag{3.11}$$

Substitute (3.11) into (3.1), we get the biased score for integrating additional qualitative evidences.

3.3 Implementation

Ideally, by assessing the BIC score of every possible network structure, we can find the one which fits the data best. However, finding the best solution is computationally intractable [Chickering et al., 1994]. For this reason, in our implementation the equations developed in the previous sections are not computed in their exact format. Instead, we apply the score computation within a greedy search strategy to find a near-optimal solution with reasonable computational cost. The algorithm below outlines the standard greedy strategy:

[0]The software implementing our model can be downloaded from http://www.complex.iastate.edu

Start with:

a random structure **G**,

a BIC scoring function **score**$(\cdot)$, and

a pre-defined threshold **t** ;

while *true* **do**

for *every offspring* G'_n *of* G **do**
compute score(G'_n);

end

Find one structure $\hat{G}'$ which has the highest score among offsprings;

if *score($\hat{G}$) - score(G)* $> t$ **then**
G $\leftarrow \hat{G}'$;

else
output $\hat{G}$;

stop;

end

end

The offspring G' is the set of all results of one-change operations on the original graph G. There are three types of such operations:

- delete an existing edge in G
- add an edge to G
- reverse the the direction of an edge in G

In each iteration, the score of every one-change offspring is computed and the one with the best score is selected. If it cannot improve the score of the parent graph G significantly, i.e., the score difference between the best offspring and the parent is less than a predefined threshold t, the algorithm decides that the maxima/minima has been reached and stops, in the other words, the algorithm converges. Obviously this greedy search strategy may be trapped into a local maxima/minima, so in practice we restart the search multiple times with random initial starting graphs G, and select the graph with the best final score among these executions.

To adapt with this greedy search strategy, we decompose the scoring method depicted by equation (3.10) into three functions. Each function adjusts the score not only based on the attribution of the edge, but also the specific change operation made in one iteration:

$$score'_{add} = \begin{cases} score + \alpha \cdot bestCCF(X, Y, tl), & (X, Y) \in R \\ score + \beta \cdot bestCCF(X, Y, tl), & (X, Y) \in nL(R) \\ score - \gamma \cdot (1 - bestCCF(X, Y, tl)), & otherwise \end{cases}$$

$$score'_{delete} = \begin{cases} score - \alpha \cdot bestCCF(X, Y, tl), & (X, Y) \in R \\ score - \beta \cdot bestCCF(X, Y, tl), & (X, Y) \in nL(R) \\ score + \gamma \cdot (1 - bestCCF(X, Y, tl)), & otherwise \end{cases}$$

$$score'_{reverse} = \begin{cases} score - \alpha \cdot dirDiff, & (X, Y) \in R \\ score & (X, Y) \notin R \end{cases}$$

The subscription of each function refers to the type of changes it associates with. *score* is the $\Pr(D|G)$ value of the parent structure G. For an add operation, the score adjustment function adds an awarding score if the added edge agrees with the prior knowledge, and takes a penalty score if not. For a delete operation, the adjustment function acts in exactly the opposite way. For a reverse operation, assume the direction before the change is $X \rightarrow Y$, we compute the difference in $bestCCF$ as in equation (3.12), and use it to encourage or discourage this reverse operation:

$$dirDiff = bestCCF(X, Y, tl) - bestCCF(Y, X, tl) \tag{3.12}$$

We use OpenPNL (https://sourceforge.net/projects/openpnl/), an open source C++ library on graphical models developed by Intel Corporation, to perform the standard greedy search algorithm. In addition our model takes qualitative evidences and insert the score adjustment model described above in each iteration in the algorithm to influence the outcome of the structure learning algorithm.

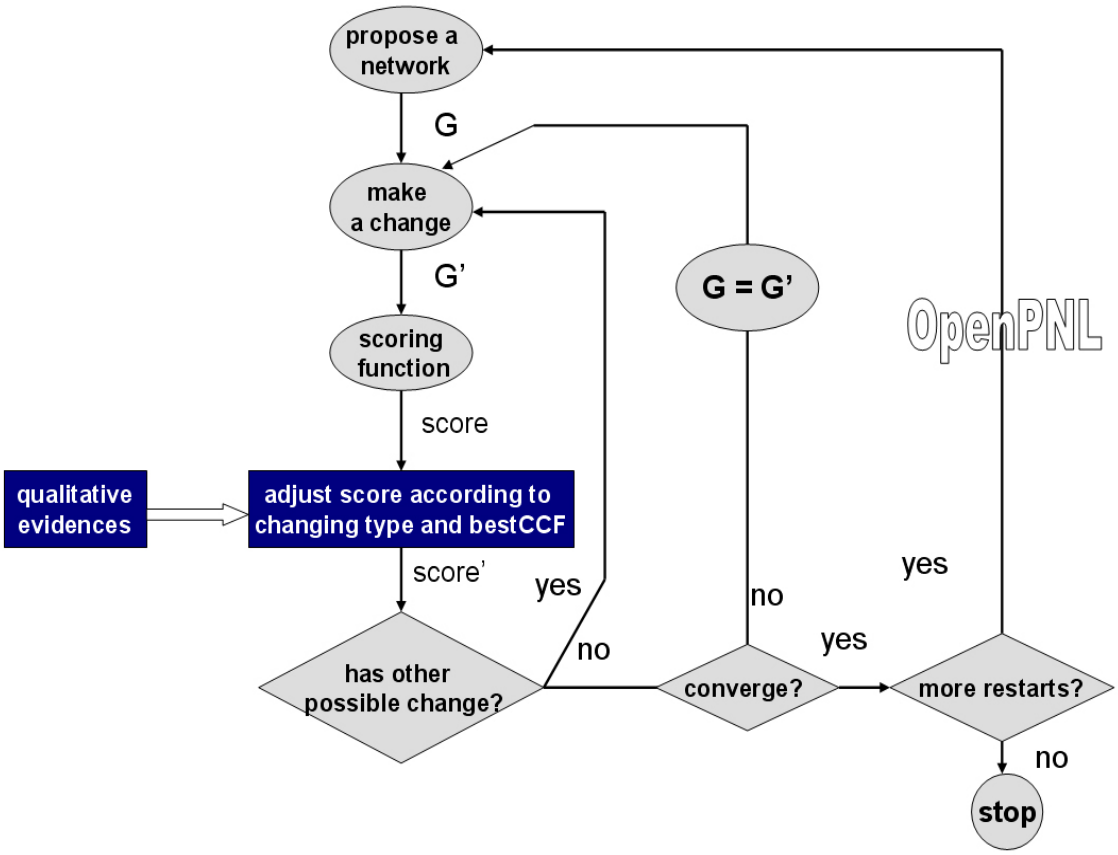

Figure 3.4 System diagram. Components drawn as rectangles are our enhancements models inserted into the standard greedy search algorithm.

3.4 Parallel BN structure learning

Although BNs have many advantages, it is difficult to use them to predict large network because of its intensive computational cost. Theoretically it has been proved that BN structure learning is NP-hard [Chickering et al., 1994]; in practice some heuristics are developed to solve the problem in polynomial time, however because of the large search space, the heuristics will run very slow once the number of nodes becomes large.

Unfortunately, most of the previously published research works which reconstruct gene networks using BNs primarily emphasize on the theoretical models, while the scalability and usability of their models were seldom addressed. Comparing to the models based on pairwise analysis that usually have thousands of nodes, most previous works based on BNs

only built gene networks in the scale of 20-30 nodes. A previous research effort reported in[Yin et al., 2004] tried to use parallel computing for learning gene regulatory networks, however, [Yin et al., 2004] did not provide any result.

In order to speed up the computation, our implementation parallelize the structure learning process so that it can make use of multiple CPUs, as shown in Figure 3.5.

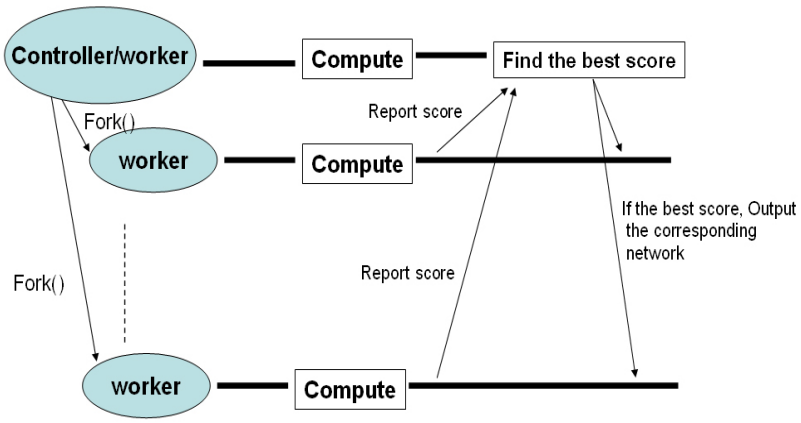

Figure 3.5 Parallel BN structure learning

We usually need to restart the greedy algorithm multiple times with random initial graph, these multiple executions are independent thus can be parallelized. Each process takes a portion of total number of executions, once it is done it reports the score via pipe to a pre-assigned controller. The controller gathers all reported score and find a best one, then instruct the correspondent process to output its network, and the other processes to stop silently.

In chapter 4, we show that by using parallel computing, our model can be used to generate larger gene networks. We build a network with 188 actual nodes in our experimental study. To our knowledge, our work is the first BN-based research effort reported to date which can generate gene networks with such scale.

CHAPTER 4. RESULTS

4.1 Data

We use the publicly available *Saccharomyces cerevisiae* cell-cycle gene expression dataset published by [Spellman et al., 1998] to evaluate our method. The dataset contains 79 gene expression measurements of 6177 genes. These 79 measurements were conducted under four synchronization protocols. Table 4.1 summarize these four protocols:

Table 4.1 Summary of the microarray dataset published by [Spellman et al., 1998]

	CDC15	CDC28	ALPHA	ELU
# of data points	24	17	18	14

The pre-processed data are downloaded from the paper companion website http://cellcycle-www.stanford.edu. Next we created two test cases:

- A network with 25 genes, including 14 well known transcription factors, and 11 related regulatees. By using MPact [Guldener et al., 2006] on the MIPS website [Mewes et al., 2000], we found 30 interactions among these genes which constitute a reference network. In the rest of this chapter we refer to it as the **small** test case.
- We use the result from [Simon et al., 2001] which has 94 genes and 209 interactions as the reference network. In the rest of this chapter we refer to it as the **large** test case.

