A MARKOV BLANKET BASED CAUSAL MODEL FOR RECONSTRUCTION OF GENE REGULATORY NETWORK

by

Ramesh Ram
B. Tech. (IT), Anna University, Chennai, INDIA

Submitted by Ramesh Ram in fulfillment of the requirements for the Degree of

Doctor of Philosophy

Main Supervisor: Dr Madhu Chetty

Gippsland School of Information Technology

Monash University, Australia

November, 2009

Dedicated to Lord Ganesha, my family, and well-wishers.

Om Gam Ganapathaye Namah!

A MARKOV BLANKET BASED CAUSAL MODEL FOR RECONSTRUCTION OF GENE REGULATORY NETWORK

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and references given.

[redacted]

Ramesh Ram Date: 27-11-2009

Table of Contents

Abstract

Understanding the interactions of genes plays a vital role in the analysis of complex biological systems. The gene regulatory networks (GRNs) are representations of gene-gene regulatory interactions in a genome and display relationships between various gene activities. GRN modeling and inference is carried out mainly with the help of gene expression microarray data. The microarray data is characterized as massive, heterogeneous and high-dimensional in nature. In a typical dataset, the number of samples n (with an order of tens) is substantially smaller than the number of genes p (with an order of hundreds or even thousands) which makes it very difficult to reconstruct a GRN from this data.

The aim of the thesis is develop a novel causal model for GRN inference which exploits the naturally existing causal gene interactions (i.e. expression of gene Y is caused by interaction with another gene X) thereby resulting in higher accuracies in reconstruction. The method is based on the decomposition of the entire GRN into sub-networks which are basically the Markov Blankets (MB) of each gene. The causal GRN model is accomplished by applying a minimal set of constraints which reduces the extremely large search space to a smaller set of possible models. These reconstructed networks are pruned further to eliminate false positives resulting in minimal connectivity and best fit GRN for the data.

Synthetic datasets allow validating new techniques and approaches since the underlying mechanisms of the GRNs, generated from these datasets, are completely known. The realistic synthetic datasets validate the robustness of the method by varying topology, sample size, time-delay, noise, vertex in-degree and presence of hidden nodes. We present a novel approach for synthetically generating gene networks using causal relationships.

To accurately and efficiently reverse engineer the gene network from time-course expression data, a Guided Genetic Algorithm (GGA) is developed to carry a heuristic search through the space of qualitative causal networks incorporating the causal relationships between genes. The GGA exploits characteristics of diversity and high level heuristics to generate fit networks quickly (less iterations) and is shown to have a superior performance compared to simple GA (SGA) that is currently applied by researchers. Building upon GGA, we further improve search process by another new technique, which we refer as FOMBGA (Frequently Occurring Markov Blanket Genetic Algorithm). The FOMBGA replaces crossover and mutation operators with a probabilistic model on frequency of occurrence of fit Markov blankets (MBs).

Estimation of GRN parameters is basically the estimation of conditional probability distributions (CPD) of the given GRN. Due to high dimensional data, exact computation of the CPDs is infeasible and computationally expensive. In the thesis work presented, given the network structures, we deduce a unique minimal I-map of the GRN by estimating the conditional probability distribution of each variable (gene) from the data set. This is achieved by using a novel variant of the Markov Chain Monte Carlo (MCMC) method whereby the search space is gradually reduced resulting in the convergence to occur quickly and in a reasonable computation time. The performance of the parameter estimation technique is further improved by integrating regulatory sequence motif data and GO annotations.

Investigations are carried out using both the synthetic dataset as well as yeast cell-cycle gene expression datasets. Experiments carried out show that the proposed modeling approach has excellent inferential capabilities and high accuracy even in the presence of noise. The gene network inferred from yeast cell-cycle data is investigated for its biological relevance using well-known interactions, sequence analysis, motif patterns and GO data available in literature. The studies resulted in discovering the known interactions and predicting novel interactions.

Acknowledgements

I wish to convey my heartfelt thanks to my supervisor, Dr. Madhu Chetty for his guidance and support throughout my student years at Monash University. His encouragement has been essential in giving me the time and space, which has allowed me to discover my research interests and passions and mature as a researcher. I owe special thanks to my associate supervisors, Dr. Trevor Dix and Dr. Dieter Bulach, for their useful and constructive comments and advice. Further, if not for my family, we would not be able to complete this work. My parents have been supportive of me in all my endeavors, thanks to mom, dad and my sister. I would also like to thank my friends and fellow PhD students at the department who have been very resourceful and have made my life enjoyable. Last but not least I would like to thank Head of the School and all the staff at the Monash University GSIT department who have been extremely helpful and especially creating this ideal environment for pursuing my career.

List of Publications

Journal Articles

1. Ram, R., and Chetty M., (2009) "A Markov Blanket-Based Model for Gene Regulatory Network Inference," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, (available online) http://doi.ieeecomputersociety.org/10.1109/TCBB.2009.70
2. Ram, R., and Chetty M., (2009) "A Markov blanket-based Probabilistic Genetic Algorithm for Causal Reconstruction of Gene Regulatory Networks", *BioSystems,* Elsevier (under review after second revision).

Book Chapter

3. Ram, R., and Chetty, M., (2010) Modeling gene regulatory networks using computational intelligence techniques. In Das, S., *et al.* (eds), *Handbook of research on Computational Methodologies in Gene Regulatory Networks.* Medical Information Science Reference, IGI Global USA, 244- 264. ISBN: 978-1-60566-686-0

Peer Reviewed Conference Articles

4. Ram, R., and Chetty M., (2009) MCMC based Bayesian Inference for Gene Networks, *Proc. of 4th IAPR Conference on Patterm Recognition in Bioinformatics, Lecture Notes in Bioinformatics,* LNBI5780, pp. 293-306, Sheffield, UK.

5. Ram, R., and Chetty M., (2008) Constraint Minimization for Efficient Modeling of Gene Regulatory Network, *Proc. of 3th IAPR Conference on Patterm Recognition in Bioinformatics, Lecture Notes in Bioinformatics,* LNBI5265, pp. 201 – 213, Melbourne, Australia.
6. Ram, R., and Chetty M., (2008) Generating Synthetic Gene Regulatory Networks, *Proc. of 3th IAPR Conference on Patterm Recognition in Bioinformatics, Lecture Notes in Bioinformatics,* LNBI5265, pp. 237-249, Melbourne, Australia.
7. Ram, R., and Chetty,M. (2007) Framework for path analysis for learning Gene regulatory network, *Proc. of 2th IAPR Workshop on Patterm Recognition in Bioinformatics, Lecture Notes in Bioinformatics,* LNBI4774, pp. 264-273, Singapore.
8. Ram, R., and Chetty,M. (2007) A Guided genetic algorithm for Gene Regulatory Network. *IEEE Congress on Evolutionary Computation*. Singapore, pp. 3862-3869.
9. Ram, R., and Chetty,M. (2007) Learning Structure of Gene Regulatory Networks. *6th IEEE International Conference on Computer and Information Science*. Melbourne, Australia, pp. 525-531.
10. Ram, R., Chetty, M. and Dix,T.I., (2006) Fuzzy Model for Gene Regulatory Networks. *IEEE Congress on Evolutionary Computation*. Vancouver, Canada, pp. 1450-1455.
11. Ram, R., Chetty,M., and Dix,T.I. (2006) Causal Modeling of Gene Regulatory Network. *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*. Toronto, Canada, pp. 1-8.

List of Figures

List of Tables

Notations

Symbol	**Description**
D	Main data set
n	Number of samples in data set
X	One-dimensional variables (genes)
x	Values of corresponding variables (genes)
Z	Universe, set of variables / nodes / genes in the domain
N	Number of variables
E	Set of edges of a Bayesian Network (BN)
P	Set of parameters of local probability density functions (pdfs) for the entire BN
E	Number of edges of the BN
G	Directed acyclic graph (DAG) of a BN
BN	Bayesian network, consists of DAG and parameters
$MB(X)$	Markov blanket of variable X
$H(X)$	Set of direct neighbors of variable X in Bayesian network
$Pa(X)$	Set of parents of X
pa_j	Set of values for where j corresponds to each member of the parents set $Pa(X)$
q_i	Number of values of the variable X_i
$r_{x,y}$	Correlation between variables X and Y

$r_{xy.z}$	Partial correlation between variables X and Y given variable Z
$r_{xy.zw}$	Second order partial correlation between variables X and Y given variables Z and W
θ	Set used to store the conditional probability distribution
$\theta(G)$	Maximum Likelihood (ML) of graph G
$\perp\!\!\!\perp$	Independence between variables
$\not\perp\!\!\!\perp$	Dependence between variables
E_s	Fitness Score of a sub-model s
$\exists\,(X, Y)$	To show there is an edge between variables X and Y
ψ	Significance threshold value of partial correlation
V	Set of vertices of a BN
Acronyms	**Descriptions**
BN	Bayesian Network
DAG	Directed Acyclic Graph
GRN	Gene Regulatory Network
CPD	Conditional Probability Distribution
SGA	Simple Genetic Algorithm
GGA	Guided Genetic Algorithm
CLM	Constraint Logic Minimization
K-Map	Karnaugh map
D-sep	Directed Separation
MB	Markov Blanket
FOMBGA	Frequently Occurring Markov Blanket Genetic Algorithm
MCMC	Markov Chain Monte Carlo
GO	Gene Ontology

Chapter 1

1. Introduction

1.1 Introduction

In the post-genomic era, holistic understanding of biological systems in all their complexity is critical in comprehending nature's way of creating life. To create a complete biological system, the biological processes and systems can be abstracted as multi-layered networks [1, 2] interacting with each other as shown in Fig. 1.1. Among these interactions, understanding the interactions amongst genes plays a vital role in the analysis of complex biological systems. Thus, gene regulatory networks (GRNs) are of tremendous importance in uncovering the underlying biological process of living organisms, providing new ideas for treating complex diseases, and the designing of new drugs. To understand and explain the underlying mechanisms of a GRN, we next present a brief description of important terms.

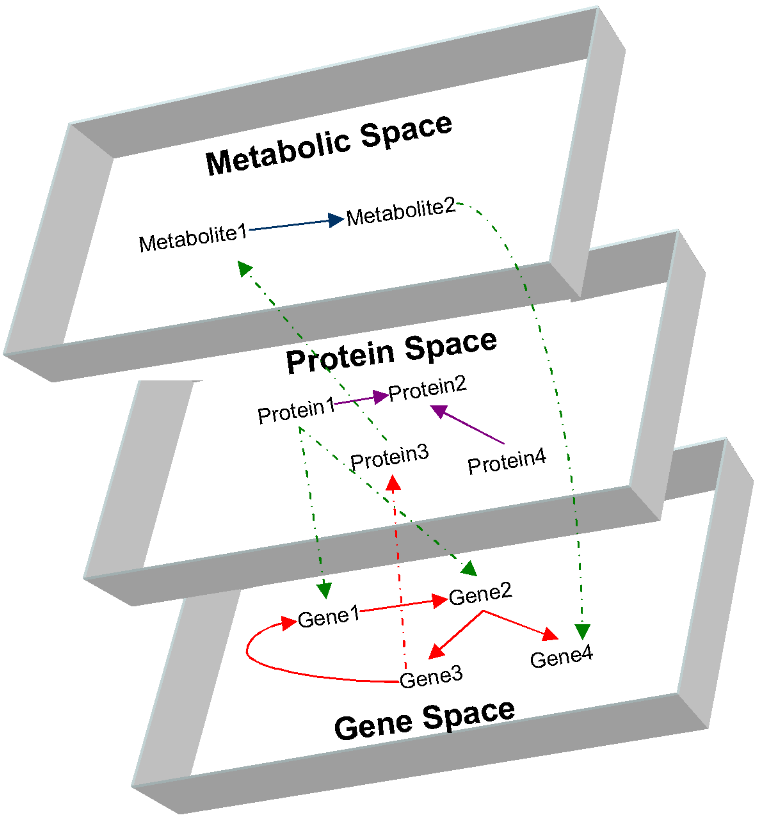


Fig. 1.1 Fundamental Interactions

1.1.1 Central dogma of molecular biology

Although a cell is the fundamental unit of all living organisms, it is complicated in terms of both its structure and function. Such complexities are mainly embodied in and regulated by three biological sequences: DNA, RNA and Protein. The DNA (DeoxyriboNucleic Acid) is a linear, double stranded polymer in which the monomer subunits are four chemically distinct nucleotides (Adenine (A), Cytosine (C), Guanine (G), Thymine (T)).

DNA is the carrier of genes and other regulatory information. A gene is a piece of DNA fragment which contains genetic information. The whole set of genes in a cell, called the genome, defines the structure and function of the cell. The functions described by genes get implemented via proteins, which are linear polymers composed of 20 different types of amino acids. Proteins play a central role in virtually all aspects of cell structure and functions. The RNAs (Ribonucleic Acid) copy the genetic information of a gene after which some RNAs translate the information into proteins. This RNA is called as messenger RNA (mRNA). The process of flow of genetic information from a gene to mRNA and a protein is called the *gene expression*. In this process, the DNA serves as the template to make RNA. This process is known as *transcription*. The RNA then serves as the source of information to make proteins in a process called *translation*. The process of conversion of DNA to protein is known as the *"central dogma in molecular biology"*. This is shown in Fig. 1.2. While the sequence (and its function) of a protein is defined by the sequence of a corresponding gene in nature, the expression of the gene is regulated by an expression of set of parent genes which are known as *transcription factors*. This process is called *gene regulation*.