The detailed gene and interaction lists of these two test cases can by found in the appendix B. The reference networks are considered to be correct and are used to evaluate the accuracy of our model. For the purpose of evaluating our supervised learner, we pretend to only know a part of the reference network, we call it training network in terms of machine learning. Consequently, the rest of edges in the reference network constitute the validation network which is unknown to the learning algorithm.

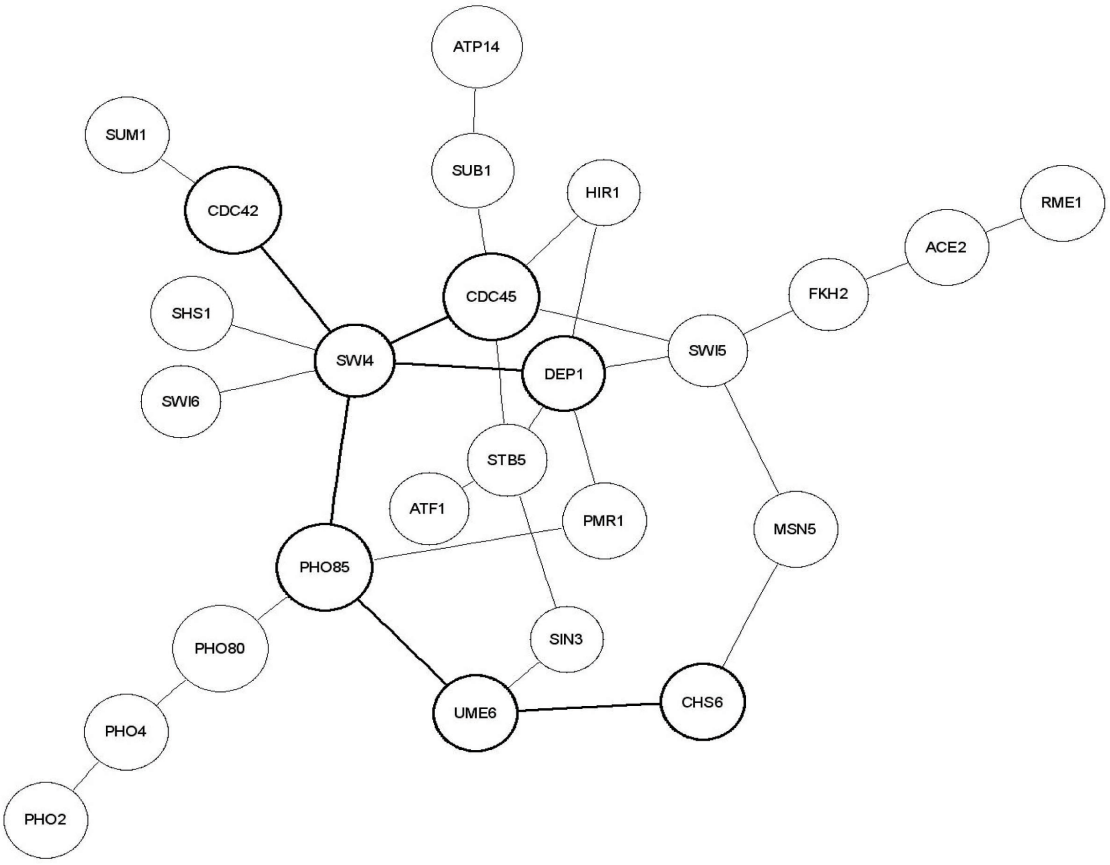

Figure 4.1 Reference Gene Network containing 25 genes and 30 interactions. 6 edges drawn in bold font: (SWI4, CDC42), (SWI4, CDC45), (SWI4, DEP1), (SWI4, PHO85), (PHO85, UME6) and (UME6, CHS6) form the training network

4.2 Experiment Setting

To evaluate the performance of our novel method, we conducted a comparative experiement. Three methods are used to generate gene networks:

- Randomly generate network topologies by a simulator. The result networks is constrainted by degree upper bound. i.e. every node in the result networks has no more than a certain number of edges. This method provides a "ground" performance standard. In the rest of this chapter we refer to it as **random** method.

- Our proposed new method. In the rest of this chapter we refer to it as **supervised** method.

- The standard DBNs which only takes microarray data. In the rest of this chapter we refer to it as **unsupervised** method in contrast to our method.

The two DBN related methods have the same settings and input data except that the supervised BN method uses the additional score adjustment function.

4.3 Results

4.3.1 Accuracy

We repeat the random method for 500000 times and take the average number of correctly predicted edges. Since it does not take microarray data, there are two cases to consider: for the small reference network, the average number of correct prediction is **6.77**. For the large reference network, the average number of correct prediction is **3.71**. The worse performance in case of large reference network can be explained by the sparseness of the network, despite the fact that there are more edges in the reference network.

For the DBN related methods, because the greedy search algorithm can only find sub-optimal network structures, and different network structures can have the same or very close scores which is known as "equivalence classes" [Chickering, 2002] of Bayesian network, the number of edges found in the validation network may also differ from time to time. To compare the performance of the two methods under this circumstance of uncertainty, we repeatedly run supervised and unsupervised methods on two test cases for 100 times each, and compare the average number of edges found in the validation network. To ensure the objectiveness of the

performance comparison, we only count the edges found in the validation set which is unknown to both methods. Table 4.2 shows the results of the supervised and unsupervised methods.

Here the sensitivity is defined as:

$$sensitivity = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

where *True Positive* is the number of edges in the validation network that are correctly predicted, *True Positive + False Negative* is the total edges in the validation network.

The values of α, β and γ are selected so that the sensitivities are maximized, Table 4.3 lists the details. Since BN structure learning process is influenced only by BIC score differences of BN structure candidates, to reduce the computational cost of searching best parameters, we fix γ to 0.1. The number of random restarts is 60, the maximum time lag tl is 3.

With the additional knowledge of only a small portion of the whole reference network as the training network, our method significantly improves the sensitivity of prediction in both test cases. The low sensitivities in the large case is conceivable since the number of genes is almost 4 times of the number of samples, makes this case severely under-sampled. However, as shown in Table 4.2, in any dataset and test case combination, comparing to the unsupervised method, supervised method can always improve the sensitivity by up to 30%. Methods using DBNs outperform the random method in almost every case, especially with large reference network, the advantage of using DBNs and microarray data is overwhelming over merely guessing.

The awarding score for edges in $nL(R)$ is not the only reason for the accuracy improvement. By studying the resulted networks of the small case, we found that the performance improvement is mainly contributed by 5 edges that were found much more frequently by the supervised method, Table 4.4 shows the detail. Two of these edges: ATP14 $\rightarrow$ SUB1 and SWI5 $\rightarrow$ MSN5, do not connect to the training network. Therefore, the successful prediction of them cannot be directly explained by the award score in our method. The score adjustment mechanism driven by the training network has to be working in concert with the standard DBNs method based on microarray data to make these correct predictions.

Table 4.2 Performance comparison

dataset	test case	method	edges found in validation set	total number of edges	sensitivity	sensitivity improvement
cdc15	Small	Unsupervised Supervised	8.88 11.45	68.2 68.8	37% 47.7%	**29.0%**
	Large	Unsupervised Supervised	11.86 15.48	400.5 394.6	7.6% 9.9%	**30.5%**
alpha	Small	Unsupervised Supervised	5.90 7.06	74.1 80.5	24.6% 29.4%	**19.5%**
	Large	Unsupervised Supervised	11.61 15.32	335.7 431.2	7.6% 10.0%	**32.0%**
cdc28	Small	Unsupervised Supervised	4.89 5.70	80.7 81.4	20.4% 23.8%	**16.7%**
	Large	Unsupervised Supervised	12.00 14.70	325.8 401.7	7.8% 9.6%	**22.5%**
elu	Small	Unsupervised Supervised	7.80 8.83	77.2 79.4	32.5% 36.8%	**13.2%**
	Large	Unsupervised Supervised	11.47 15.36	331.1 414.7	7.5% 10.0%	**33.7%**

Table 4.3 Parameter selections

dataset	test case	*alpha*	*beta*
cdc15	Small	2.4	2.55
	Large	3.5	3.2
alpha	Small	1.6	1.9
	Large	1.2	1.9
cdc28	Small	2.3	3.3
	Large	3.2	2.8
elu	Small	1.3	4.9
	Large	1.7	1.5

Table 4.4 Contributors of the performance improvement

	occurrence in Supervised BN	occurrence in unsupervised BN
SIN3 → UME6	83	0
ATP14 → SUB1	79	1
SWI5 → CDC45	85	42
SWI5 → MSN5	88	55
SWI4 → SHS1	91	49

4.3.2 Consistency

First we use the standard deviation of the number of correctly predicted edges as a measurement of consistency. Table 4.5 shows the results. Since the number of correctly predicted edges have different averages, to make the comparison fair, we use supervised method as the base, and adjust the standard deviation of the other two methods against it. As we can see from table 4.5, our method has smaller standard deviation in every case, and the random method performed the worst.

We also have an interesting finding that with certain parameter settings, the results of our method are very consistent on specifically a set of edges over repeated experiments. To illustrate this, we counted occurrences of every distinct edge in the result networks of 100 experiments, and summarize the number of edges in five occurrence ranges. The result is given in Table 4.6.

As shown in Table 4.6, both supervised and unsupervised methods have almost the same number of edges that are found just a few times, i.e., in the range of 1-19. Other than these, there is a significant difference in the way the numbers of edges are distributed into the other four ranges. For the unsupervised method, these edges are relatively evenly distributed. For our supervised method, a dominant majority, 68 out of 73 edges are found more than 80 times, and there is no edge found between 20 times to 59 times!

Figure 4.3.2 summarizes the number of all occurrences for each range. Clearly for supervised method most occurrences of edges fall into the range of 80-100 times, but for unsupervised method the occurrences are also evenly distributed. This shows that our method tends to converge to a few very similar network structures, while the results of unsupervised method are by far more diversified. It is conceivable that the fewer similar networks are more likely to resemble the correct gene network than a diversified set that are dissimilar.

4.3.3 Performance of the paralell implementation

Our parallel BN structure learning program is tested on a system with 4 AMD Opeteron CPUs, we also run the serial code performing the same computation on single CPU to compare

Table 4.5 A comparison of standard deviations – The standard deviations of Random networks and Unsupervised method are justified regarding to the supervised method, the original standard deviation values are given in parentheses

dataset	test case	Random adjusted (original)	Unsupervised adjusted (original)	Supervised
cdc15	Small	3.578 (2.116)	1.634 (1.267)	1.239
	Large	8.120 (1.965)	4.008 (3.072)	3.647
alpha	Small	2.207 (2.116)	1.784 (1.493)	1.689
	Large	8.113 (1.965)	4.027 (3.051)	3.333
cdc28	Small	1.782 (2.116)	1.745 (1.495)	1.618
	Large	7.785 (1.965)	3.798 (3.100)	3.356
elu	Small	2.759 (2.116)	2.139 (1.890)	1.659
	Large	8.135 (1.965)	4.649 (3.477)	3.477

Table 4.6 Number of edges in 5 groups according to their occurrences

range	80-100	60-79	40-59	20-39	1-19
Supervised	68	5	0	0	330
Unsupervised	20	28	24	45	321
Ratio	**3.4**	**0.18**	**0**	**0**	**1.03**

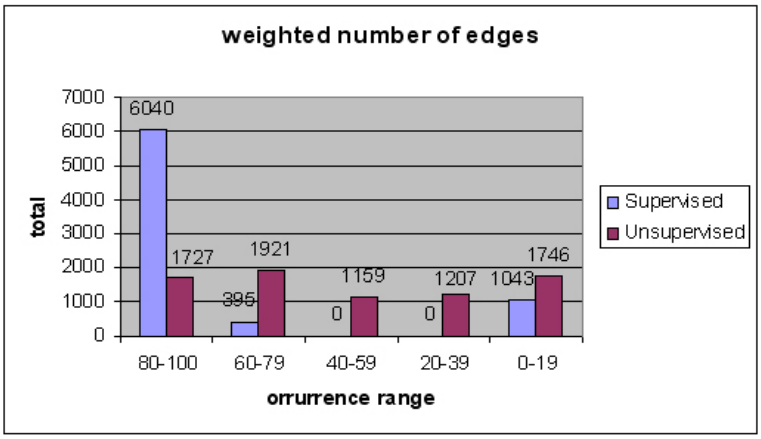

Figure 4.2 Weighted edge counts

the performances. The results are summarized in table 4.7:

Table 4.7 Running time comparison

	Parallel	Serial	Ratio
Supervised	179.23 sec	556.5 sec	**3.10**
Unsupervised	179.56 sec	549.68 sec	**3.06**

Every running time in table 4.7 is the average of 100 executions. Despite some synchronization overhead, our parallel code can speed up the computation by approximately 3 times.

CHAPTER 5. RECONSTRUCTING GENE NETWORKS USING SHORT-TIME CORRELATION

5.1 Background

The usage of CCF in Chapter 3 tries to capture the linear relationships between entire expression profiles. However, the real gene interactions usually happen within specific time periods and conditions. For example, the development of a yeast cell can be divided into four stages: G1, S, G2 and M as illustrated in figure 5.1. Some genes will only be "turned on" during a specific stage, therefore the expression of these genes may show a strong pattern only in a relatively short period of time. If we compute CCF of the entire gene expression profiles such short-duration interactions might be missed.

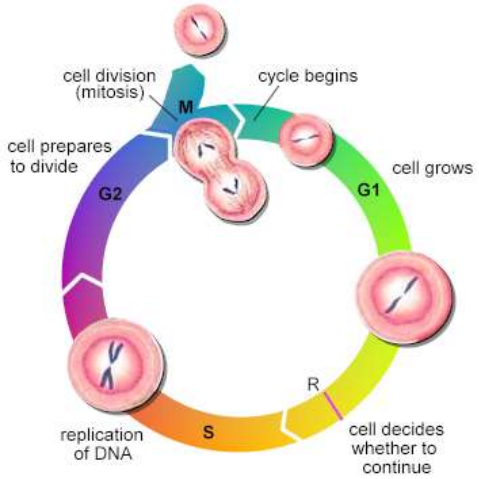

Figure 5.1 Yeast cell cycle

As figure 5.2 shows, if there exists regulatory relationship between two genes, it is possible that such relationship may be reflected by a part of expression profile. To capture the short-time high correlation region, we need to define a sliding window so that partial expression

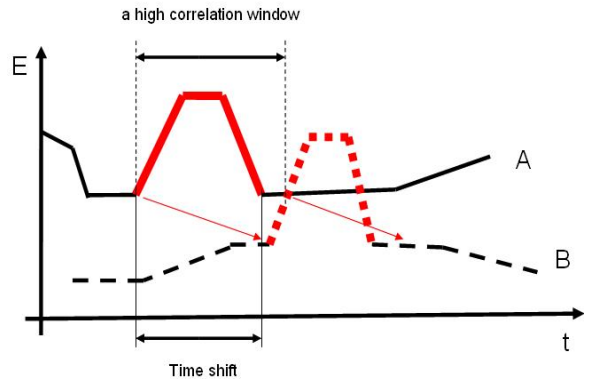

Figure 5.2 Using sliding window for time-series microarray gene expression data

data will be used for computing correlation, while the rest of expression data (possibly low correlation region) will be ignored.

In this chapter, we develop a new model which combines short time correlation and DBN to integrate biological data in addition to the microarray data to improve the performance of network inference in terms of both accuracies and computational cost. In our new model we add window size as another parameter for the CCF calculation, thus only a portion of the time-series (the length is adjustable) is used for CCF calculation. We applied the new model on yeast cell cycle microarray expression data and compare the accuracy of the predicted gene networks with those produced by standard DBN methods. The results show that our model outperformed the method without considering sliding window by finding gene interactions whose sub-sequences produce the highest correlation. [1]

5.2 Method

We adopt the correlation formula developed by [Du, 2005] which consider sliding window. In short-time correlation, a slide window function is multiplied with the expression profiles. Time correlation is computed over the profiles under the nonzero window function. The short-time correlation coefficient can be defined as:

[1] Collaborated with Tian Xia and Pan Du

$$r_{ijmM}(\tau) = r_{ijmM}(l\Delta t) = cov(x'_{imM}, x'_{jmM}) / \sqrt{var(x'_{imM}) var(x'_{jmM})} \tag{5.1}$$

where

$$x'_{imM}[k] = w_M(k-m)x_i[k], x'_{jmM}[k] = w_M(k+l-m)x_j[k+l],$$
$$k = 1, \cdots, N, m = 1, \cdots, N, 1 \leq k+l \leq N$$

$$w_M(k) = \begin{cases} 1, & -[M/2] \leq k \leq [M/2] \\ 0, & otherwise \end{cases} \tag{5.2}$$

$r_{ijmM}(\tau)$ represents short time correlation coefficient between expression profiles X_i and X_j with time delay τ, window size M at time frame m, m is the time index where the center of window function is located, M is the size of the window function $W_m(k)$, τ is the time shift between gene profiles X_i and X_j, $\tau = l\Delta t$, Δt is the sample interval, l is the number of sample intervals shifted between two profiles, N is the profile length. For periodic time profiles, circular time correlation is effective, i.e., the time points at the end of the time series are rewound to the beginning of the series after time shifting. The estimation of time delay τ and edge direction during time interval m is exactly the same as time correlation. Since the window size is changeable, in order to make the time correlation with different window size comparable, we translate all the short-time correlation coefficients $r_{ijmM}(\tau)$ as p-values. To simplify the notation, in the rest of this chapter, we use p-value in the rest of this chapter to denote the measurement of short-time correlation coefficients.

Suppose the p-value between two time-series profile is written as $pValue(X_i, X_j, tl, w)$, where X_i and X_j are time series, w is window size, tl is the time shift. We further define a function $bestPValue$ returning the minimum of p-values with difference window size:

$$bestPValue(X_i, X_j, tl) = min(pValue(X_i, X_j, tl, w)), w \in W \tag{5.3}$$

where W is the set of all valid window sizes.

Next we use $1/bestPValue$ as a substitution of $bestCCF$ defined in equation 3.9, consequently the scoring function in chapter 3 is changed to:

$$edgeScore(X,Y) = \begin{cases} \alpha \cdot 1/bestPValue(X,Y,tl), & (X,Y) \in R \\ \beta \cdot 1/bestPValue(X,Y,tl), & (X,Y) \in nL(R) \\ -\gamma \cdot 1/bestPValue(X,Y,tl), & otherwise \end{cases} \tag{5.4}$$

This scoring function is used in the same procedure described in chapter 3 for gene network reconstruction.

5.3 Experimental study

5.3.1 Data

We used the public yeast (Saccharomyces cerevisiae) mitotic cell cycle data set [Cho et al., 1998]. The data was synchronized by arresting cdc28-13 cells in late G1. 17 time points with 10 min intervals were collected. The Affymetrix Genechip was used for measuring mRNA accumulation levels. Our analysis was based on 140 genes listed at the paper companion website, including both cell cycle TFs (Transcriptional Factor) and their target genes. The relationships between the TF and the target genes were identified using genome-wide location analysis[Simon et al., 2001].

We selected 20 genes from the dataset, by studying the previous biological literature we constructed a reference network shown in Figure 5.3. The edges in reference network are considered reliable and therefore the target of network inference models.