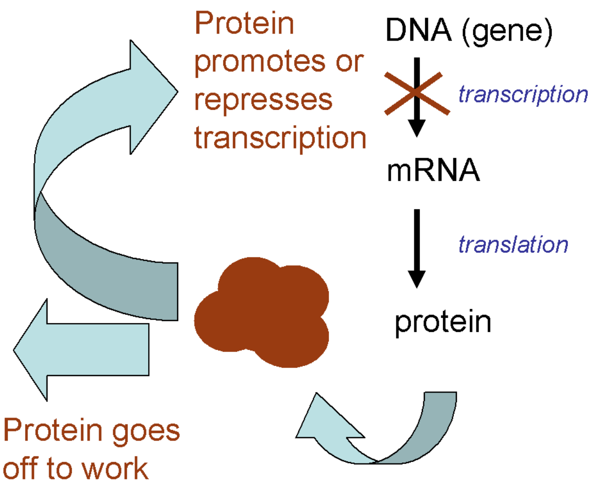
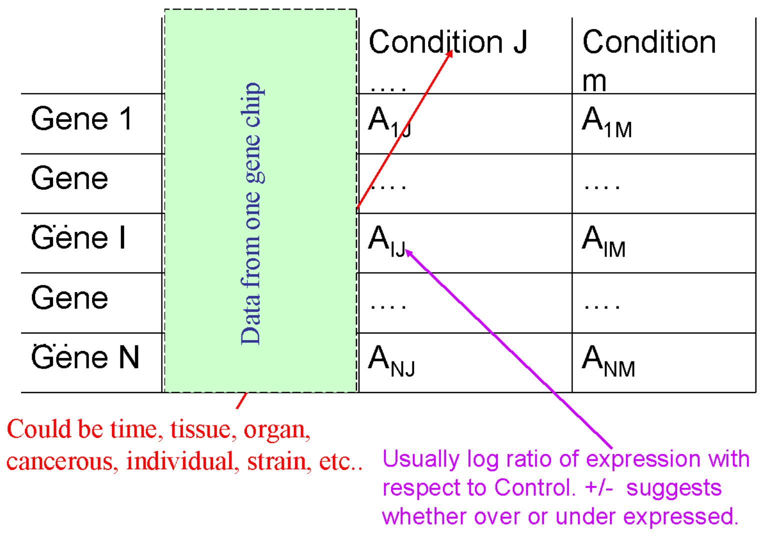


Fig. 1.2 Central Dogma of Molecular Biology

Gene expression is regulated both temporally and spatially [3, 4]. The temporal expression of a gene refers to the process that a gene expresses (or is regulated) at the appropriate time and keeps itself silent otherwise [4, 5]. It also indicates a gene has different expression patterns at different times [6]. For example, the expression patterns of tumor suppressor gene *p53* are different at different stage in modulating cellular functions such as DNA repair, cell cycle arrest, and apoptosis.

There is also spatial control of gene expression [4, 7]. Although cells from the same organism have identical genomes, cells in the different parts of an organism may have different gene expression patterns due to the various functions they fulfill. Therefore, the regulation of gene expression is an essential part of life [8-10]. There are two types of regulations: positive and negative. Given two genes *X* and *Y*, if an expression level of *Y* is affected by the expression level of *X*, we say *X* regulates *Y*. If an increase in the expression level of *X* leads to an increase of expression level of *Y*, it is a positive regulation; otherwise, it is a negative expression. DNA microarray technology allows us to measure the amount of RNA associated with many genes in parallel (using gene chips) and determine which are expressed in a particular cell type. The data accumulated using the microarray technology is commonly represented as shown in Fig. 1.3. The rows represent the genes and the columns represent the conditions which can be either temporal or spatial.



Data from one gene chip
Condition J ....
Condition m
Gene 1
Gene
Gene I
Gene
Gene N
A_{1J}
A_{1M}
....
A_{IJ}
A_{IM}
A_{NJ}
A_{NM}
Could be time, tissue, organ, cancerous, individual, strain, etc..
Usually log ratio of expression with respect to Control. +/- suggests whether over or under expressed.

Fig. 1.3 Example Gene Expression Data

Regulatory events in a set may depend on each other and form some regulatory chains, named regulatory pathways. Moreover, a chain may form a loop. There are two types of loops: negative and positive. If the sum of the negative regulations of the loop is odd, it is a negative loop; otherwise, it is positive. Regulatory loops maintain the stability or development of cells [11, 12] and are necessary to understand the gene regulation system.

1.1.2 Gene networks

The network of relationships among the genes is known as a *gene network* or a gene regulatory network (GRN). It is a representation of gene-gene regulatory interactions in a genome that display relationships between various gene activities. It is estimated that each gene on average interacts with four to eight other genes, and is involved in 10 biological functions [3]. The illustrative Fig. 1.4 shows the regulatory relationships (represented by arrows) between pairs of genes to form a gene regulatory network of 6 genes. The complexity of a living cell is achieved by the concentrated activity of many genes and their products (proteins). This activity is often coordinated by the organization of the genome into regulatory modules, or sets of co-regulated genes that share a common function [13]. The global gene expression pattern is therefore the result of the collective behavior of individual regulatory pathways. In such highly interconnected cellular signaling networks, gene functions depend on the cellular context.

Genes (proteins) work together as a team to accomplish certain processes that no single protein can do alone, such as metabolism, detoxification, and various responses to the environment. This partially explains why many novel "gene targeted" drugs have failed during clinical trials because of side effects and poor specificity. Thus, understanding a gene network as a whole is essential, and learning gene networks is an important central theme in post genomic research [10, 14-16].

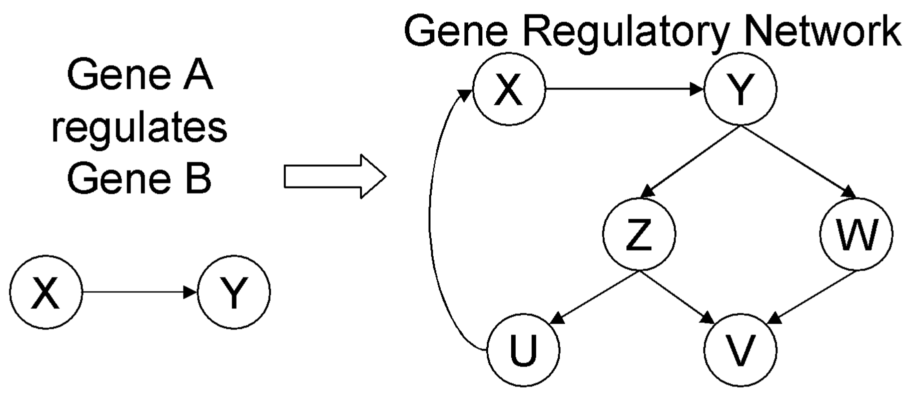


Fig. 1.4 Gene network

This thesis is devoted to the study of gene networks. There can be several applications and advantages of this study, for example,

- Gene networks provide a large-scale, coarse-grained view of the physiological state of an organism at the mRNA level [16]. Gene networks describe a large number of interactions in a concise way. They also present the dynamic properties of the gene regulatory system. They are capable of being the annotation of genomics and functional genomics data.
- It is an important step to uncover the complete biochemical networks of cells [17].
- Knowledge about gene networks might provide valuable clues for the therapeutics of complex diseases [18-20].
- As most phenotypes are the result of the collective response of a group of genes, gene networks help to explain how complex traits arise and which groups of genes are responsible for them [21, 22].
- Gene networks are well suited for comparative genomics [23]. Comparing gene networks from different genomes helps with the understanding of evolution.

1.2 Motivation

The early work on understanding of gene regulation had taken a biological experimental approach. This traditional approach was inherently local: examine and collect data on a single gene, a single protein or a single reaction at a time, then analyze the binding sites and reactions one by one. With this approach, it would normally take one or more years to discover the regulation of a gene. Over the years, this 'manual' lab-based approach did make remarkable achievements, and allowed in inferring highly accurate biochemical models of individual genes for small sized genome, such as the bacteria phage lambda [24, 25].

However, taking into account the huge information stored in a genome (there are thousands or even tens of thousands of genes in a genome [26]), it is far from possible to construct a network by the conventional way. The number of experiments that are necessary for constructing a network are far too many. In addition, a network learnt by the experimental approach can only describe the regulation relationship but does not have the ability to predict the properties which have not been observed. To obtain a full picture of even a medium size genome using the experimental approach is not only time consuming but also expensive [27, 28]. Therefore, it is unrealistic to obtain a holistic understanding of a regulation by analyzing regulation pathways one by one [29].

With the development of high-throughput genomics and functional genomics, massive data on thousands of cellular species are being gathered. This is a significant shift from the traditional molecular biology approach of focusing on single molecules and reactions. The need is now data-driven, and there is great urgency to find methods that can handle the massive data in a global manner and that can analyze large systems at some intermediate level [7].

A gene network can be represented by various mathematical models. A model is a representation of reality used to simulate a process, understand a situation, predict an outcome, or analyze a problem. The success of the computational approach in learning gene networks has been proven biologically, with many exciting results reported in recently published literature [30-35] and reviewed in detail in next chapter. Furthermore, the models have levels of abstractions and have assumptions for simplifying the problem under study.

Early computational approaches were based on learning the relationships among genes either by studying mutual information or the correlation among their expression values. The representatives of such approaches are pair-wise interaction [35, 36] and clustering [13, 37, 38], which seek to directly find correlations among genes. Since then, Boolean networks [39, 40] have been used in several works, where gene expression levels are represented by Boolean values and the gene regulatory relationship is represented by a set of Boolean functions. Linear and nonlinear models followed, and they represent regulatory relationships by linear functions and non-linear functions.

Recently, Bayesian networks are being investigated for modeling GRNs. In the well-known seminal paper by Friedman *et al.* [17], an algorithm for learning gene networks using the Bayesian network was presented. Since then, several extensions of the Bayesian network have been proposed, such as the Bayesian network integrated with nonparametric regression [41], Dynamic Bayesian network (DBN) [42-45], etc. However, to the best of our knowledge, these Bayesian models had some critical limitations including,

- Biologically significant results were obtained from small datasets.
- Gene networks were learnt by making time series gene expression data discrete rather than continuous
- Scalability: failed to learn gene networks from medium or large datasets due to inefficient search techniques.

- Some important biological factors (including: variable time delays in the gene regulatory system, the effect of noise (both biological and technical) in the prediction of gene regulation, various topological structures, the number of parent genes per gene in the network, and so on) were overlooked.
- Parametric estimation was erroneous due to small sample sizes available in the real dataset.

1.3. Aims and Objectives

To resolve the problems outlined above, we set out the following aims and objectives of this thesis:

- To develop a novel Bayesian model (referred as causal Bayesian model) for modeling gene regulatory network and identifying regulatory interactions.
- To refine the model further to identify spurious relationship in the model by the application post processing path analysis technique
- To speed up the process and support large scale data by minimizing the computation overhead.
- To develop synthetic benchmark datasets for testing and validating the model for parametric variations such as noise, number of samples, delay, structure, number of parent genes and scalability.
- To develop fast and efficient search technique that can explore the space of gene networks efficiently and accurately.
- To investigate further model enhancement by sampling and estimation of parameters and deducing a minimal network with minimal set of important interactions
- To improve biological accuracy of the model by integration of the prior parameters using data from sequence analysis, GO and motif analysis.

1.4 Contributions

To achieve the aims and objectives outlined above, we proposed and developed the following model and related techniques in this thesis:

- *A causal model learning framework based on the Bayesian network:* This framework represents various aspects such as network structure, direction, time delay and sign/ orientation of regulation (i.e. up/down regulation). We present a method where the predicted network is decomposed into triplets of genes, and causal inferences using partial correlation is applied in order to detect whether or not connections are direct or indirect with either partial or full or no effect (explanation). Partial correlation constraints allow us to recover much of the network structure given the data. The direction and sign of regulation are recovered by estimating the time delay and time correlation between expression profiles of pairs of genes. All the aspects are modeled as flexible scoring metric which reflects the goodness of fit of a putative structure to the data. A set of improvements are made to increase the efficiency and accuracy of the learning process: (i) an improved Markov blanket based approach is presented which decomposes the network into Markov blankets of each gene (comprising of parents, children and parents of children); (ii) a new structure learning search algorithm which is suited to learning gene networks; (iii) a constraint minimization technique to speed up in the case of large datasets. In addition, unlike the traditional Bayesian network, the proposed framework uses real continuous values rather than discrete values of the data.

- *Path Analysis post processing Method:* Traditional Bayesian network learning methods that use partial correlations to help find regulatory relationships [46] can predict meaningful relationships. However, it is reported that nearly 80% regulation relationships can be found to be false positive or spurious due to the nature of the gene expression data being noisy and high dimensional. Hence a barrier exists, preventing the use of partial correlations to infer large regulatory pathway [47]. We propose a path analysis algorithm as a post processing step to the learning step. The path analysis is based on conditional dependence/independence d-separation rule together with rules based on alternative hypothesis of paths and time delays to extract more regulatory relationships. Basically, if two genes X and Y are connected or regulated by a path Z, a series of tests are carried out on the path Z for its validity and as a combined effect of the tests a decision is taken as to whether or not the path needs to be pruned for prevented spuriousness. This idea enables us to find regulations that are highly accurate.