5.3.2 Result

We first compute the $bestPValue$ for every gene pair, and record the time shift and window size associated with this best p-value. Table 5.1 shows the details about all 15 edges in the

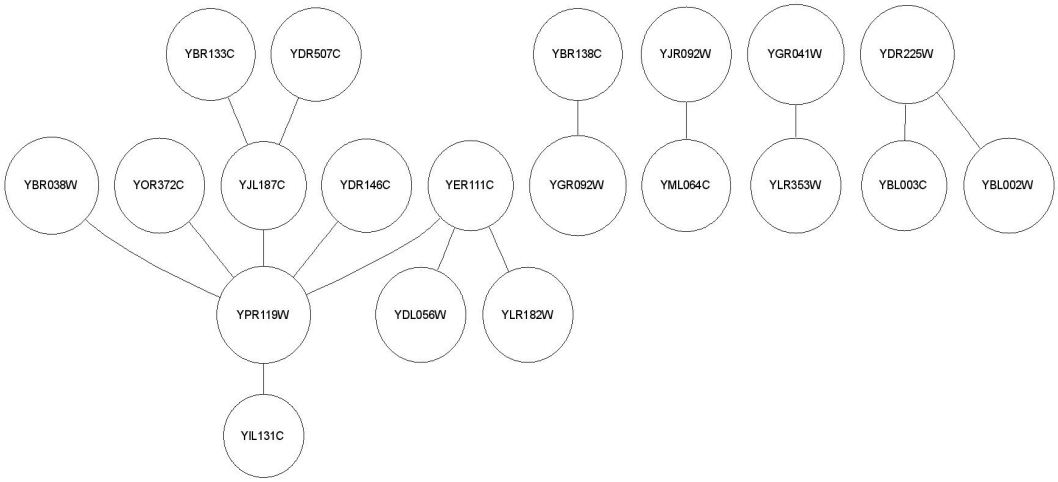

Figure 5.3 Reference network. Every edge indicates an interaction reported by literature.

reference network. Since there are 17 data points in our microarray dataset, the maximum window size is 17. To prevent the unreliable result produced by very short sequences, we limit the window size in the range of 10 to 17. There are 5 gene pairs in Table 5.1 (YBR138C – YGR092W, YOR372C – YPR119W, YBR133C – YJL187C, YER111C – YDL056W and YJR092W – YML064C) have $bestPValue$ produced by window size smaller than 17, which indicates that partial time sequence sometimes can better capture the likeliness of two microarray profiles.

Next, we use these 5 edges as qualitative evidence in the aforementioned data integration model, and use p-values in Table 5.1 as $bestPValue$ in equation 5.4 to quantify them. In case the p-value is 0, we set a large number for $1/bestPValue(X, Y, tl, w)$. In order to evaluate the performance of our data integration model, we also conducted an experiment which uses standard DBN on the same microarray dataset.

Figure 5.4 shows the network produced by the standard DBN, while Figure 5.5 shows the network produced our method. Among those five edges with $bestPValue$ produced by partial sequence, one edge (YOR372C – YPR119W) is missed in the network produced by standard DBN method which only computes correlation of two full length time-series. On the other hand, our method found all five edges.

Table 5.1 minimum p-values and their associated time shift and window size (rows with bold font are those gene pairs whose minimum p-values are produced by sliding window shorter than the full length of the time series)

Gene A	Gene B	time shift	winSize	p-value
YBR038W	YPR119W	0	17	2.86E-07
YBR138C	**YGR092W**	**1**	**10**	**0**
YOR372C	**YPR119W**	**2**	**10**	**1.18E-05**
YJL187C	YPR119W	4	17	1.99E-05
YBR133C	**YJL187C**	**1**	**13**	**4.15E-05**
YDR507C	YJL187C	0	17	5.22E-07
YER111C	**YDL056W**	**-4**	**14**	**0**
YER111C	YPR119W	5	17	1.03E-05
YER111C	YLR182W	0	17	4.80E-05
YDR146C	YPR119W	5	17	6.75E-06
YPR119W	YIL131C	4	17	2.33E-05
YJR092W	**YML064C**	**1**	**14**	**0.002156**
YGR041W	YLR353W	1	17	2.51E-06
YDR225W	YBL003C	0	17	3.43E-05
YDR225W	YBL002W	0	17	0.000169

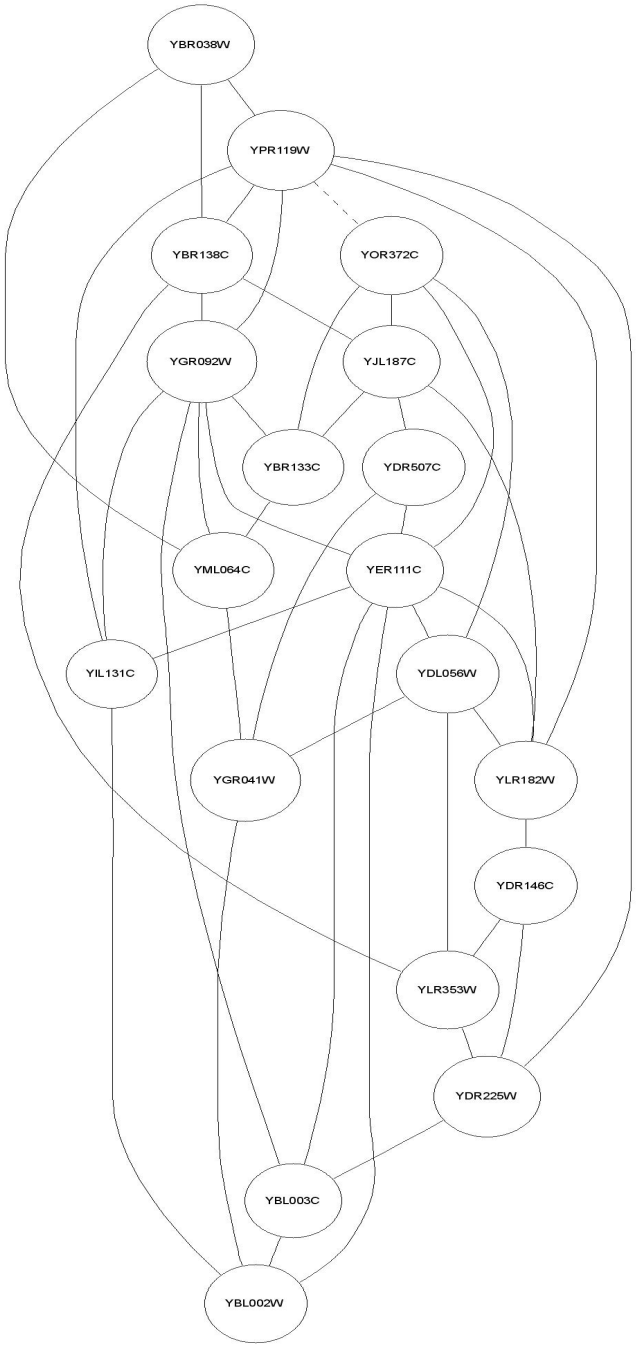

Figure 5.4 The network produced by standard DBN. (only takes microarray data as input), the dashed line is the biologically meaningful gene relationship with $bestPValue$ produced by partial sequence, however standard DBN method failed to predict it.

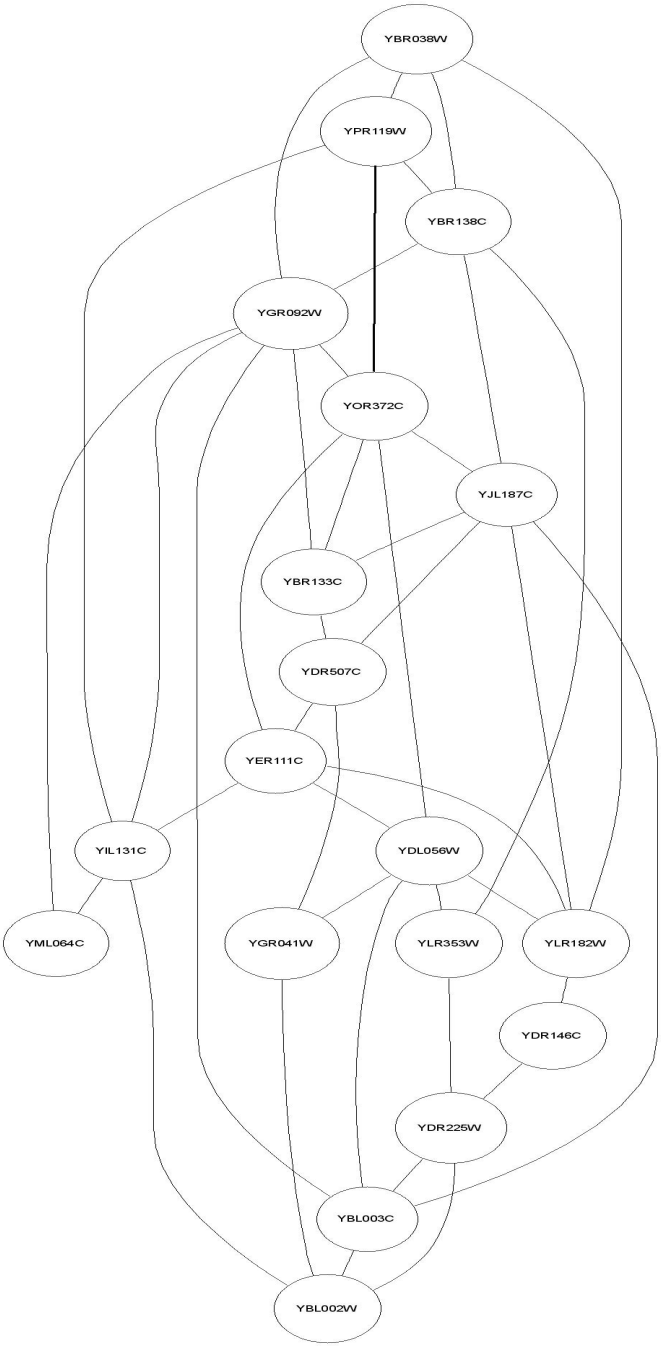

Figure 5.5 The network produced by our method (takes both microarray data and qualitative prior). The edge between "YPR119w" and "YOR372C" (with bold line) is missed in the standard DBN's prediction.

CHAPTER 6. VISUALIZATION

6.1 Introduction

For biologists, it is crucial to understand the pathways that describe the relationships and the interactions among molecules functioning in a cell. However, it is difficult for humans to understand a very large graph directly from the raw data, so visualization techniques are required to transform raw data into a more understandable format.