- *Synthetic data generation:* As there are no suitable benchmark datasets available for validation and evaluation of GRN reconstruction techniques, it becomes difficult to analyze the model for its robustness and to compare against other techniques. Synthetic datasets allow validating new techniques and approaches since the underlying mechanisms of the GRNs, generated from these datasets, are completely known. We present an approach for synthetically generating gene networks using causal relationships. The synthetic networks can have varying topologies such as small world, random, scale free, or hierarchical topologies based on the well-defined GRN properties. These artificial but realistic GRN networks provide a simulation environment similar to a real-life laboratory microarray experiment. These networks also provide a mechanism for studying the robustness of the proposed causal model reconstruction method to individual and combination of parametric changes such as topology, noise (background and experimental noise) and time delays.

- *Frequently occurring Markov Blanket Genetic Algorithm technique:* A Bayesian network is represented as a directed acyclic graph (DAG). As the number of possible BN structures fitting microarray data is astronomically large and the search problem is NP-hard, the strategies for structure search have to be advanced and robust. Therefore, an efficient search technique is an important factor in learning gene network and demands a tailored search mechanism. We identified the complexities involved in developing such a search technique and then developed a Guided Genetic Algorithm (GGA). The GGA uses a high level guided crossover and mutation operators, a diversity switch, a rank based knowledge acquisition process and ambiguity decision maker. Further improvement was made on this algorithm to sample a population using a probability distribution. This learning algorithm was called *Frequently occurring Markov Blanket Genetic Algorithm (FOMBGA)* as the proposed technique uses the frequency of occurrence of Markov Blankets to estimate the probability distribution in extracting the underlying structure of a gene network. Compared to the GGA, this method is more appropriate for learning a large sized gene network as it is not random sampling.

- *Parameter estimation and Integration of related data:* A Markov chain Monte Carlo (MCMC) algorithm is proposed to infer the parameters of the Bayesian gene networks obtained using the search technique. Due to the size and quality of the data that is currently available, the search results in many plausible structures that equally satisfy the dataset. Therefore, the parameter estimation process is challenging. Markov Chain Monte Carlo (MCMC) method is run parallel across three best network structures obtained from the search technique. The samples accumulated in a combinatorial fashion. A Markov Blanket based ranking technique is used for the order in which the samples are drawn. The parameter space is iteratively reduced by clamping values to converged genes resulting in the required convergence within a reasonable computation time. We then estimate the conditional probability distribution of each variable (gene) from the samples data set and deduce a unique minimal I-map (Independence map). The inferred GRN is closer to the real gene regulatory process and gives better learning performance. An uniform prior probability is used in the estimation of probabilities of each gene. To make the reconstruction more realistic, related data such as data from sequence analysis and Gene Ontology annotations are combined and used to vary the prior probabilities of the genes based on their position on the network and their biological significance.

1.5 Organization of the thesis

The various chapters in this thesis contain a number of related themes for the investigation of modeling GRN. As a common goal, these chapters together present an effort for investigating both the structure and parametric learning for the modeling of GRN based on the causal modeling approach. Chapter 2 gives a detailed literature review covering the research area. Some existing models and algorithms for learning gene networks are introduced. Chapter 3 models a gene network as a Bayesian network composed of Markov Blanket's that represent parents, children and neighbors of each gene, specifies the time delays and the orientation (positive/negative) of regulation. Chapter 4 deals with the generation of synthetic datasets and robustness evaluation of the modeling technique. Chapter 5 describes the use of guided search technique in learning gene networks. Chapter 6 describes a Markov chain Monte Carlo sampling technique used to estimate the parameters of the model and the integration of related biological data. Chapter 7 concludes the thesis and gives some further perspectives to the project.

Chapter 2

2. Background and related work

2.1 Introduction

As early as 1969, mathematical formalism was proposed to describe gene regulatory networks [24]. Traditionally, the emphasis has been on simulation techniques [10, 48] instead of structure reconstruction. With more experimental data available, automatic structure reconstruction techniques are gaining popularity. In recent years, the number of papers in learning gene networks has grown exponentially. In this chapter, we briefly review the mathematical techniques in reconstructing gene networks from microarray gene expression data. But, firstly we look at the microarray technology which is used in the production of gene expression data.

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

2.1.1 Microarray Technology

Microarray technology is a high throughput experimental method, where mRNA expression levels of a number of genes can be measured simultaneously on a single chip. The underlining principle of microarray technology is base-pairing (i.e., A-T and G-C for DNA; A-U and G-C for RNA). Probes with known identity are planted on the microarray chips in very high density, and used to determine complementary binding. The expression of each gene is reflected by the accumulation level of the corresponding mRNA. There are two major application forms of microarray technology:

(i) Identification of sequence, and
(ii) Determination of expression level of genes.

In genetic network inference, the microarray is used to measure the gene expression levels. There are two variants of the microarray technology: The first method is traditionally called cDNA microarray, or spotted microarray.

For a cDNA microarray, probe cDNA is immobilized to a solid surface such as glass using robot spotting and exposed to a set of targets either separately or in a mixture. Usually two samples, dyed with different dyes (Cyanine 3 and Cyanine 5), are hybridized to a single slide. One of the samples is treated as reference. The dyes fluoresce at different wavelengths, so it is possible to get separate images for each dye. The colour strength of each spot image on the microarray slide reflects the mRNA accumulation level of the particular gene corresponding to the spot probe. The ratio of the colour strength of two dyes reflects the relative change of mRNA accumulation levels between the sample and the 11 reference sample. Data analysis of cDNA microarray data is usually based on the colour strength ratios of the two dyes. The second method uses DNA chip, also called Affymetrix Gene Chips. In this method, an array of oligonucleotide or peptide nucleic acid (PNA) probes is synthesized either on-chip or by conventional synthesis followed by on-chip immobilization. The array is exposed to labelled sample DNA, hybridized, and the identity/abundance of complementary sequences is determined. Unlike the cDNA microarray, Affymetrix only use one sample during hybridization, and the colour strength of the dye reflects the relative level of mRNA accumulation. The manufacture and design of Affymetrix chips is more complex than cDNA microarray. The main shortcoming of microarray is that the measured values are not quite accurate, i.e., microarray data is noisy.

The measured expression values are a highly asymmetric distribution, and large variations in expression values can lead to inferring spurious causal relationships. Log transformation is one major preprocessing step to gene expression data as the distribution of data values is approximated as symmetric and normal. There are two main types of gene expression microarray data: static and time series microarray data. In static expression experiments, a snapshot of the expression of genes in different samples at a given instant in time is measured while in time series expression experiments, the expression values over a period of time are measured.

For this thesis work, we choose time series data source to learn a gene regulatory network since gene expression itself is a temporal process [49].

The rest of the chapter is organized as follows. Section 2.2 briefly presents the various models used in the reconstruction of GRN. In Section 2.3 the various methods of learning Bayesian network from data is presented. In Section 2.4, special emphasis is laid on the Bayesian network modeling used for reconstruction of GRN. Finally Section 2.5 gives the summary of the chapter.

2.2 Literature Review

2.2.1 Pair-wise Methods

Pair-wise methods seek to discover the relationships among genes solely by pair-wise comparisons. They do not take into account interactions where the expression of one gene is achieved by the combined effects of multiple other genes. Arkin *et al.* [25, 50] proposed correlation metric construction (CMC). CMC computes the magnitude of gene pairs by cross-correlation. A distance matrix is constructed for each gene pair by comparing their similarities to other genes. Then a diagram is constructed to summarize the strength of interaction and predict mechanistic connections between the genes. Chen *et al.* [51] proposed activation/inhibition networks to find regulation based on whether peaks in one signal precede peaks in another signal. Chen *et al.* [51] proposed grouping the genes with similar expression profiles. Then a prototype is generated for each group of genes by averaging the expression values of genes in the group. Each prototype represents a group of genes with similar expression patterns and is represented as a series of peaks. The correlations between prototype pairs are calculated to determine the type of regulatory relationships (activation, inhibition or unmatched) and measure the strength of the regulatory relationship between any two prototypes. Finally, the regulation matrix is generated by the scores.

2.2.2 Clustering

One of the main problems that hinder research on gene network reconstruction is the dimension problem, i.e. there are many genes with a few replicates. A useful approach is to cluster genes with similar expression patterns into clusters, then infer the regulatory relationship among the clusters. Researchers believe genes with similar expression patterns have similar functions or are involved in the same biological events [37]. Currently, several clustering methods are used for this purpose. Different clustering methods can generate very different results. Each combination of distance measurement and clustering algorithm tends to emphasize a different type of regularities in the data. There is no single criterion for choosing the best clustering method. Given clusters, there are also several methods to find the interactions among them. Wahde and Hertz [16] clustered 65 genes from rat CNA datasets into four waves using the FITCH hierarchical clustering algorithm. Then, by a genetic algorithm, they built a four-node continuous time recurrent neural network. Chen *et al.* [52] reduced 3131 yeast genes into 308 clusters by average linkage clustering.

Later, simulated annealing was to optimize a qualitative network based on the timing of peaks in the data. Someren [15] reduced 2467 yeast genes into clusters and represented each cluster by a "prototype" gene calculated from the cluster. A linear model of the prototype genes is then generated by linear regression. D'Haeseleer and others [13, 36, 38, 47, 53] proposed grouping genes into clusters, and then find the representative genes for the clusters. The connections among the representative genes are modeled by differential equation. Toh *et al.* [54] proposed averaging the gene expression values of each cluster, and then discover the regulatory relationships by Graphical Gaussian Modeling (GGM). During the initial stages of the thesis, we experimented with a novel fuzzy-logic clustering method and the results were reported in [55]. Since then there have been several improvements based on our work published by various authors in this field.

In summary, the pros and cons of clustering are:

- *Pros*: Groups together genes with similar expression patterns
- *Cons*: Does not reveal structural relations between genes

2.2.3 Boolean model

The Boolean network model is the simplest network model, and was first proposed by Kauffman [24]. It uses a binary variable to define the state of a gene and uses Boolean functions (AND, OR, NOR, NAND) to define the gene relationships. In a simplified way, gene expression level can be roughly represented as a binary state: either active (i.e. *on* or 1) or inactive (i.e. *off* or 0). The interactions among genes can be represented by Boolean functions which calculate the state of a gene from the activation of other genes regulating it. The result is a Boolean network. Due to its simplicity, a Boolean network can analyze large-scale networks in an efficient way, but its simplicity makes a Boolean network *waste* a lot of useful information, for example, the detailed quantity information and time delay information for time series.

The general approach of a Boolean network is to discretize gene expression values into Boolean values, then find a set of Boolean functions which describe the state changes of each gene. Liang *et al.* [40] proposed REVEAL (REVerse Engineering ALgorithm) to resolve the problem. REVEAL uses information theoretic principles to reduce the search space and establish how the given genes are connected in the networks, and then determines the functions that specify the interactions among genes. REVEAL needs to enumerate all possible state transitions to build a Boolean network. To decrease complexity, a maximum fan-in, k ($1 \leq k \leq N$ where N is the number of genes in the dataset), is applied to each gene. For each gene, all possible subsets with less than *k* genes are considered to be its candidate regulators. If a subset is found fully determining the state changes of the given gene, it is said to be the regulator of the gene. An implementation of the algorithm proved to be capable of reliably reproducing networks with $N = 50$ and $k = 3$ given 100 state transition pairs (out of 1015 possible pairs). Working on a similar idea, Akutsu *et al.* [6, 39] proposed a simpler algorithm which proves that only $O(log\ N)$ state transition pairs (from 2^N pairs) are necessary and sufficient to identify the original Boolean network of N genes with high probability. Furthermore, Akutsu *et al.* [6, 39] extended a Boolean network to a qualitative network to model a gene network. Corresponding algorithms have also been proposed to learn the qualitative model. The Boolean system oversimplifies the gene regulation system and assumes the transitions to take place simultaneously, which is not the usual case in reality. Several improvements of Boolean networks have been proposed, such as Fuzzy Logic Models [55-58] and Probabilistic Boolean Networks [59].

In summary, the pros and cons of Boolean model are:

- *Pros*: Simplicity, can analyze large-scale networks in an efficient way
- *Cons*: Static network structure, a lot of useful information like: the detailed quantity information and time delay information for time series is "wasted".

2.2.4 Linear model

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

A general linear genetic network model is represented in the following equations [10, 15]:

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

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

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

In summary, the pros and cons of linear model are:

- *Pros*: Deterministic fully-connected network, scalable
- *Cons*: Under-constrained, assumes linearity of interactions

2.2.5 Differential equation model

Using a differential equation to model a gene network is computationally more intensive and requires the assumption of specific kinetic schemes. However, using smaller time-steps and continuous variables, a differential equation may get a more accurate physical representation of a gene network [10, 15, 52, 60].