Some approaches of pathway visualization have been proposed previously. Kyoto Encyclopedia of Genes and Genomes (KEGG) [Kanehisa and Goto, 2000] provides metabolic pathway data both in XML-format files and manually created 2-D maps. Although these maps are of very good quality, one obvious drawback is that every change of the pathway may cause considerable effort to redraw the map. KEGG also provides a viewer that automatically draws pathways in 2-D space. However due to the limitation of 2-D layout, in these maps many crossings between lines and texts can be seen, especially when the pathways are complex.

Figure 6.1 shows a comparison of the visual effectiveness between 2-D and 3-D layout. Figure 6.1(A) is a part of the pathway Pantothenate and CoA biosynthesis in 2-D space that is drawn by KEGGs automatic pathway viewer. It is obvious that many lines, nodes, and texts cross each other, which makes it hard for users to understand the relationships among them, whereas in Figure 6.1(B), the same relations in 3-D space is much clearer to understand.

We believe that when the number of nodes and edges in a graph becomes large, it is more preferable to use 3-D technology to visualize it. The advantage of 3-D visualization is that users always have the ability to change the viewpoint to make a better observation.

[0]This chapter is an extended version of the paper: Li, S. and Chou, H.-H. (2005). Ubviz: a software tool for exploring metabolic pathways in 3-d space. Biotechniques, 38(4):540, 542. [Li and Chou, 2005]

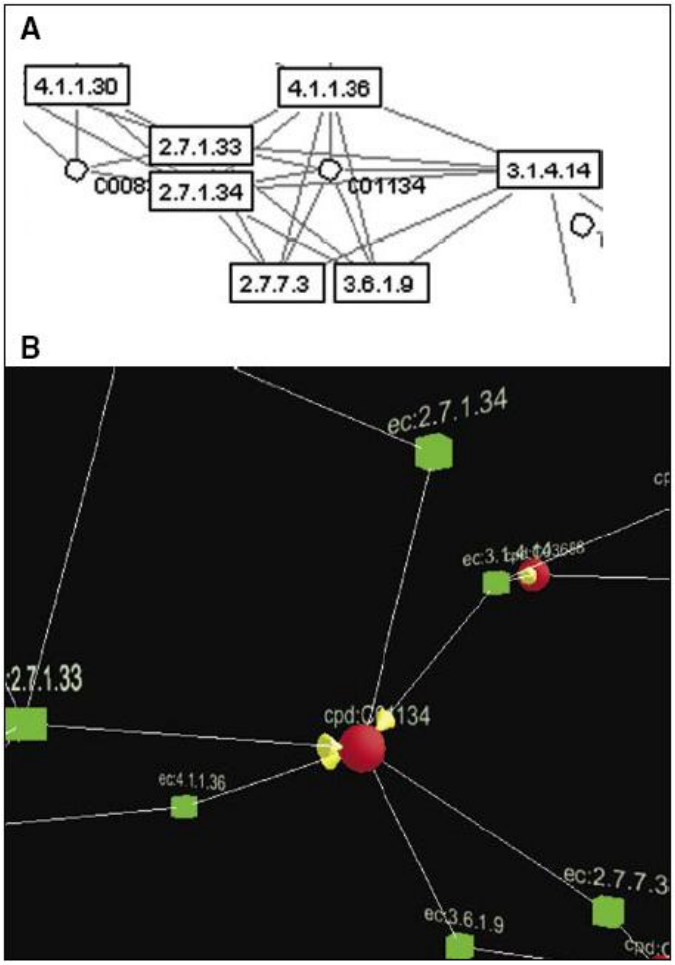

Figure 6.1 A comparison between 2-D and 3-D visualizations of the same metabolic pathway. (A) Pathway in 2-D space (source: www.kegg.org). (B) The same pathway in 3-D space.

For example, if we draw the same pathway Pantothenate and CoA biosynthesis in 3-D space, as shown in Figure 1B, a user can zoom-in and rotate to an appropriate angle in 3-D space so that the links to node C00134 can be seen more clearly.

In the system described in [Rojdestvenski, 2003], a 3-D graph layout algorithm was used on metabolic pathways, and the results were stored in Virtual Reality Modeling Language (VRML) format. Users can download these VRML files and view the 3-D VRML images with a VRML browser. This approach allows data providers to publish 3-D pathway maps on their web site. However, users can only passively receive the data in the visual format that the data providers have already decided.

6.2 Method

In this paper, we introduce a novel approach for bringing metabolic pathways into 3-D space. Compared with the aforementioned system, this approach has several advantages. First, our system, UBViz, only needs logical descriptions of the pathways from the data providers and performs the 3-D layout algorithm locally. As a result, less data is needed to be transferred through the network, since the coordinate information is not needed by UBViz. Second, the UBVizs XML interface enables existing XML format pathway files from KEGG to be immediately usable; no change on the KEGG side is needed. Therefore, UBVizs users can make use of all existing database resources, such as the pathway data KEGG provides, and benefit from modern computer 3-D technology immediately without requiring the data providers to expend any effort. Finally, UBViz is a standalone highly efficient C++ program that does not depend on any other software to execute and is easily installable by end-users.

UBViz can use its embedded FTP functionality to download pathway files from KEGGs FTP site directly. Incoming XML files are parsed by a module based on Expat (http://expat.sourceforge.net). Useful information extracted from the XML files is then kept by UBVizs parser module. This includes (i) a list of nodes and edges that are essential for 3-D layout; (ii) each nodes type and name; (iii) information about whether an edge is directed or not, which is used in the map drawing; and (iv) the URL of the reference page for a node,

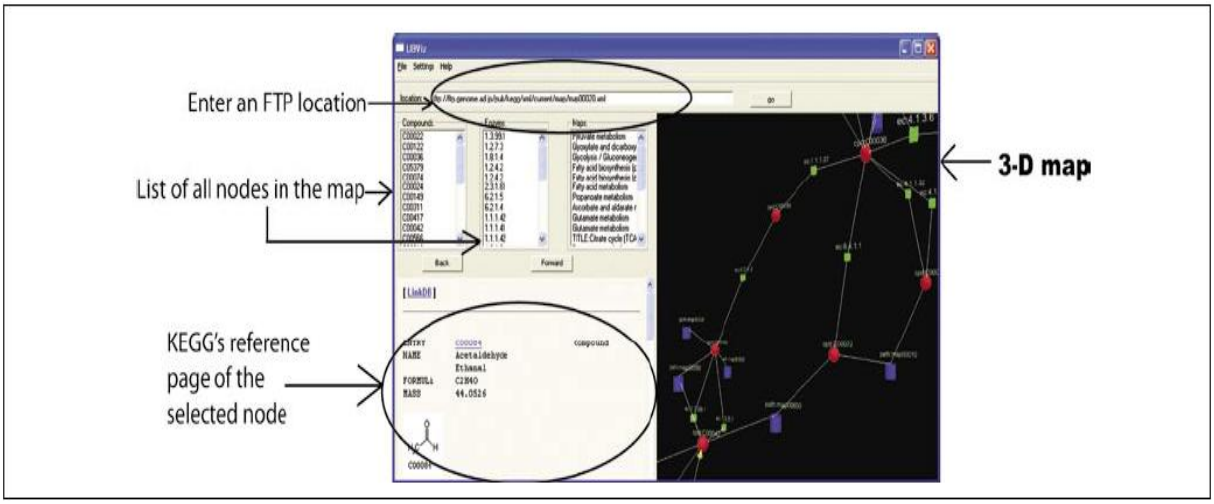

Figure 6.2 UBViz's user interface

which is used to show the HTML-format explanation of the node. Next, the node and edge information is fed into a 3-D layout module based on GraphViz [Ellson et al., 2003].

After the layout computation, 3-D coordinates for each node are generated and are used to draw the objects on the screen. The results are rendered by OpenGL (http://www.opengl.org). As shown in Figure 6.2, we use three basic 3-D shapes to represent the three different types of nodes: (i) a sphere (red) represents a compound; (ii) a box (green) represents an enzyme; and (iii) a cylinder (purple) represents a link to the connected pathway. For a directed edge, an arrow (yellow) is added to one end of the edge to denote its direction. The viewpoint of the scene can be easily changed by using a set of the 3-D navigation methods implemented in UBViz, which provides adequate power for users to explore the pathways in a virtual 3-D space.

UBViz has a user-friendly graphical interface developed using wxWidgets (http://www.wx-widgets.org), an open-source multiplatform graphical user interface (GUI) toolkit. As shown in Figure 6.2, all the users need to know is the name of the metabolic pathway file that UBViz should load. Users can choose to load a file stored in a local disk or to get the file directly from KEGG by inputting the file name in the FTP Location text field. After clicking Go, pathway files will be downloaded and processed in the background. All nodes in the map will

be displayed in three list boxes on the left side of the window according to their types, and the 3-D map will be shown on the right side. If users want to know more details about a compound or an enzyme, they can click on the corresponding name in the list boxes. KEGGs reference page will then be shown in the HTML panel below, and the viewpoint of the 3-D map will be automatically adjusted to face the node just selected by the users. Although we have not explored the maximum size of a pathway that can be easily rendered and manipulated in UBViz, the program has successfully handled the largest pathway graph available from the KEGG database, a representation of purine metabolism, which contains 283 nodes and 488 edges. UBViz is distributed under the Gnu General Public License and can be downloaded from http://www.complex.iastate.edu/download/UBViz/index.html. UBViz is available on all major computing platforms including Windows, Linux, and Mac OS X.

CHAPTER 7. CONCLUSION

In this dissertation I presented a novel method to improve the quality of gene network reconstruction. Our method combines many essential ideas developed in previous works on the topic of gene network reconstruction, and significantly extends the capability of the prior approaches. Most of the previously proposed approaches in this field only have one of the two desired features of a gene network reconstruction method: (1)integrating heterogeneous data, and (2)generating cyclic gene network reflecting temporal programming of gene transcription activity. Our method, on the other hand, distinguishes itself by having both of these advantages as well as the capability to utilize qualitative information objectively.

In comparison with the existing published methods, we adopt the idea of training network from [Kato et al., 2005] and [Yamanishi et al., 2004], where the prior knowledge is posed in the format of a partial network serving the purpose of training the supervised learner. Their works, however, are based on kernel methods and therefore the output graph has indirect edges. The works of [Imoto et al., 2003] and [Nariai et al., 2004] integrate the knowledge such as protein-protein interactions under the BN framework, our method differs from their methods in two aspects. First, our model can be applied on temporal data. Second, in their works, each protein-protein interaction has to be assigned a value to be used for calculating the BIC score. This pre-processed protein-protein interaction data, in the strict sense, is not qualitative anymore. In terms of using DBNs and time-series microarray data, the work reported in [Bernard and Hartemink, 2005] has some similarity with ours, but our method differs by focusing on integrating qualitative prior knowledge.

We also integrated short-time correlation analysis into our Bayesian network based GRN reconstruction process. The experiment results shows that it is a promising direction.

In this dissertation we also focus on the scalability issue of BN-based gene network learning, which is usually ignored by the previous research works. Our work successfully uses parallel computing and multi-CPU system to speed up the BN structure learning process. The system reported in [Yin et al., 2004] has the similar idea with ours; however, [Yin et al., 2004] did not provide any result. In our experimental study, we build a network with 94 genes, since we use DBN with 2 time slices, there are 188 actual nodes in the result BN.

Our software system can be extended to systems with more CPUs. We believe that with more microarray data generated, using supercomputer to reconstruct genome-scale gene network would be feasible in the future, and our system could be a valuable prototype for that task.

APPENDIX A. THE DERIVATION OF VIOLATION FUNCTION

[1]Without loss of generality, the qualitative evidences are represented as the answers of a expert regarding to a certain matter, the answer can be either "yes" or "no". Next we define a violation function $violation(\theta, C)$ that indexes the extent to which the CPTs violate expert's answer. Violation takes the value 0 if there is no violation at all and some positive value otherwise which increases with the seriousness of the violation.

A computationally convenient function that meets these requirements is the following one:

$$\Pr(answer = yes|\theta, C) = exp(-w \cdot violation(\theta, C)) \tag{A.1}$$

where the positive weight w determines how quickly the probability decreases from its maximum of 1 as the extent of violations increases.

We define $x_{i1} < x_{i2} < \ldots < x_{in_i}$ for every node X_i with n_i discrete states that is involved in a qualitative influence. A qualitative influence is denoted by $S^?(X_w, X_z)$, where $? \in \{+, -\}$ describes the quality (+ or -) of a monotonic relationship between a variable X_w and one of its children X_z. Two kinds of qualitative influences exist: If a positive one holds, an increase in the state of X_w causes an increase (or at least no decrease) in the state of X_z. If the relationship is negative, an increase in X_w 's state causes a decrease (or at least no increase) in X_z's state.

A positive qualitative influence $S^+(X_w, X_z)$ can be described as follows: For any given value of X_z, an increase in the value of X_w will not decrease the probability that the value of X_z is equal to or greater than that given value. Formally, for all states $x_z m$ of X_z with m¿1 and all distinct pairs of states $x_w i$, $x_w j$ of X_w such that i ¿ j and for all possible state

[1]This appendix is a summary of [Wittig and Jameson, 2000]

configurations y of X_z's parents other than X_w, the following inequality must hold:

$$\Pr(X_z \geq x_{zm}|x_{wi}, y) \geq \Pr(X_z \geq x_{zm}|x_w j, y) \tag{A.2}$$

In terms of the conditional probabilities for individual states of X_z, the inequality A.2 can be written in the following format:

$$\sum_{l=m}^{n_z} \Pr(x_{zl}|x_{wi}, y) \geq \sum_{l=m}^{n_z} \Pr(x_{zl}|x_{wj}, y) \tag{A.3}$$

and

$$c'^{?wz}_{mijy} := \sum_{l=m}^{n_z} \Pr(x_{zl}|x_{wi}, y) - \sum_{l=m}^{n_z} \Pr(x_{zl}|x_{wj}, y) \geq 0 \tag{A.4}$$

For every violated positive constraint, there is at least one case does not satisfy the inequality of A.4. Note so far we only consider the positive constraint, however the mathematically representation of negative constraint can be defined analogously.

The individual violation quantitity is given as:

$$c^{?wz}_{mijy} = \begin{cases} -c'^{?wz}_{mijy}, & if ? = + \, and \, c'^{?wz}_{mijy} < 0 \\ c'^{?wz}_{mijy}, & if ? = - \, and \, c'^{?wz}_{mijy} > 0 \\ 0, & otherwise \end{cases} \tag{A.5}$$

Finally, the violation function is defined as the sum of all of the individual violations:

$$violation(\theta, C) = \sum_{m,i.j.y,w,z} c^{?wz}_{mijy} \tag{A.6}$$

APPENDIX B. GENES AND REFERENCE NETWORKS OF TWO EXPERIMENTS

B.1 Experiment 1 (small)

List of genes:

DEP1	HIR1	PHO2	SHS1	SWI5	UME6
SUM1	MSN5	SWI4	PHO4	PMR1	RME1
STB5	CHS6	CDC45	ACE2	SWI6	CDC42
ATP14	SUB1	FKH2	PHO80	SIN3	ATF1
PHO85					

Reference Network:

SWI4-SWI6	SWI4-DEP1	SWI4-SHS1
SWI4-CDC42	SWI4-CDC45	SWI4-PHO85
ATF1-STB5	PHO4-PHO2	PHO4-PHO80
SWI5-MSN5	SWI5-DEP1	SWI5-CDC45
CDC45-STB5	CDC45-SUB1	DEP1-PMR1
DEP1-STB5	STB5-SIN3	PHO85-PHO80
PHO85-PMR1	SUB1-ATP14	RME1-ACE2
FKH2-SWI5	FKH2-ACE2	SUM1-CDC42
UME6-PHO85	UME6-SIN3	MSN5-CHS6
CHS6-UME6	HIR1-DEP1	HIR1-CDC45

Training Network:

SWI4-CDC42	SWI4-CDC45	SWI4-DEP1
SWI4-PHO85	PHO85-UME6	UME6-CHS6

B.2 Experiment 2 (large)

List of genes:

CLN3	HTB2	HTA2	MCM2	HHF1	HHT1
TEC1	HSL7	MCD1	PSA1	MBP1	DUN1
PCL2	PCL9	HO	SWI5	HTB1	HTA1
GIC2	MFA1	GIN4	MCM3	MNN1	RNR1
RAD51	SWI4	STE2	ALK1	SCW11	AGA2
OCH1	CDC20	MCM6	MSB2	BUD9	DBF2
TEL2	CLB6	SKN1	RSR1	CRH1	SCW4
WSC4	GIC1	SPO12	IRR1	PCL7	SIM1
FKH1	FAR1	CIS3	SWE1	CDC6	BUD4
CWP1	CWP2	RAD27	ASH1	STE6	SIC1
RAX2	CDC45	ACE2	SWI6	CLB4	CDC46
CTS1	EXG1	BUD8	TEM1	MCM1	PDS5
CTF18	CLN1	GAS1	FKH2	MFA2	APC1
CHS1	PCL1	EGT2	AGA1	CDC21	NDD1
HOS3	HHO1	CLN2	ARP7	OPY2	CLB2
CLB5	KRE6	HTA3	GLS1		

Reference network:

PCL9-SWI5	CDC6-SWI4	CDC6-SWI6
CDC6-MBP1	CDC6-MCM1	SIC1-SWI5
SWI4-SWI4	SWI4-SWI6	SWI4-MBP1
SWI4-MCM1	PCL2-SWI4	PCL2-SWI6
PCL2-ACE2	PCL2-SWI5	CLB6-SWI4
CLB6-SWI6	CLB6-MBP1	CLB6-FKH2
CLB5-MBP1	CLB5-SWI6	SWE1-SWI4
SWE1-SWI6	SWE1-MBP1	PCL1-SWI4
PCL1-SWI6	PCL1-MBP1	PCL1-FKH2
PCL1-MCM1	CLN2-SWI4	CLN2-SWI6
CLN1-SWI4	CLN1-SWI6	CLN1-MBP1
CLN1-FKH1	CLN1-FKH2	OPY2-MBP1
OPY2-SWI6	NDD1-SWI4	NDD1-SWI6
CLB4-FKH1	SIM1-SWI4	SIM1-SWI6
SIM1-FKH2	SIM1-MCM1	PCL7-SWI5
HSL7-FKH2	APC1-FKH1	ACE2-FKH1
ACE2-FKH2	ACE2-MCM1	ACE2-NDD1
CLB2-SWI4	CLB2-SWI6	CLB2-FKH1
CLB2-FKH2	CLB2-MCM1	CLB2-NDD1
SWI5-FKH2	SWI5-MCM1	SWI5-NDD1
TEM1-FKH1	TEM1-FKH2	CDC20-FKH2
CDC20-MCM1	CDC20-NDD1	SPO12-MBP1
SPO12-SWI6	SPO12-FKH2	SPO12-MCM1
SPO12-NDD1	CLN3-MCM1	CLN3-ACE2

CLN3-SWI5	DBF2-MCM1	FAR1-MCM1
CHS1-SWI5	TEC1-SWI5	EGT2-ACE2
EGT2-SWI5	GIC2-SWI4	GIC2-SWI6
GIC2-MBP1	SCW11-ACE2	SCW11-SWI5
GIN4-SWI4	GIN4-SWI6	GIN4-MBP1
BUD9-SWI4	BUD9-SWI6	BUD9-FKH1
BUD9-MCM1	BUD9-ACE2	BUD9-SWI5
OCH1-SWI4	OCH1-SWI6	CTS1-FKH2
CTS1-ACE2	CTS1-SWI5	RSR1-SWI4
RSR1-SWI6	CRH1-SWI4	CRH1-SWI6
CRH1-MBP1	CRH1-SWI5	MSB2-SWI4
MSB2-SWI6	MNN1-SWI4	MNN1-SWI6
EXG1-SWI4	EXG1-SWI6	EXG1-MBP1
EXG1-FKH1	EXG1-FKH2	EXG1-ACE2
EXG1-SWI5	GLS1-SWI4	GLS1-SWI6
GAS1-SWI4	GAS1-SWI6	PSA1-SWI4
PSA1-SWI6	PSA1-ACE2	KRE6-SWI4
KRE6-SWI6	GIC1-SWI4	GIC1-SWI6
GIC1-FKH2	CWP1-SWI4	CWP1-SWI6
CWP1-FKH2	CIS3-SWI4	CIS3-SWI6
CIS3-FKH2	CWP2-SWI4	CWP2-SWI6
CWP2-MBP1	BUD4-FKH1	BUD4-FKH2
BUD4-MCM1	BUD4-NDD1	WSC4-ACE2
BUD8-FKH1	SCW4-SWI4	SCW4-SWI6
RAX2-SWI4	RAX2-SWI6	RAX2-FKH2
RAX2-MCM1	RAX2-NDD1	SKN1-MCM1
RNR1-SWI4	RNR1-SWI6	RNR1-MBP1