A popular model is the linear differential equation [52]. Chen *et al.* [52] proposed a linear differential equation to model gene expressions. Both transcription and translation are modeled in the dynamic system by kinetic equations with feedback loops from translation product proteins to transcription, and incorporating the degradation of proteins and mRNAs; the system is as follows:

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

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

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

n: Number of genes in the genome

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

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

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

L: Translational constants, *n x n* non-degenerate diagonal matrix

V: Degradation rates of mRNAs, *n x n* non-degenerate diagonal matrix

U: Degradation rates of Proteins, *n x n* non-degenerate diagonal matrix

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

Watanabe and Maki [61] proposed an S-System to infer a gene network from sets of time-course data, each of which has resulted when a specific is disrupted. They proposed that the expression level of a gene is computed by the power-law function:

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

where n is the total number of state variables or reactants. g_{ij} and h_{ij} are the interactive affectivity of X_j to X_i. The first term represents all influences that increase X_i whereas the second term represents all influences that decrease X_i and X_j are some positive coefficients. The parameters are inferred by genetic algorithm [62]. Wahde and Hertz [16] built a non-linear differential equation based on continuous- time recurrent neural networks:

$$T_i\dot{x}_i + x_i = g(b_i + \sum_j w_{ij}x_j) \tag{2.6}$$

where for $i = 1,...,n$, i is a rate constant, x_i is the expression level and x_i is its derivative with respect to time. For a set of n genes, there are n x n weights (w_{ij}), n

bias terms (b_i) and n time constants x_i. The equation can be extended to include higher order terms in a network. The non-linear activation function g is defined as:

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

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

In summary, the pros and cons of linear model are:

- *Pros*: Model detailed quantities changing over time, dynamic
- *Cons*: As there are many parameters, they need more measurements; used for small scale network; assumption the concentrations of the substances change continuously and deterministically. Numerical simulation shows that in many cases that there are no qualitative differences between differential equation solutions and those based on the linear approximation.

2.2.6 Bayesian network model

In recent years, several models based on the Bayesian network (belief networks) have been proposed for learning gene networks [17, 19, 33, 41, 42, 44, 45, 59, 63-65]. Because of its suitability for learning gene networks, the Bayesian network is one of the most widely used models in the research area nowadays. In this section, we will define what a Bayesian network. In the next section, we see how a Bayesian network is learnt, and finally, in Section 2.4 we see how gene networks are learnt using a Bayesian network.

Bayesian Network

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

$$p(X_1,\ldots,X_n)=\prod_{i=1}^{n}P(X_i\mid X_1,\ldots,X_{i-1}) \tag{2.8}$$

or

$$p(X_1,\ldots,X_n)=\prod_{i=1}^{n}P(X_i\mid X_{i+1},\ldots,X_n) \tag{2.9}$$

Where $X_1, \ldots, X_n$ are the random variables.

- Example: As shown in Fig. 2.1, the probability of the variables X, Y, Z and W can be represented as follows:

$$P(X,Y,Z,\mid W,Q)=P(X\mid Y,Z,W)P(Y\mid Z,W)P(Z\mid W)P(W) \tag{2.10}$$

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

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

In the network G, any joint distribution can be decomposed in the product form:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid Pa(X_i)) \tag{2.11}$$

where $Pa(X_i)$ are the parents of X_i in G.

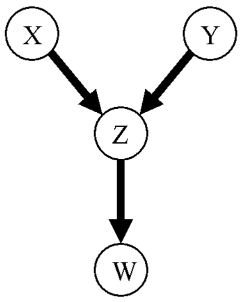


Fig. 2.1 Sample Bayesian network for illustration

Example: As shown in Fig. 2.1, the probability of the variables X, Y, Z and W can be represented as:

$$P(X, Y, Z, W, Q) = P(X)P(Y)P(Z \mid X.Y)P(W \mid Z) \tag{2.12}$$

Given D and the corresponding structure G, the parameter set θ can be estimated [66] by encoding θ in a prior distribution $P(\theta)$. The distribution is then updated using D, thereby obtaining the posterior distribution $P(\theta|D)$ by applying the following Bayes' rule:

$$P(\theta \mid D) = \frac{P(D \mid \theta)P(\theta)}{P(D)} \tag{2.13}$$

Based on Equation 2.10, θ_i can be estimated independently.

Before, we go into further details about learning Bayesian network; we look at a comparative analysis of the various models discussed above. Fig. 2.2 briefly shows various types of models with varying degree of abstraction of modeling and also the amount of information needed. The Fig.2.2 is a result of conceptual comparison and is not empirical. Ordinary differential equations (ODEs) models are used for small scale network that require least amount of information. Although gene regulations modeled via ODEs are successful to represent some reactions like linear production and degradation, they cannot describe the small system variability of the actual reactions. The system variability can be modeled using Markov chain models where there are sets of states and sets of state transitions along with their associated probabilities.

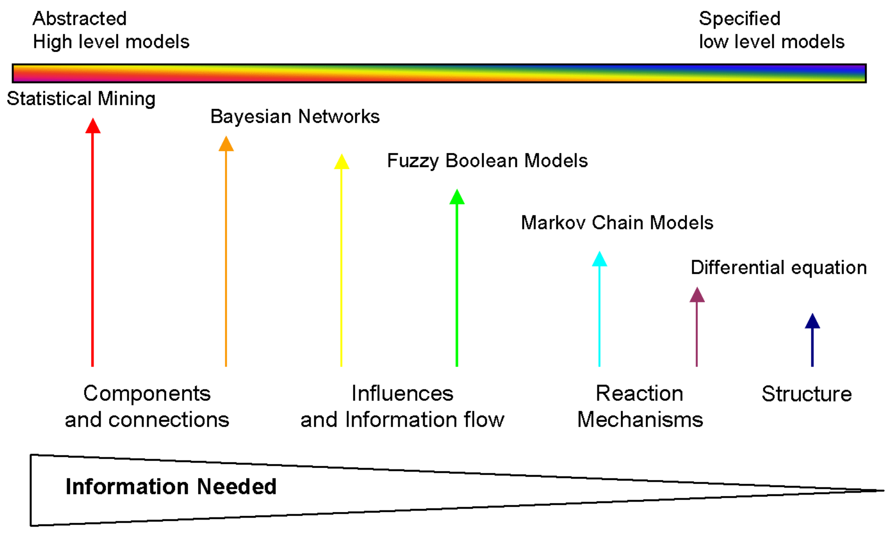


Fig. 2.2 Abstraction of GRN modeling

Next level of abstraction for inferring genetic regulatory interaction is the well accepted Boolean network model. Each gene is modeled as being either "ON" or "OFF" and the state of each gene at the next time step is determined by a Boolean function of its inputs at the current time step. In a real cell, however, gene expression is a continuous variable. So, fuzzy models have been proposed as alternatives where membership functions for the expression levels are classified as, for example, high, medium and low. The fuzzy and Boolean models are capable of modeling influences (activation/repression) and their corresponding direction of information flow, but require significant amount of data in order to deliver valid models. Bayesian networks give a more accurate model of network behaviour, based on Bayesian probabilities for the variables [66]. In the graphical representation of a Bayesian network, variables (genes) are represented as nodes and edges between nodes represent conditional dependence and causal relations. Bayesian networks include most of the previously proposed models as special cases and are inherently capable of incorporating existing knowledge. The Bayesian model is by far the most complicated and probably most efficient way of gene network modeling in the presence of noise. However, the accuracy of the results relies on quantity of data. Well established statistical data mining techniques can deliver the highest level of abstraction in modeling, however they are not tolerant to noise in the data. The domain expert's acceptance of Bayesian network models is a good choice as it provides a graphical representation of a GRN and is facilitated by the stochastic and white-box nature of Bayesian networks (BNs). The advantages of using Bayesian network for GRN modeling is presented in Section 2.3.3. In the next section we will look at the two types of learning mechanisms in Bayesian networks.

2.3 Learning Bayesian Network

Given D, the problem of learning a Bayesian network structure can be stated as follows: Given a training set $D = D_1, \ldots, D_n$ of independent instances of X, find a network $B = \{G; \theta\}$ that best explains D. There are two main approaches for finding structures. The first approach learns a Bayesian network as a constraint satisfaction problem [46, 54, 67-77]. In this approach, properties of conditional independence among variables are estimated by a statistical hypothesis test, such as T-test or chi-square test [67]. A network is then built to exhibit the observed dependencies and independencies. The second approach, which is more popular, learns a Bayesian network as an optimization problem [14, 30, 31, 33, 41, 42, 64, 66, 78].

2.3.1 Constraint based Learning

Using constraints is another way of learning BN structure. The constraints are typically conditional independence statements. The conditional independence tests that are used in practice are statistical tests on the data set. In order to use the results to reconstruct the structure, several assumptions have to be made [67, 77]: Causal Sufficiency, Causal Markov, and Faithfulness defined below.

- *The Causal Sufficiency Assumption:* It states that there are no unobserved variables in the domain that might explain the independencies that are observed in the data, or lack thereof. It is a crucial assumption for applications that need to determine the true underlying (causal) structure of the domain.
- *The Causal Markov Assumption:* It expresses a minimum set of independence relations that exist between every node and its non-descendants, given a BN model. From these, and a set of axioms described in [67, 77], one can produce the entire set of independence relations that are implied by that BN model.
- *Faithfulness Assumption*: A BN graph G and a probability distribution P are faithful to one another if and only if every one and all independence relations valid in P are those entailed by the Markov assumption on G.

With these assumptions in place, one can ascertain the existence of an edge between two variables and the direction of that link. Before going further, we briefly look at the concept of *d-separation* which is vital for understanding constraint based learning.

A path is a sequence of consecutive edges (of any directionality) in the graph. Two nodes X and Y in a directed acyclic graph are *d-separated* if every path between them is blocked. As an example, considering 3 disjoint sets of variables X, Y, and Z, represented as nodes on a DAG, we say a path is said to be d-separated, or blocked, by a set of variables Z iff the path (a) contains a chain (b) or a fork (c) contains an inverted fork, or collider, such that the middle variable m is not in Z and such that no descendant of m is in Z. (see Fig. 2.3). Further, a set Z is said to d-separate X from Y iff Z blocks every path from a variable in X to a variable in Y.

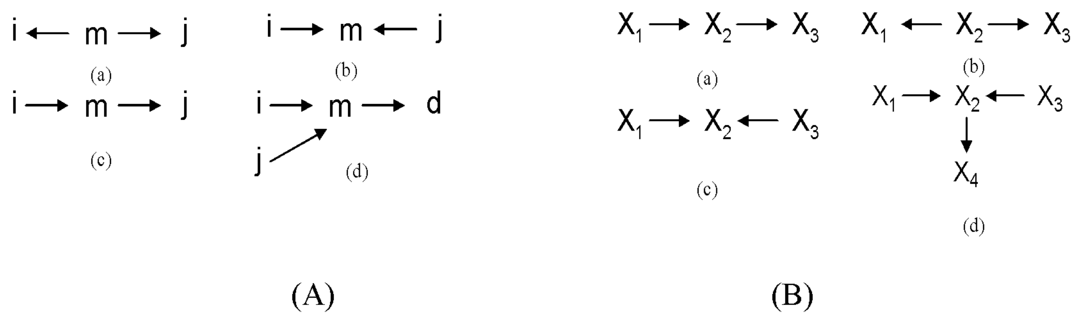


Fig. 2.3 D-separation Illustrative example figure

The Fig. 2.3 (A) shows a no descendant example. If *d* is in *Z*, then the path from *i* to *j* is unblocked even if *m* is not in *Z*.

Based on these definitions of d-separation, the useful theorem can be stated as follows.

- *Theorem:* If *X* and *Y* are d-separated by *Z* in a DAG, then $X \perp\!\!\!\perp Y \mid Z$. Conversely, if *X* and *Y* are not d-separated by *Z* in the DAG, then *X* and *Y* are dependent conditional on *Z*.

To help understanding this theorem, four basic graphical structures (Fig. 2.3 (B)) and the independences implied by each.

(a) *X2* is an intermediate variable. The only independence implied by this structure is $X1 \perp\!\!\!\perp X3 \mid X2$. It is NOT true that $X1 \perp\!\!\!\perp X3$.

(b) *X2* is a common cause. The only independence implied by this structure is $X1 \perp\!\!\!\perp X3 \mid X2$. It is NOT true that $X1 \perp\!\!\!\perp X3$.

(c) *X2* is a common effect. The only independence implied by this structure is $X1 \perp\!\!\!\perp X3$. It is NOT true that $X1 \perp\!\!\!\perp X3 \mid X2$.

(d) *X2* is a common effect. *X4* is an effect of a common effect. The independences implied by this structure are $X1 \perp\!\!\!\perp X3$, $X4 \perp\!\!\!\perp X1 \mid X2$, and $X4 \perp\!\!\!\perp X3 \mid X2$. It is NOT true that $X1 \perp\!\!\!\perp X3 \mid X2$ or that $X1 \perp\!\!\!\perp X3 \mid X4$. This is the trickiest structure you will find.

Paths in a Markov Blanket

To search a set of d-separating paths inside a sub-model such as a Markov Blanket (see Chapter 3 Section 3.3.3) is by separating them as upward, downward and sideway paths (Fig. 2.4). The upward path (red arrows) is the blocking path to a node from the parents, the downward path (green arrows) is the open path from the node through its children and sideway path (blue arrows) is the path between the node and its spouse node.