RNR1-FKH2	RAD27-MBP1	RAD27-SWI6
CDC21-MBP1	CDC21-SWI6	IRR1-MBP1
IRR1-SWI6	MCD1-MBP1	MCD1-SWI6
PDS5-MBP1	PDS5-SWI6	PDS5-FKH1
PDS5-FKH2	RAD51-MBP1	RAD51-SWI6
RAD51-FKH2	DUN1-MBP1	DUN1-SWI6
ALK1-MCM1	CTF18-FKH1	HHF1-FKH1
HHT1-FKH1	HTB2-SWI4	HTB2-SWI6
HTB1-SWI4	HTB1-SWI6	HTA1-SWI4
HTA1-SWI6	HTA2-SWI4	HTA2-SWI6
HHO1-SWI4	HHO1-SWI6	TEL2-FKH1
ARP7-FKH1	HTA3-SWI4	HTA3-SWI6
CDC45-MBP1	CDC45-SWI6	MCM2-MBP1
MCM2-SWI6	MCM6-MCM1	ASH1-SWI5
AGA2-MCM1	AGA1-SWI4	AGA1-SWI6
AGA1-MBP1	AGA1-MCM1	HO-SWI4
HO-SWI6	MFA1-MCM1	MFA2-MCM1
MFA2-SWI5	STE6-MCM1	STE2-MCM1

Training network:

CDC6-SWI4	CDC6-SWI6	CDC6-MBP1
CDC6-MCM1	SWI4-SWI4	SWI4-SWI6
SWI4-MBP1	SWI4-MCM1	CLN1-SWI4
CLN1-SWI6	CLN1-MBP1	CLN1-FKH1
CLN1-FKH2	NDD1-SWI4	NDD1-SWI6

ACE2-FKH1	ACE2-FKH2	ACE2-MCM1
ACE2-NDD1	SWI5-FKH2	SWI5-MCM1
SWI5-NDD1	SPO12-MBP1	SPO12-SWI6
SPO12-FKH2	SPO12-MCM1	SPO12-NDD1
BUD9-SWI4	BUD9-SWI6	BUD9-FKH1
BUD9-MCM1	BUD9-ACE2	BUD9-SWI5
CTS1-FKH2	CTS1-ACE2	CTS1-SWI5
CRH1-SWI4	CRH1-SWI6	CRH1-MBP1
CRH1-SWI5	PSA1-SWI4	PSA1-SWI6
PSA1-ACE2	GIC1-SWI4	GIC1-SWI6
GIC1-FKH2	CIS3-SWI4	CIS3-SWI6
CIS3-FKH2	BUD4-FKH1	BUD4-FKH2
BUD4-MCM1	BUD4-NDD1	

Bibliography

[Akutsu, 1999] Akutsu, T. (1999). Identification of genetic networks from a small number of gene expression patterns under the boolean network model.

[Alizadeh et al., 2000] Alizadeh, A. A., Eisen, M. B., Davis, R. E., Ma, C., Lossos, I. S., Rosenwald, A., Boldrick, J. C., Sabet, H., Tran, T., Yu, X., Powell, J. I., Yang, L., Marti, G. E., Moore, T., Hudson, J., Lu, L., Lewis, D. B., Tibshirani, R., Sherlock, G., Chan, W. C., Greiner, T. C., Weisenburger, D. D., Armitage, J. O., Warnke, R., Levy, R., Wilson, W., Grever, M. R., Byrd, J. C., Botstein, D., Brown, P. O., and Staudt, L. M. (2000). Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769):503–511.

[Alon et al., 1999] Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, D., and Levine, A. J. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc Natl Acad Sci U S A*, 96(12):6745–6750.

[Altman and Raychaudhuri, 2001] Altman, R. B. and Raychaudhuri, S. (2001). Whole-genome expression analysis: challenges beyond clustering. *Curr Opin Struct Biol*, 11(3):340–347.

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

[Ben-Dor et al., 2000] Ben-Dor, A., Karp, R., Schwikowski, B., and Yakhini, Z. (2000). Universal dna tag systems: a combinatorial design scheme. *J Comput Biol*, 7(3-4):503–519.

[Bernard and Hartemink, 2005] Bernard, A. and Hartemink, A. J. (2005). Informative structure priors: joint learning of dynamic regulatory networks from multiple types of data. *Pac Symp Biocomput*, pages 459–70.

[Bodenreider, 2004] Bodenreider, O. (2004). The unified medical language system (umls): integrating biomedical terminology. *Nucleic Acids Res*, 32(Database issue):D267–D270.

[Buntine, 1991] Buntine, W. L. (1991). Theory refinement of Bayesian networks. In *Uncertainty in Artificial Intelligence*.

[Butte and Kohane, 2000] Butte, A. J. and Kohane, I. S. (2000). Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. *Pac Symp Biocomput*, pages 418–429.

[Carmona-Saez et al., 2006] Carmona-Saez, P., Chagoyen, M., Rodriguez, A., Trelles, O., Carazo, J., and Pascual-Montano, A. (2006). Integrated analysis of gene expression by association rules discovery. *BMC Bioinformatics*, 7(1):54.

[Cheng and Church, 2000] Cheng, Y. and Church, G. M. (2000). Biclustering of expression data. *Proc Int Conf Intell Syst Mol Biol*, 8:93–103.

[Chickering, 2002] Chickering, D. M. (2002). Learning equivalence classes of bayesian-network structures. *J. Mach. Learn. Res.*, 2:445–498.

[Chickering et al., 1994] Chickering, D. M., Geiger, D., and Heckerman, D. (1994). Learning Bayesian Networks is NP-Hard. Technical Report MSR-TR-94-17, Microsoft Research.

[Cho et al., 1998] Cho, R., Campbell, M., Winzeler, E., Steinmetz, L., Conway, A., Wodicka, L., Wolfsberg, T., Gabrielian, A., Landsman, D., Lockhart, D., and Davis, R. (1998). A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, 2:65–73.

[Cooper and Herskovits, 1992] Cooper, G. and Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347.

[Cooper and Yoo, 1999] Cooper, G. and Yoo, C. (1999). Causal discovery from a mixture of experimental and observational data. In *UAI*.

[De Bie et al., 2005] De Bie, T., Monsieurs, P., Engelen, K., Moor, B. D., Cristianini, N., and Marchal, K. (2005). Discovering transcriptional modules from motif, chip-chip and microarray data. *Pac Symp Biocomput*, pages 483–94.

[de la Fuente et al., 2004] de la Fuente, A., Bing, N., Hoeschele, I., and Mendes, P. (2004). Discovery of meaningful associations in genomic data using partial correlation coefficients. *Bioinformatics*, 20(18):3565–3574.

[Dean and Kanazawa, 1990] Dean, T. and Kanazawa, K. (1990). A model for reasoning about persistence and causation. *Comput. Intell.*, 5(3):142–150.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. of the Royal Statistical Society, Series B*, 34:1–38.

[DeRisi et al., 1997] DeRisi, J. L., Iyer, V. R., and Brown, P. O. (1997). Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 278(5338):680–686.

[D'haeseleer et al., 1998] D'haeseleer, P., Wen, X., Fuhrman, S., and Somogyi, R. (1998). Mining the gene expression matrix: inferring gene relationships from large scale gene expression data. In *IPCAT '97: Proceedings of the second international workshop on Information processing in cell and tissues*, pages 203–212, New York, NY, USA. Plenum Press.

[D'haeseleer et al., 1999] D'haeseleer, P., Wen, X., Fuhrman, S., and Somogyi, R. (1999). Linear modeling of mrna expression levels during cns development and injury.

[Druzdzel and van der Gaag, 1995] Druzdzel, M. and van der Gaag, L. (1995). Elicitation of probabilities for belief networks:combining qualitative and quantitati. In *Proceedings of the 11th Annual Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pages 141–14, San Francisco, CA. Morgan Kaufmann.

[Du, 2005] Du, P. (2005). *Multi-scale genetic network inference based on time series gene expression profiles*. PhD dissertation, Iowa State University.

[Duggan et al., 1999] Duggan, D. J., Bittner, M., Chen, Y., Meltzer, P., and Trent, J. M. (1999). Expression profiling using cdna microarrays. *Nat Genet*, 21(1 Suppl):10–14.

[Eisen et al., 1998] Eisen, M. B., Spellman, P. T., Brown, P. O., and Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *PNAS*, 95(25):14863–14868.

[Ellson et al., 2003] Ellson, J., Gansner, E., Koutsofios, E., North, S., and Woodhull, G. (2003). Graphviz and dynagraph – static and dynamic graph drawing tools. In Junger, M. and Mutzel, P., editors, *Graph Drawing Software*, pages 127–148. Springer-Verlag.

[Ewing et al., 1999] Ewing, R. M., Kahla, A. B., Poirot, O., Lopez, F., Audic, S., and Claverie, J. M. (1999). Large-scale statistical analyses of rice ests reveal correlated patterns of gene expression. *Genome Res*, 9(10):950–959.

[Filkov, 2005] Filkov, V. (2005). *Handbook of Computational Molecular Biology (Chapman & All/Crc Computer and Information Science Series)*, chapter 27. Chapman & Hall/CRC.

[Friedman and Koller, 2000] Friedman, N. and Koller, D. (2000). Being Bayesian about network structure. In *UAI*.

[Friedman et al., 1998] Friedman, N., Murphy, K., and Russell, S. (1998). Learning the structure of dynamic probabilistic networks. In Cooper, G. F. and Moral, S., editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 139–147, San Francisco. Morgan Kaufmann.