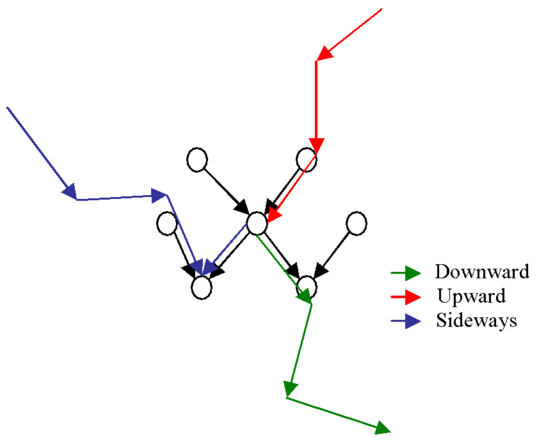


Fig. 2.4 Paths in Markov Blanket

SGS Algorithm (Sprites – Glymour – Scheines)

In that, the existence of an edge between two variables, say X and Y, is tested using a number of conditional tests. Each of these conditions is a subset of universal subset U. If Faithfulness holds and there exists an edge, then all these independence tests should be false. If there is no edge, then there must exist a subset d-separating them. Assuming that there is no direct edge between X and Y in the true model, one such subset is the set of parents of one of the nodes. By trying all possible subsets of U, the SGS algorithm can make a conclusion on the existence of an edge between every pair of variables in the domain. After the undirected connectivity is determined, SGS attempts to determine the directionality of these edges. This is done by examining triples of variables X, Y, and Z, such that there are no subset that includes Z can d-separate X and Y, then the directionality of respectively. This is repeated for all such triples, and is followed verifying acyclic behavior of the graph.

As we mentioned above, the algorithm and any other independence-based algorithm for that matter cannot necessarily assign directions to every edge. Doing so depends on the true underlying structure of the BN model. For example for a BN of three variables, Z, the direction of either edge cannot be determined by any set of independence statements, because two other networks with the same undirected structure, namely belong to the same equivalence class with respect to conditional independence statements implied by their respective structures. (See Chapter 3 Section 3.5 for more details)

Another algorithm, similar in flavor to the SGS is the IC, or "inductive causation" algorithm by Pearl and Verma [67, 79]. Other algorithms exist in the literature that do not make use of independence tests but take into account d-separation in order to discover structure from data. Cheng *et al.* [63] for example uses mutual information instead of conditional independence tests. The algorithm requires the ordering of the variables to be given to the algorithm in advance. Constraint-based algorithms have certain disadvantages. The most important one, manifesting frequently in practice, is their poor robustness. By the term "robustness" here we mean large effects on the output of the algorithm i.e. the structure of the BN, for small changes of the input i.e. single errors in the independence tests. The problem seems to have its roots on the dependency of later parts of the algorithms to earlier ones, something that seems difficult to avoid. For example, step 2 of the IC algorithm [67, 79] determines the direction of pairs of edges that were found in step 1. Therefore a missing edge might prevent another edge's direction to be recovered. In addition, step 3 propagates the directions of edges determined in step 2, so an error in step 2 might be propagated in step 3, possibly resulting in a structure with directed cycles which is illegal under the present BN formulation. Another disadvantage is that the SGS (and the IC) algorithm, as we can see from its algorithmic description, is exponential in time in the number of variables of the domain, because it is conducting independent tests conditional on all 2 subsets of set S with S containing n^2 variables. This makes it impractical for large domains of tens or even hundreds of variables. A more efficient algorithm is the PC algorithm. Its efficiency comes from ordering the conditional independence tests, from small to large. The algorithm is presented in detail in Spirtes *et al.* [77].

2.3.2 Score based Learning

A statistically motivated scoring function, termed scoring metrics, such as Minimum Message Length (MML) [80] or Bayesian score [66], is introduced to evaluate a network with respect to D, and the optimal network according to this score is computed. Bayesian Score (*ScoreB*) [47, 66] is a popular score metrics and it is defined as follows.

$$\begin{aligned} ScoreB(G:D) &= \log P(G \mid D) \\ &= \log \frac{P(D \mid G)P(G)}{P(D)} \\ &= \log P(D \mid G) + \log P(G) + C \end{aligned} \qquad (2.14)$$

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

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

$$Score_B(G:D) = \sum_i Score_B(X_i \mid Pa(X_i):D) \qquad (2.15)$$

It is not feasible to compute maximum likelihood as it involves computing marginal likelihood *P(D) = P(G) P(D; G)* which is the sum over an exponential number of models. Bayesian Information Criterion (BIC) is proposed to approximate the posterior.

$$\log P(G \mid D) \approx \log P(D \mid G, \hat{\theta}_G) - \frac{\log N}{2} \Delta G \qquad (2.16)$$

$$\log P(G \mid D) = \sum_i \log P(X_i \mid Pa(X_i) : D) \qquad (2.17)$$

where *N* is the number of samples, *G* is the dimension of the models (the number of free parameters if *D* is fully observed, i.e. without hidden variables) and *θ(G)* is the Maximum Likelihood (ML) estimate of the parameters. The decomposability of the score is crucial to learning a Bayesian network. With it, the learning problem, which is known to be NP-hard [81], can be solved by a set of local searches.

2.3.3 Advantages of Bayesian network for GRN modeling

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

- Bayesian networks are particularly useful for describing processes composed of locally interacting components [67]. That is, the value of each component directly depends on the values of a relatively small number of components.
- Statistical foundations for learning Bayesian networks from observations, and computational algorithms to do so are well understood and have been used successfully in many applications [66, 82, 83].
- Bayesian networks provide models of causal influence [5, 67, 68, 74, 76, 77, 84]. Although Bayesian networks are mathematically defined strictly in terms

of probabilities and conditional independence statements, a connection can be made between this characterization and the notion of direct causal influence.

- Because of its firm statistic basis, the Bayesian network can deal with the stochastic aspects of gene expression and the noisy measurements of microarray data in a natural way [85, 86].
- Bayesian networks are able to handle a large number of variables with only a few replicates [33, 47, 68, 71, 87]. It is especially useful when learning gene networks, since microarray data generally have thousands or even tens of thousands genes but only tens of replicates. Besides, Bayesian networks are capable of estimating the confidence of different features in networks [36]. The absence of data often leads to the consequence that many networks explain the data equally well. The confidence is useful for measuring to measure whether a statistic feature of the network is likely to be true.
- Learning gene networks is NP-hard [81]. The decomposability [77] of Bayesian networks ensures local searches achieve global optimization, thus making the learning easier.
- Hidden variables in a network and missing values in gene expression data are easy to handle with the Bayesian network. Many methods have been established for in learning Bayesian network with latent variables and missing values.

Bayesian networks can capture many types of relationships among genes: linear, non-linear, combinatorial, stochastic and so on. It remains unclear which types of relationships a gene regulatory system may pursue. The ability of Bayesian networks to grasp various types of relationships makes it appropriate for learning gene networks. The problems with using Bayesian networks is discussed in a very early paper by Henrion [88]

2.4 Learning Gene Network Using Bayesian Network

Friedman [17] proposed modeling a gene network as a Bayesian network. The variables of the BN are the genes from the gene expression data chip. The space of the values that each variable gets is specified using two approaches. The first one is the discrete approach, in which each variable gets three values: under-expressed, normal and over expressed (compared to a control that could define the normal value as the average expression value of each gene over all experiments or the average expression value of all gene expression measurements in each experiment). The second one is the linear Gaussian approach, in which a linear regression model is learnt for each variable given its parents. Since the data contains only a few dozen samples and thousands of variables, it is not possible to learn the exact structure of the network from the data (in order to learn the exact structure of a BN with thousands of variables one needs many more samples). Thus, only small features of the network could be learnt. Each feature is a pair of genes related in one of two possible ways. The first relation, called Markov relation, contains pairs of genes having an edge between them in the BN or genes which are parents of a third gene in the BN. The second relation, called order relation, contains pairs of genes with a path between them in the BN. In order to understand to what extent the data supports a certain feature a bootstrap method was applied, in which "pertubed" versions of their data (smaller data sets that contain part of the samples of the full data, chosen randomly) was generated and the BN structure was learnt. The confidence of the feature was measured as the percentage of the number of times it appeared in the BN structures learned from the "pertubed" data versions. In order to learn the BN structure from data, an algorithm was developed that reduced the graphs search space, by identifying a relatively small number of candidate parents for each gene, and restricting the search space to networks with these parents. The search algorithm developed is an iterative with each stage adding more candidate parents to each gene, if they contribute to the gene score. Friedman showed that the results obtained by the sparse candidate learning algorithm are biologically meaningful results by examining the results with a set of statistic measurements: robust test, order relation, Markov relation, and so on. Since then, many works based on the Bayesian network frame work have been proposed, and biologically relevant results have been obtained.

Hartemink [89] extended Friedman's work by adding these annotations to edges: "+","-" or "+/-" which represent positive, negative or unknown regulation. Beal *et al.* [31] proposed including the unmeasured genes as the hidden factors to learn a gene network. They proposed implementing the step by state-space models (SSMs). Lee *et al.* [90] proposed a modularized learning approach based on the assumption that most genes are likely to be related to other genes in the same biological modules rather than the genes in different modules. They proposed finding overlapping modules in the genes, and learning the sub-networks in modules with a Bayesian network. Murphy & Mian [42] used the Dynamic Bayesian Network (DBN), which is an extension of the Bayesian network, to model gene networks. In this model, a gene at a time point is regulated by its parent in the previous time point. Thus, the acyclic limitation of the Bayesian network is overcome in DBN. Murphy *et al.* gave a thorough report in [42, 43] on the application of DBN in learning gene networks. Imoto *et al.* [41] further extended Bayesian networks and DBN by integrating nonparametric regression into the models, so that the methods can use continuous gene expression values instead of the discrete values in the general Bayesian network approaches. Their method is capable of capturing the non-linear relationships among genes. Yu *et al.* [91] presented an influence score to measure the magnitudes of regulatory strength of the edges. It is useful for eliminating the false positives as well as distinguishing the positive or negative regulation of edges.

With more and more works using Bayesian networks as the framework to tackle the gene network reconstruction problem [17, 19, 33, 41, 42, 44, 45, 59, 63-65], it can be seen that the Bayesian network is becoming a widely used approach in learning gene networks.

2.5 Summary

In this chapter, we briefly discussed the models used in gene regulatory network reconstruction. While comparing the different models and their advantage and disadvantages, we can observe that Bayesian network model is a suitable candidate for effective reconstruction of realistic gene network models. Various methods that involve Bayesian network have also been presented. Although the existing methods have their merits they are still faced with problems related to accuracy of reconstruction, handling noise, searching through an enormous space of structures, complexities associated with the gene network structures, parameter learning and so on. To cope with these limitations, there is a need for a novel sophisticated intelligent method which can bring out the essence of a realistic gene regulatory network that could provide insight into the complexities in the nature. The suitability of a constraint based approach for learning Bayesian network is highlighted due to its flexibility and reliability in providing robust results. In the next chapter we present the novel causal modeling method for reconstruction of gene regulatory network models.

Chapter 3

3. Causal Learning and inference of GRN Structure

3.1 Introduction

As stated in previous chapters, a gene regulatory network (GRN) represents a set of all interactions among genes determining the temporal and spatial patterns of expression. Early attempts to reverse engineer gene networks from microarray were somewhat limited. For example, although real gene expression data showed that gene expression levels tend to be continuous rather than discrete, a discretized data was often used to simplify evaluation [17, 40, 61, 72, 73, 92, 93]. Further, many researchers assumed either a no-time delay or a constant-time delay in gene expression [46, 70, 94, 95] in spite of different gene pairs having varying time delays for gene regulation [3]. Chen *et al.* [52] amongst others attempted to incorporate various time delay factors into the gene network learning process by using differential equations. However, the algorithm was of high computational complexity and without any suitable experiments being reported. Although the effect of regulation is likely to be nonlinear [10], linear causal models can provide the necessary simplification for any biological system which they represent. With the number of samples available from most microarray experiments being limited (often as few as six or seven samples), causal models appear promising because they use a small number of parameters to represent activation and repression relationships between genes. Complex models can better represent a wider range of relations such as thresholds or combinatorial interactions, but there is an increased risk of over fitting with small sample sizes. Recently, a variety of linear models have been presented for modeling gene regulation [10, 14, 16, 37, 47, 60, 96] which essentially represent the expression (or a change in expression) of a gene as a linear function of the expression levels of other genes. Some approaches focus on finding predictive (but not necessarily causal) relations between genes. For example, D'Haeseleer *et al.* [47] used a multiple regression method that identifies correlation between gene expression levels but could not determine whether genes are linked directly or indirectly connected through other genes. The major reasons for these above mentioned difficulties are: (i) microarray datasets contain thousands of genes (fields, attributes) and only a few time steps (records) (ii) presence of noise in expression values (iii) existence of splice variants, etc. Moreover, significant computational time is required to analyse large volumes of data, and the complexity of a regulatory network model increases with the number of genes used for the model which results in longer execution time and poor scalability.

In this chapter we present a novel method for learning the structure of a causal Bayesian network for GRN because in a BN (as mentioned in Chapter 2), learning structure and learning probabilities are treated as two separate problems where the former is considered to be much more challenging and known to be NP-hard [81]. The causal model aims at representing the underlying physical mechanisms that generated the data. The model represents the independency/ dependency relations among genes and nature of regulation (down or up). The correspondence of the conditional independencies (CI) (see Section 2.3.1) in the graph and the data is called *faithfulness*. Causal structure learning algorithms try to construct a faithful graph based on the conditional independencies found in the experimental data. Knowledge about the independencies among the genes that exist in a GRN domain is an extremely useful piece of information that researchers need to elicit during their investigation. The reason for this is the fact that conditional independence statements concerning observed quantities in GRNs (i.e. genes) can reduce the number of parameters under consideration and greatly aid in the understanding of the interactions or significant insights to the interactions occurring amongst the genes.

In this chapter, we present Markov blanket based GRN learning algorithm and Path analysis algorithm for inference of the structure of a BN from conditional independence tests. We first describe an approach based on Markov blanket (MB). It operates by identifying the local neighborhood of each variable in the Bayesian network as a preprocessing step, in order to facilitate the recovery of the exact structure around each variable in subsequent steps. A flexible qualitative measure is formulated to determine how well a putative MBG (Markov Blanket Graph) fits the gene expression data. In our work, the first and second order partial correlations are used as tools in comparing the putative network model against the data. Although these correlations have a high false discovery rate, they are known to have excellent capabilities for entailing meaningful relationships [46]. Further, a suitably designed genetic algorithm (GA) (presented in Section 3.3) is applied to efficiently search the solution space of BN structures. In the first step of the GA, the search aims to increase the correctness of the network structure by maximizing the overall network score and in doing so the search can ignore the occurrence of false edges in the optimal solution. Hence, in the second step, using a set of tests, these reconstructed networks are pruned of the false positives resulting in minimal connectivity and best fit GRN for the data. The Markov Blanket method has the added advantage of being easier to verify and possibly minimize the number of CI tests needed. When there are large number of variables densely connected (e.g. in yeast where there are 6000 genes), a Constraint Minimization version of the algorithm is proposed which employs a K-map minimization technique to minimize the computation and still ascertains the same result with high accuracy in the presence of noise.

The rest of the chapter is organized as follows. Section 3.2 presents an outline of causal modeling of GRN. In Section 3.3, the various V, Y and MB causal methods of learning GRN are presented and a comparison is shown between the 3 methods. In Section 2.4, a Markov Blanket based method for learning GRN structure is presented. Section 3.5 presents a optimized version of the Markov Blanket method by incorporating constraint minimization. Section 3.6 presents a path analysis pruning algorithm to eliminate spurious interactions. Section 3.7 presents experiments and results using yeast cell cycle datasets. Finally Section 3.8 gives the summary of the chapter.

3.2 Causal GRN modeling preliminaries

A linear causal model represents relationship between a node v_i and its parents $Pa(v_i)$ using the following Eqn. 3.1.

$$v_i = \sum_{k=1}^{Ki} \alpha_k Pa_k(v_i) + R_i \tag{3.1}$$

Here, k_i is the number of parents to the node v_i, α_k is the path coefficients representing the causal effect of variable in $Pa(v_i)$ on v_i and finally R_i is the error term. Fig. 3.1 shows a graphical representation of the causal model, where nodes represent the expression profiles, the arrows represent path coefficients and the direction of causal influence and sign on the arrow represents activation or repression. No sign represents positive influence (activation) by default.

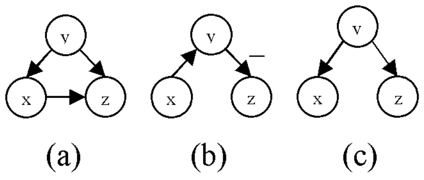


Fig. 3.1 Three node causal models of gene regulation

The causal model can specify the exact influence between the genes. For example, the model in Fig. 3.1 (a) indicates that the expression of gene Z is directly influenced by the expression of gene X and gene Y. However, the model in Fig. 3.1 (b) indicates that the expression of gene Z is directly influenced by expression of gene Y and indirectly influenced by expression of gene X through gene Y. In other words, gene Y is an intervening gene. In Fig. 3.1 (c), the expression of gene X and gene Z are independent of each other and are directly influenced by gene Y. Here gene Y is anteceding gene. At a more detailed level, the sign on a causal arrow (e.g. the sign on $Y \rightarrow Z$ in Fig. 3.1 (b)) specifies type of causal interaction. A positive or no sign on the link $Y \rightarrow Z$ signifies that expression of gene Z should increase with increase in expression on gene Y (activation), while a negative sign on the link signifies that expression of gene Z should decrease with increase in expression of gene Y (repression). All the models given in Fig. 3.1 are acyclic graphs as they do not include any feedback loop. To reduce complexity, in this thesis work, we will restrict our attention to models without a feedback. Further, the network learned is a Bayesian network (as defined by Eqn. 3.1), and acyclic behavior is essential for ensuring consistency.

Linear causal models make predictions that can be scored against data. In the approach presented, three scores are assigned to a model for a given expression profile data, namely structure, direction of causality and the sign on causal arrow.

i) Scoring predicted structure

As the putative gene network is usually an extremely large model, we propose to decompose it into a series of three-variable sub-models similar to those shown in Fig. 3.1. Each sub-model is assigned a score and the sum of the scores indicates how well the sub-model structure fits the data. Partial correlation is used to evaluate the sub-models.

A partial correlation corresponds to the correlation between two variables when one variable controls the effect of the other variable. Partial correlations are significant because they help in determining whether correlated variables are linked directly or otherwise and to detect whether the correlation is spurious [5]. For example, in Fig. 3.1 (b), the partial correlation coefficient ($r_{xz.y}$) between gene X and gene Z controlling the effect of gene y is calculated using the following Eqn. 3.2.

$$r_{xy.z} = \frac{(r_{xy} - r_{xz}r_{zy})}{\sqrt{(1-r^2{}_{xz})(1-r^2{}_{zy})}} \tag{3.2}$$

where r_{xz}, r_{xy} and r_{yz} are Pearson correlation coefficients over the expression profiles of pairs of genes. A zero or a small partial correlation coefficient indicates that the variables are connected by a path that does not have a third variable involved. In the case under consideration, the model entails that gene X and gene Z are intervened through Y and if data implies $r_{xz.y} = 0$, it can be concluded that the structure fits the data. However, in case of the model shown in Fig. 3.1 (a), $r_{xz.y} >> 0$ is because there is a direct link between gene X and gene Z.

Table 3.1 explains various outcomes while evaluating a three gene sub-model using partial correlation. The causal inferences evaluated are as follows.

i) *No effect* occurs when the original and partial correlations are equivalent in magnitude and sign.

ii) *Explanation* occurs when the control variable is an anteceding cause of the independent and dependent, or when it is an intervening variable on the path from the independent to the dependent, and there is no direct causal path from the independent to the dependent. In this case, the partial correlation approaches 0 and for random samples should test as not significant. This is also called a control effect.

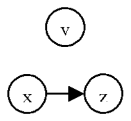
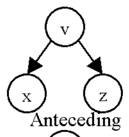
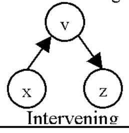
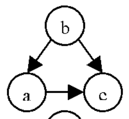
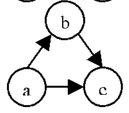
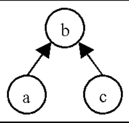
iii) *Partial explanation* occurs when there is a direct path from the independent to the dependent variable, but the control variable is also either an anteceding or intervening cause. In this case, partial correlation drops only part way to 0 compared to the original bi-variate correlation.

iv) *Spurious correlation* is a computational correlation without actual causal connection, such as that which occurs when the effect tested by partial correlation is partial or full explanation.

Given a model and data, the score function will be of the following form.

$$Score_1 = \sum_{i=1}^{n} (te_i + tpe_i - (fe_i * \alpha_i) - (fpe_i * \beta_i) - nf_i - sc_i) \tag{3.3}$$

Table 3.1 Comparing model and data for causal inference

Model	Causal Inference	Outcome
	**1. No Effect** If model entails *Effect* and data implies $r_{xz} = r_{xz.y}$	*Noeffect*
	**2. Explanation** If model entails *Explanation* and data implies $r_{xz.y} \neq 0$	False *Explanation*
	If model entails *Explanation* and data implies $r_{xz.y} \approx 0$	True *Explanation*
	**3. Partial explanation** If model entails *Partial Explanation* and data implies $r_{xz.y} \approx 0$	False *Partial Explanation*
	Else If data implies $r_{xz.y} >> 0$	True *Partial Explanation*
	**4. Spurious correlation** If model entails partial or full explanation and data implies $R_{xz} = 0$ & $\lvert r_{xz.y} \rvert > 0$	*Spurious correlation*

where n is the number of sub models and te_i, tpe_i, fe_i, fpe_i, are respectively the number of true/false, partial and full explanation. The selection of partial or full explanations is based on the sub-model structure shown in Table 3.1. The terms nf_i and sc_i respectively denote no effect and spurious correlation causal influences inferred for the sub-model *i*. These variables are assigned a value of either 1 or 0 based on the condition they satisfy. The value of α_i and β_i are derived from the partial correlation of the sub-model and corresponds to the extent to which the model is predicted falsely. This enables assigning higher scores to networks having higher probability to predict the correct structure for the given data compared to the networks that have a poorer structure.

ii) Scoring direction of causality

Partial correlation constraints let one recover much of the structure as most graphs will imply different constraints. However, the model $X \rightarrow Y \rightarrow Z$ has equivalent partial correlation constraints as $X \leftarrow Y \leftarrow Z$, which is a special case of the concept of equivalent graphs [66]. This can be resolved by estimating the time delay between expression profiles *X, Y* and *Z*. The time correlation between gene x and gene y can be expressed as shown in Eqn. 3.4

$$r_{XY}(\tau) = \sum_{n} g_X(n) g_Y(n-\tau) \tag{3.4}$$

Here, g_X and g_Y are expression profiles of genes *X* and *Y*. The term n is the time point and τ is time delay. For a periodic time profile, the expression values of time points at the end of the time series are rewound to the beginning of series after time shifting. The time delay τ' between the two expression profiles, is the value of τ for which maximum value of $|r_{xy}(\tau)|$ occurs. It enables to determine the direction of causality.

If $\tau' < 0$ then $X \rightarrow Y$ (i.e. gene x regulates gene y)

If $\tau' > 0$ then $X \leftarrow Y$ (i.e. gene y regulates gene x)

If $\tau' = 0$ then $X \leftrightarrow Y$ (i.e. x and y are co-regulated)

With the sub model and direction assignments known, the score function for direction is formulated as follows.

$$Score_{2_i} = \sum_{XY, X \neq Y} f(dir_m(X,Y), dir_d(X,Y)) \tag{3.5}$$

Here, $dir_m(X, Y)$ returns the direction between genes X and Y for the model and $dir_d(X, Y)$ returns direction between the same genes from data. The function f returns a value 1 if both directions are identical and 0 if they are different. The score is calculated as a summation over all the edges in the sub model.

iii) Scoring sign of regulation

The sign on a causal edge shows its regulation type. A positive sign indicates an up regulation and a negative sign indicates a down regulation.

Considering a three gene sub-model, suppression occurs when partial correlation $r_{XY.Z}$ is higher than the original bivariate correlation, r_{XY} (see Eqn. 3.2). In this case, either one connection or all three possible causal connections will have a negative sign.

$$| r_{XY.Z} | > | r_{XY} | \tag{3.6}$$

It the above condition is satisfied, then to specifically identify the sign of regulation from the sign of the time correlation between expression profile of two genes X and Y at time delay τ’. That is,

If $r_{xy}(\tau') < 0$ then there is negative regulation

If $r_{xy}(\tau') > 0$ then there is positive regulation

With model and sign assignments known, the scoring function will be of the following form.

$$Score_{3_i} = \sum_{XY, X \neq Y} f(\mathrm{sgn}_m(X,Y), \mathrm{sgn}_d(X,Y)) \tag{3.7}$$

Here, $\mathrm{sgn}_m(X,Y)$ returns sign of the causal edge between genes X and Y as specified by the model and $\mathrm{sgn}_d(X,Y)$ returns sign between the genes as determined from data. The function f returns a value 1 if both these signs are identical and 0 if they are different. The score is calculated as a summation over all the edges in the network model.

The score of the overall putative network is obtained as a weighted linear combination of the MBG consistency scores.

$$Fitness\ Score = \sum_i (w_1 * Score_{1_i} + w_2 * Score_{2_i} + w_3 * Score_{3_i}) \tag{3.8}$$

Here w_1, w_2 and w_3 are weights assigned to each of the three sub model scores such that $w_1+w_2+w_3=1$. The weights are adjusted to scale the individual sub scores so that they do not dominate the overall fitness score.

3.3 Genetic Algorithm

A simple genetic algorithm (GA), applied to explore this structure space, begins with a sample population of randomly selected network structures and their fitness calculated. Iteratively, random crossovers and mutations of networks within a population are performed and the best fitting individuals of the population are kept for future generations. As generations pass, the population evolves leaving the fitter structures while those performing poorly become extinct.

The hypothetical network structures are constructed with each gene having a set of M parents, where the value of M ranges between 2 and 7 [16]. The *nxn* chromosome matrix, encodes the network structure with each row corresponding to a tail of an edge and each column corresponding to the head. The chromosome encodes the presence of a directed edge between two genes, its direction and sign of regulation using values {1, 0,-1}, where 1 indicates positive regulation, -1 indicates negative regulation and 0 indicated no regulation. If for example, there is an edge between gene *X* to gene *Y*, with a negative sign of regulation, the chromosome encodes Chromosome $(X, Y) = -1$.