[Gardner et al., 2003] Gardner, T. S., di Bernardo, D., Lorenz, D., and Collins, J. J. (2003). Inferring Genetic Networks and Identifying Compound Mode of Action via Expression Profiling. *Science*, 301(5629):102–105.

[Gardner and Faith, 2005] Gardner, T. S. and Faith, J. J. (2005). Reverse-engineering transcription control networks. *Physics of Life Reviews*, 2(1):65–88.

[Gasch and Eisen, 2002] Gasch, A. P. and Eisen, M. B. (2002). Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. *Genome Biol*, 3(11):RESEARCH0059.

[Geman and Geman, 1984] Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. 6(6).

[Getz et al., 2000] Getz, G., Levine, E., and Domany, E. (2000). Coupled two-way clustering analysis of gene microarray data. *Proc Natl Acad Sci U S A*, 97(22):12079–12084.

[Gilks et al., 1996] Gilks, W., Richardson, S., and Spiegelhalter, D. (1996). *Markov Chain Monte Carlo in Practice*. Chapman and Hall.

[Golub et al., 1999] Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D., and Lander, E. S. (1999). Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science*, 286(5439):531–537.

[Guldener et al., 2006] Guldener, U., Munsterkotter, M., Oesterheld, M., Pagel, P., Ruepp, A., Mewes, H.-W., and Stumpflen, V. (2006). MPact: the MIPS protein interaction resource on yeast. *Nucl. Acids Res.*, 34(suppl_1):D436–441.

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

[Heckerman, 1998] Heckerman, D. (1998). A tutorial on learning with Bayesian networks. In Jordan, M., editor, *Learning in Graphical Models*. MIT Press.

[Ideker et al., 2000] Ideker, T., Thorsson, V., and Karp, R. M. (2000). Discovery of regulatory interactions through perturbation: inference and experimental design. In *Proc. of the Pacic Symp. on Biocomputing*, volume 5, pages 302–313.

[Imoto et al., 2003] Imoto, S., Higuchi, T., Goto, T., Tashiro, K., Kuhara, S., and Miyano, S. (2003). Combining microarrays and biological knowledge for estimating gene networks via bayesian networks. *csb*, 00:104.

[Jenkins and Watts, 1969] Jenkins, G. and Watts, D. (1969). *Spectral analysis and its applications*, volume 1 of *Holden-Day series in time series analysis*. Holden-Day, San Fransisco.

[Kanehisa and Goto, 2000] Kanehisa, M. and Goto, S. (2000). KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucl. Acids Res.*, 28(1):27–30.

[Kato et al., 2005] Kato, T., Tsuda, K., and Asai, K. (2005). Selective integration of multiple biological data for supervised network inference. *Bioinformatics*.

[Kohonen, 1989] Kohonen, T. (1989). *Self-organization and associative memory: 3rd edition*. Springer-Verlag New York, Inc., New York, NY, USA.

[Kondor and Lafferty, 2002] Kondor, R. I. and Lafferty, J. (2002). Diffusion kernels on graphs and other discrete structures.

[Lanckriet et al., 2004] Lanckriet, G. R. G., De Bie, T., Cristianini, N., Jordan, M. I., and Noble, W. S. (2004). A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635.

[Li et al., 2006] Li, J., Li, X., Su, H., Chen, H., and Galbraith, D. W. (2006). A framework of integrating gene relations from heterogeneous data sources: an experiment on Arabidopsis thaliana. *Bioinformatics*, page btl345.

[Li, 2007] Li, S. (2007). Integrate qualitative biological knowledge to build gene networks by parallel dynamic bayesian network structure learning. In *Proceedings of IEEE 7th International Symposium on Bioinformatics & Bioengineering (BIBE 2007)*.

[Li and Chou, 2005] Li, S. and Chou, H.-H. (2005). Ubviz: a software tool for exploring metabolic pathways in 3-d space. *Biotechniques*, 38(4):540, 542.

[Liang et al., 1998] Liang, S., Fuhrman, S., and Somogyi, R. (1998). Reveal: a general reverse engineering algorithm for inference of genetic network architectures.

[Lockhart and Winzeler, 2000] Lockhart, D. J. and Winzeler, E. A. (2000). Genomics, gene expression and dna arrays. *Nature*, 405(6788):827–836.

[MacQueen, 1967] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *5th Berkley Symposium on Mathematics and Probability*, pages 281–297.

[Margolin et al., 2006] Margolin, A., Nemenman, I., Basso, K., Wiggins, C., Stolovitzky, G., Favera, R., and Califano, A. (2006). Aracne: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, 7(Suppl 1):S7.

[Mewes et al., 2000] Mewes, H. W., Frishman, D., Gruber, C., Geier, B., Haase, D., Kaps, A., Lemcke, K., Mannhaupt, G., Pfeiffer, F., Schuller, C., Stocker, S., and Weil, B. (2000). MIPS: a database for genomes and protein sequences. *Nucl. Acids Res.*, 28(1):37–40.

[Michaels et al., 1998] Michaels, G. S., Carr, D. B., Askenazi, M., Fuhrman, S., Wen, X., and Somogyi, R. (1998). *Cluster analysis and data visualization of large-scale gene expression data*, volume 3. Pacific Symposium on Biocomputing.

[Minka, 1999] Minka, T. (1999). From Hidden Markov Models to Linear Dynamical Systems. Technical report, MIT.

[Murphy, 2002] Murphy, K. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD dissertation, University of California, Berkeley.

[Murphy and Mian, 1999] Murphy, K. and Mian, S. (1999). Modelling gene expression data using dynamic bayesian networks.

[Nariai et al., 2004] Nariai, N., Kim, S., Imoto, S., and Miyano, S. (2004). Using protein-protein interactions for refining gene networks estimated from microarray data by Bayesian networks. *Pac Symp Biocomput*, pages 336–47.

[Pearl, 2000] Pearl, J. (2000). *Causality: Models, Reasoning and Inference*. Cambridge Univ. Press.

[Pearl and Verma, 1991] Pearl, J. and Verma, T. (1991). A theory of inferred causation. In *Knowledge Representation*, pages 441–452.

[Rabiner, 1989] Rabiner, L. R. (1989). A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–286.

[Robinson, 1973] Robinson, R. W. (1973). Counting labeled acyclic digraphs. In Harary, F., editor, *New Directions in the Theory of Graphs*, pages 239–273. Academic Press.

[Rojdestvenski, 2003] Rojdestvenski, I. (2003). Metabolic pathways in three dimensions. *Bioinformatics*, 19(18):2436–2441.

[Roweis and Ghahramani, 1999] Roweis, S. and Ghahramani, Z. (1999). A Unifying Review of Linear Gaussian Models. *Neural Computation*, 11(2).

[Sasik et al., 2001] Sasik, R., Hwa, T., Iranfar, N., and Loomis, W. F. (2001). Percolation clustering: a novel approach to the clustering of gene expression patterns in dictyostelium development. *Pac Symp Biocomput*, pages 335–347.

[Schmitt et al., 2004] Schmitt, W. A., Raab, R. M., and Stephanopoulos, G. (2004). Elucidation of gene interaction networks through time-lagged correlation analysis of transcriptional data. *Genome Res*, 14(8):1654–1663.

[Schwarz, 1978] Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464.

[Segal et al., 2001] Segal, E., Taskar, B., Gasch, A., Friedman, N., and Koller, D. (2001). Rich probabilistic models for gene expression.

[Segal et al., 2003] Segal, E., Wang, H., and Koller, D. (2003). Discovering molecular pathways from protein interaction and gene expression data. *Bioinformatics*, 19(90001):264i–272.

[Shatkay et al., 2000] Shatkay, H., Edwards, S., Wilbur, W. J., and Boguski, M. (2000). Genes, themes and microarrays: using information retrieval for large-scale gene analysis. *Proc Int Conf Intell Syst Mol Biol*, 8:317–328.

[Shmulevich et al., 2002] Shmulevich, I., Dougherty, E. R., Kim, S., and Zhang, W. (2002). Probabilistic Boolean Networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, 18(2):261–74.

[Simon et al., 2001] Simon, I., Barnett, J., Hannett, N., Harbison, C. T., Rinaldi, N. J., Volkert, T. L., Wyrick, J. J., Zeitlinger, J., Gifford, D. K., Jaakkola, T. S., and Young, R. A. (2001). Serial regulation of transcriptional regulators in the yeast cell cycle. *Cell*, 106(6):697–708.

[Spellman et al., 1998] Spellman, P., Sherlock, G., Zhang, M., Iyer, V., Anders, K., Eisen, M., Brown, P., Botstein, D., and Futcher, B. (1998). Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. *Mol Biol Cell*, 9(12):3273–97.

[Spirtes et al., 2000] Spirtes, P., Glymour, C., and Scheines, R. (2000). *Causation, Prediction, and Search*. MIT Press. 2nd edition.

[Tamayo et al., 1999] Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., Lander, E. S., and Golub, T. R. (1999). Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. *Proc Natl Acad Sci U S A*, 96(6):2907–2912.

[Toronen et al., 1999] Toronen, P., Kolehmainen, M., Wong, G., and Castrn, E. (1999). Analysis of gene expression data using self-organizing maps. *FEBS Lett*, 451(2):142–146.

[Troyanskaya, 2005] Troyanskaya, O. G. (2005). Putting microarrays in a context: Integrated analysis of diverse biological data. *Brief Bioinform*, 6(1):34–43.

[Verma and Pearl, 1990] Verma, T. and Pearl, J. (1990). Equivalence and synthesis of causal models. In *UAI*.

[Wittig and Jameson, 2000] Wittig, F. and Jameson, A. (2000). Exploiting qualitative knowledge in the learning of conditional probabilities of bayesian networks. In *UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 644–652, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Yamanishi et al., 2004] Yamanishi, Y., Vert, J.-P., and Kanehisa, M. (2004). Protein network inference from multiple genomic data: a supervised approach. *Bioinformatics*, 20(Suppl 1):i363–370.

[Yin et al., 2004] Yin, L., hsi Huang, C., and Rajasekaran, S. (2004). Parallel data mining of bayesian networks from gene expression data. *Poster Book of the 8th International Conference on Research in Computational Molecular Biology(RECOMB)*, pages 122–123.