The crossover operations between two networks can be explained with the aid of Fig. 3.2. Taking two random individuals from the population, gene edges (3, 4) and (3, 2) are selected at random as shown in Fig. 3.2 and are swapped between the pair of networks.

Mutation is applied on an individual edge of a network. For our study, we incorporate the following four types of mutations.

i) Deleting a randomly selected existing edge from the network
ii) Randomly creating a new edge to the network
iii) Change direction of a randomly selected edge and
iv) Change sign of regulation on a randomly selected edge.

As both the crossover and mutation operations directly impact the structure of the network, the following issues need to be satisfied before an edge is created or manipulated so as to maintain the stipulations of a Bayesian network structure.

i) The addition or manipulation of an edge leaves the number of parents, M to a gene (which ranges between 2 and 7) unaffected.
ii) The addition or manipulation of an edge does not create a directed cycle (feedback loop).

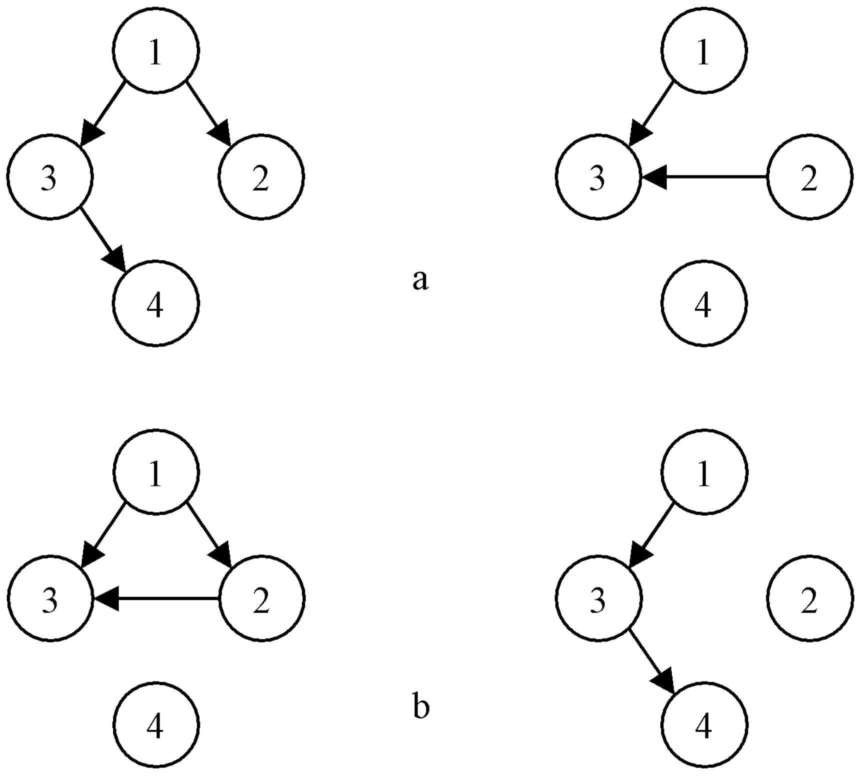


Fig. 3.2 Crossover: (a) Two four gene network structures before crossover. The nodes represent genes and edges represent regulatory connections. (b) Network structures after crossover where in edges between nodes 3-4 and 2-3 are swapped

The overall algorithm that includes the causal modeling of the GRN and the stochastic search of the network space using GA is as follows.

1. Create initial population of network structures. For each individual, genes and set of parent genes are selected (within the range from 2 to 7) based on a random Poisson distribution and edges are created such that the resulting network has no directed cycle.
2. From each chromosome, decompose the network structure into sub-models containing triplets of genes suitable for evaluation using the causal model explained before.

3. Evaluate each network using the fitness function given by Eqn. 3.8 and sort the chromosomes based on the fitness value.
    a. Generate new population by applying cross over and mutation on the previous population (as shown in Fig. 3.2).
    b. Evaluate each individual using the fitness function and use it to sort the individual networks.
    c. Take best individuals from the two populations based on fitness score and create the population of elite individuals for next generation.
4. Repeat steps a) - c) until either of the stopping criteria (given below) is reached. As the evolution goes on, the emerging new edges will increase the average score of the population of networks.
5. When the GA stops, due to stopping criteria, save the best chromosome and reconstruct a gene network.
6. Repeat steps 1)-5) recurrently a specified number of times (5 in our study) and generate specified number of different gene networks. Combine the results obtained to reconstruct the final gene network.

Depending on the problem domain, genetic algorithms are repeated from 5 to 10 runs, obtain robust results. Due to the stochastic nature of GA, we repeated our experiments for 5 times for this thesis work. The repetitions are also useful for discovering most significant connections in the network i.e. repeated occurrences of connections in each GA run. Such connections can then be combined to reconstruct the predicted genetic network.

The following two criteria are applied for stopping the genetic algorithm.

i) *Maximum Limit*: When the iteration reaches a predefined maximum number of generations or

ii) *No Improvement*: When the difference between the current fitness average and previous fitness total is less than a specified value (= 0.0001 in our case), the GA search is stopped.

In the next sub-section, we analyze three variants of sub-model fitness metrics that prove effective for structure discovery and which will be useful in the development of the MB algorithm presented in Section 3.4). These variants of sub-model are namely, the "V" sub-structure (triplets), "Y" sub-structure (quadruplets) inference and the Markov Blanket (MB) sub-structure (i.e. set of direct causes, direct effects, and direct causes of the direct effects) inference. Much of our work in learning the V, Y and MB BN sub-models has been devoted to the derivation of their individual scoring metrics. The objective is to induce a network (or a set of sub-networks) that "best describes" the training data. This optimization process was implemented by using heuristic search technique (GA) to find the best candidate over the space of possible networks. This search process relies greatly on the derived scoring function that assesses the merits of each candidate network. For the work being presented here, all aspects are handled conveniently while integrating scoring functions into the GA.

3.3.1 The V-structure

As shown in Fig. 3.3, a V-structure forming the sub-network of the putative Bayesian network contains three nodes *X*, *Y* and *Z* which are node labels to represent the genes in the network. While traversing the path from node *X* to node *Y*, three types of connection patterns may be encountered in a GRN as illustrated in Fig. 3.3 (a-c). These are called as serial, diverging, and converging connections. However, due to the nature of representations, they are also known as chain, fork, and collider connections. Generally, if the two parent nodes of a collider are not directly connected by an arrow, this structure is termed an unshielded collider, or a V-structure. However, a chain or a fork can also be considered as a V-structure when nodes are oriented in a V shape.

A V-structure is inferred from the data, if the following conditions hold:

$$Y \perp\!\!\!\perp Z$$
$$Y \not\perp\!\!\!\perp X$$
$$Z \not\perp\!\!\!\perp X$$
$$Y \not\perp\!\!\!\perp Z \mid X$$

where $\perp\!\!\!\perp$ indicates independence and $\not\perp\!\!\!\perp$ indicates dependence.

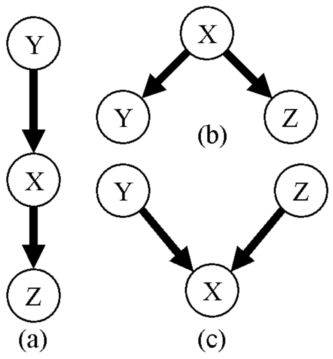


Fig. 3.3 V-Structures (a) Chain (b) Fork (c) Collider

In the case of a chain or a fork, X and Y are d-separated (independent) through X whereas for a collider, X and Y are d-connected (dependent) through X. To learn V-structures, it is necessary to test conditional independencies among the random variables in V structure with the help of the training data [77]. When such CI tests are carried out on all three node models shown in Fig. 3.3, we found that the algorithm had to perform unnecessary CI tests on non-essential variables leading to reduced causal performance. This is because of the fact that, the more the CI tests that have to be performed, the lower is the accuracy in the results. This will be confirmed experimentally later in Section 3.3.4, where we show that this loss of accuracy may lead to more spurious relations compared to the proposed two new variants under study

Next we look at the Y- structure.

3.3.2 The Y-Structure

A Y-structure sub-network of a Bayesian network contains four nodes and has the structure shown in Fig 3.4.

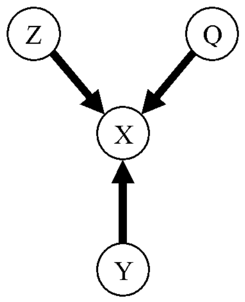


Fig. 3.4 Y- Structure

Y-structure is inferred from the data, if the following conditions hold true.

$$\begin{array}{ll}
Z \perp\!\!\!\perp W & Z \not\perp\!\!\!\perp Y \\
Z \not\perp\!\!\!\perp X & W \not\perp\!\!\!\perp Y \\
W \not\perp\!\!\!\perp X & Z \perp\!\!\!\perp Y \mid X \\
Z \not\perp\!\!\!\perp W \mid X & W \perp\!\!\!\perp Y \mid X \\
X \not\perp\!\!\!\perp Y &
\end{array}$$

Y-Structure includes three V-structures: $Z \rightarrow X \leftarrow W$, $Z \rightarrow X \leftarrow W$ and $Z \rightarrow X \leftarrow Y$. The node X is the middle node in all the three V-structures. We investigate Y-structures because, CI tests based upon the middle node X of the Y-structure, can alone help determine the causal faithfulness of the whole Y-structure using the training data, clearly avoiding unnecessary CI tests over three different V-structure. This results in performing tests using smaller condition sets and eventually three times computationally faster. However, in practice, there are several different variants of Y structures encountered. For example, the V-structure $Z \rightarrow X \leftarrow W$ in Fig 3.4 can appear as a serial, fork, shielded collider, etc. as shown in Fig 3.4 (a) and (b). Considering another example, $X \leftarrow Y$ in Fig 3.4 can appear as $X \rightarrow Y$ in the Y-structure. The effect of this will be to vary the size of condition sets needed for performing CI tests and thus result in varying computational and causal performance compared to V-structures. This is investigated further in Section 3.3.4.

3.3.3 Markov Blanket

The concept of the Markov blanket [67] of a variable or a set of variables is central to this thesis work. The Markov blanket comprises the parents, the children, and the parents of the children of the node of interest. In Fig. 3.5, a Markov blanket of a node *X*, denoted as *MB(X)* is a minimal set of variables, such that every other variable is independent of *X* given *MB(X)*, i.e.

$$\forall Y \in \{X_1, \ldots, X_n\} \setminus \{MB(X), X\}, X \perp\!\!\!\perp Y \mid \mathrm{MB(X)} \tag{3.9}$$

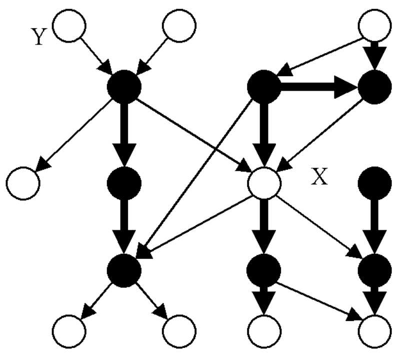


Fig. 3.5 Markov Blanket of X

A Markov Blanket Graph (MBG) of gene *X*, comprising of a gene and its MB neighbors is illustrated in Fig. 3.5. The genes (black) are considered as MB neighbors of gene *X* as there is an edge between them, or if they have a child in common. The gene *Y* (white) are independent of gene *X* given the MB neighbors of *X*. A MB DAG is formed by combining the MB of all nodes in the dataset *D*.

The notion of a Markov Blanket (MB) for a node *X* in a dataset *D* is significant in GRN context for two reasons. First, the nodes within each Markov blanket have a similar set of dependencies and therefore exhibit a similar behavior. Similarly, many genes in a cell are organized into small groups, in which sets of genes required for the same biological function or response are co-regulated by the same inputs in order to coordinate their joint activity. Second, they can also have a causal interpretation: a directed edge from one variable to another, $X \rightarrow Y$, represents the claim that *X* is a direct cause of *Y* with respect to other variables in a DAG, i.e., if other variables were to be held fixed at appropriate values, and *X* were varied by an intervention (e.g., activation/repression), *X* and *Y* would co-vary [67, 77]. A MB DAG can thus provide both biological and causal insight into relations between a reduced set of predictor nodes and the target node.

3.3.4 Comparison of V, Y and MB methods

Searching the space of causal models is often performed as an optimization process, that is, the algorithm looks for a structure optimizing some goodness of fit measure, the latter being a decomposable scoring function that involves several tests. In the previous work, a GA was developed for performing this optimization task. In this chapter we continue to use the same algorithm and only varying the fitness function. The fitness involves three score, score for structure, score for direction of regulation and score for sign of regulation. As the goal is to exploit the structure discovery module, we derive learning algorithms for various sub-models using statistical CI tests based on partial correlation. In the algorithms presented below, partial correlation upto 2nd order has been used.

Table 3.2 Algorithm to learn V-structure

```
*Algorithm* V
**Input**: database $D$, putative v-structure $\Gamma_v$
**Output**: Fitness-Score, $E_s$
    1.  $E_s \leftarrow 0$
    2.  compute $r_{xy}$ , $r_{xy,z}$
    3.  If $r_{xy} \cong r_{xy,z}$ then $Z \perp\!\!\!\perp X, Z \perp\!\!\!\perp Y$
                s.t. if $\exists$ $(X,Z) \in \Gamma_v$ then $E_s \leftarrow E_s$ -1
                s.t. if $\exists$ $(Y,Z) \in \Gamma_v$ then $E_s \leftarrow E_s$ -1
    4.  Lim $\psi \rightarrow 0$
    5.  If $r_{xy,z} \cong \psi$ then $X \perp\!\!\!\perp Y \mid Z, Y \not\perp\!\!\!\perp Z, X \not\perp\!\!\!\perp Z$
                s.t. if $\exists$ $(X,Y) \in \Gamma_v$ then $E_s \leftarrow E_s$ -1
                else $E_v \leftarrow E_s$ +1
    6.  If $r_{xy,z}$ ?  $\psi$ then $X \not\perp\!\!\!\perp Y, Y \not\perp\!\!\!\perp Z, X \not\perp\!\!\!\perp Z$
            But if $r_{xy} \cong \psi$ then $X \not\perp\!\!\!\perp Y \mid Z, Y \not\perp\!\!\!\perp Z, X \not\perp\!\!\!\perp Z$
                s.t. if $\exists$ $(X,Y) \in \Gamma_v$ *s.t.* then $E_s \leftarrow E_s$ -1
                else $E_s \leftarrow E_s$ +1
            Otherwise
                if $\exists$ $(X,Y) \notin \Gamma_v$ then $E_s \leftarrow E_s$ -1 else $E_s$
        $\leftarrow E_s$ +1
    7.  return $E_s$
*End of algorithm*
```

Table 3.3 Algorithm to learn Y-Structure

```
*Algorithm* Y
**Input**: database $D$, putative Y-structure $\Gamma_Y$
**Output**: Fitness-Score, $E_s$
    1.  $E_s \leftarrow 0$
    2.  compute $r_{xy}$ , $r_{ay}$ , $r_{by}$ , $r_{ay,x}$ , $r_{by,x}$
    3.  if $r_{xy} \cong 0$ then $X \perp\!\!\!\perp Y$
                s.t. if $\exists$ $(X,Y) \in \Gamma_v$ then $E_s \leftarrow E_s$ -1
    4.  Lim $\psi \rightarrow 0$
    5.  if $r_{ay.x} \cong \psi$ and $r_{by.x} \cong \psi$ then $X \not\perp\!\!\!\perp Y$, A $\perp\!\!\!\perp$ Y |
        X, B $\perp\!\!\!\perp$ Y | X and hence A $\not\perp\!\!\!\perp$ B | X
                s.t. if $\exists$ $(X,Y) \in \Gamma_v$ then $E_s \leftarrow E_s$ +1
    6.  if $r_{ay.x}$ ?  $\psi$ and $r_{by.x}$ ?  $\psi$
        a.  but $r_{ay} \cong \psi$ and $r_{by} \cong \psi$ then
        $X \not\perp\!\!\!\perp Y$, A $\not\perp\!\!\!\perp$ Y | X, B $\not\perp\!\!\!\perp$ Y | X and hence A $\not\perp\!\!\!\perp$ B |
        X
                s.t. if $\exists$ $(X,Y) \in \Gamma_v$ then $E_s \leftarrow E_s$ +1
        b.  otherwise
        A $\not\perp\!\!\!\perp$ Y, B $\not\perp\!\!\!\perp$ Y and hence A $\perp\!\!\!\perp$ X, B $\perp\!\!\!\perp$ X
                s.t. if $\exists$ $(X,Y) \in \Gamma_v$ then $E_s \leftarrow E_s$ -1
    7.  return $E_s$
*End of algorithm*
```

The equation for partial correlation of first order is given by Eqn.3.2. However, the equation for partial correlation of second order is given below.

$$r_{xy.zw} = \frac{(r_{xy.z} - r_{xz.w} r_{yz.w})}{\sqrt{(1 - r^2_{xz.w})(1 - r^2_{yz.w})}} \tag{3.10}$$

Table 3.2, shows the algorithm for learning V-structure (Fig. 3.3) from data named *D*. ψ is the significance threshold which is set close to 0 in order to avoid false positives. In the V algorithm, the fitness score goes negative if the putative structure does not agree with the data. Table 3.3, shows the algorithm for learning Y-structure (Fig. 3.4) from data named *Y*. The Y-structure has interesting dependence and independence properties which lead to reduction in the condition set compared to learning V-structure Table 3.4, shows the algorithm for learning Markov blanket structure (Fig. 3.5) from data.

Table 3.4 Algorithm to Learn Markov Blanket structure

```
Algorithm MB
Input: database D, putative structure $\Gamma_{MB}$
Output: Fitness-Score, $E_s$
  1.  $E_s \leftarrow 0$
  2.  for each node $V \in \{\Gamma_{MB} - X\}$ s.t. $V \in$ {parents of X)
      a.  Search node set $C_v$ in $\Gamma_{MB}$ that are directly
          conditional on X, s.t. $V \perp\!\!\!\perp C_V \mid X$
      b.  compute $r_{vci.x}$ where i=1..|$C_v$|
              if $r_{vci.x} \cong \psi$ then $E_s \leftarrow E_s + 1$
      c.  Search parent node set $P_{Cv}$ of Ci that are
          conditional on X and Ci, s.t. $V \perp\!\!\!\perp P_{Cv} \mid X, C_i$
      d.  For each $P_{Cvi}$ s.t. $X \not\perp\!\!\!\perp P_{Cv} \mid C_i$ where i=1..| $P_{Cv}$|
              i.  Compute $r_{xpci.ci}$
                      if $r_{xpci.ci}$ ? $\psi$ and $r_{xpci} \cong \psi$ then
                  $E_s \leftarrow E_s + 1$
             ii.  Compute $r_{vpci.xci}$ (second order)
                      if $r_{vpci.xci} \cong \psi$ then $E_s \leftarrow E_s + 1$
      e.  {R} $\in \Gamma_{MB}$ that contains nodes that are not
          comply with the condition set.
  3.  $E_s \leftarrow E_s$ - | R |
  4.  return $E_S$
End of algorithm
```

MB algorithm involves iterative application of conditional independence (CI) tests of increasing orders of partial correlation. The fitness score determines the number of correct independencies and dependencies identified from the input putative structure. Now, in order to compare the performance of the V, Y and MB algorithm, we test the algorithm using a synthetic dataset. The synthetic dataset used is obtained from [97].

i) Experiments

The expression data for the 40 gene artificial network is taken from Gupta *et al.* [97] is where 6 genes have 3 regulatory inputs, 10 genes have 2 regulatory inputs, while the remaining genes have a single regulatory input. We design for 33 interactions to have a time delay of zero, 21 interactions have a time delay of one and 9 have a time delay of two time points. Given this topology of the regulatory network, gene expression values are computed for each one of the 40 genes at 10 time points. The derivatives are computed by employing forward difference. The starting value for the bound for each gene is set to 1.0 and a bound increment value $\delta=0.1$ is employed for computation. The assumed network constituted 63 interactions with known regulatory weights and time delays associated with these interactions.

Recovering the correct structure was evaluated using this artificial network. The algorithm V, Y and MB were compared using data generated by the network. For comparison, we selected the threshold of 0.005 for testing V and Y and a threshold ψ of 0.003 for the MB algorithm providing better accuracy for this algorithm than using a threshold of 0.005.

Structural correctness for the algorithms is evaluated using two types of errors due to extra edges (EE) and missing edges (ME). The total structural error accounting for both errors was evaluated using Eqn.3.10.

$$\mathrm{SE} = \sqrt{EE^2 + ME^2} \qquad (3.11)$$

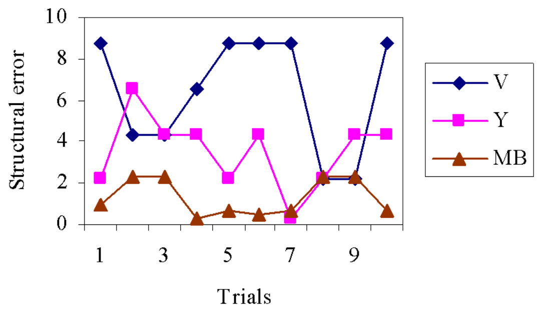


Fig. 3.6 Plot of structural error (SE)

As shown in Fig. 3.6, the algorithm evaluated over 10 trials and the MB algorithm yielded structures with the smallest total structural error over other algorithms under study. Further, we evaluated the algorithm by performing three trials for identifying computational complexity. First trial involves a node and its Markov blanket as input to the algorithms. Second trial involves groups of 3 interconnected V-structures (in order to form Y-structures) as input and third trial involves Y-structures as input. The trials are independently performed exhaustively over the entire artificial network and the average of the number of CI tests performed by individual algorithms in each trial is calculated. This is plotted in Fig. 3.7. The MB performed least number of CI tests during first trial because a MB's (Markov Blankets) were given as input. The reduction in CI tests achieved by the Y algorithm compared to the V algorithm for trial 1 is shown in Fig. 3.7. However, this is not consistent with trial 3. This may be due to the variations in the Y-structures, present in the artificial network. The MB algorithm uses CI tests of order 2 which causes an increase in the computation complexity during trials 2 and 3. However, there is almost no reduction in CI tests of order 1 in V and Y algorithms. As a result, complexity of V and Y are almost equal to the CI tests of MB algorithm except for their structure accuracy and fewer numbers of nodes under consideration.

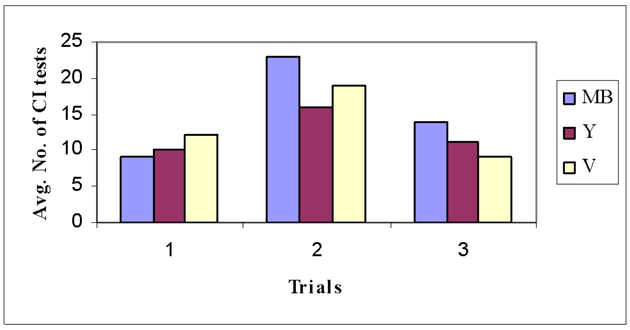
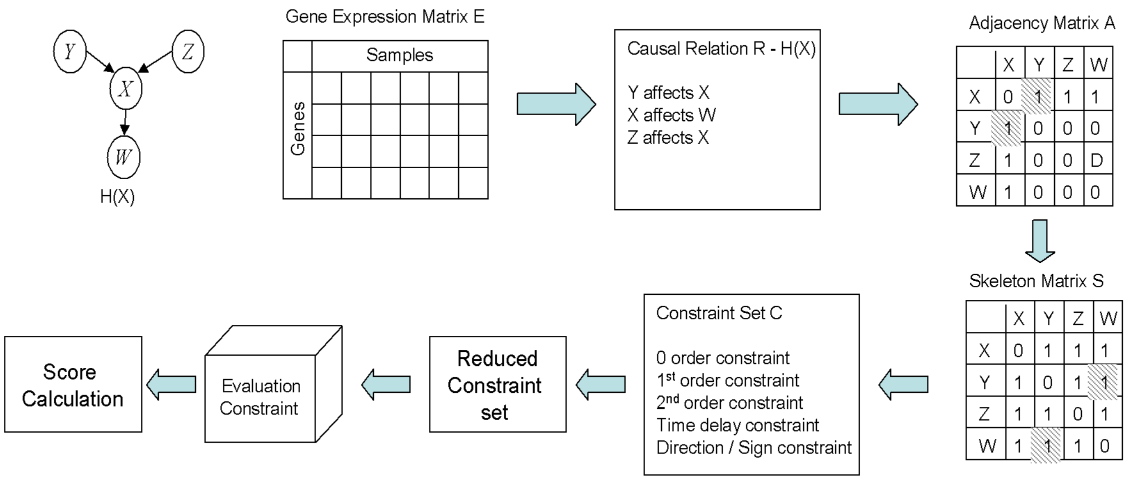


Fig. 3.7 Average number of CI tests observed for 3 sub-model cases

Based on the experimental results, it is clearly evident that the MB method has good computational and causal performance. In the next section, we present a detailed description of the MB method.

3.4 The Markov Blanket-based Method for GRN Inference

The schematic of the proposed causal modeling approach is shown in Fig. 3.8 below. The method requires decomposing a putative network (an individual GRN) into sub models which are the Markov blanket graphs of the type illustrated in Fig. 3.5. To each of the MBs of the network, following steps are then sequentially applied.



	X	Y	Z	W
X	0	1	1	1
Y	1	0	0	0
Z	1	0	0	D
W	1	0	0	0

	X	Y	Z	W
X	0	1	1	1
Y	1	0	1	1
Z	1	1	0	1
W	1	1	1	0

Fig. 3.8 Schematic diagram of the proposed MB method

A) Gene Expression Matrix E

Using dataset D, obtain a matrix E corresponding to the set of genes that are affected by gene X (i.e. MB of gene X). This set will contain parents, children and spouse of gene X.

B) Causal relation R

For the MB of the network $H(X)$, (as shown in Fig. 3.8), the causal relationships between genes are defined as gene X affecting gene Y either directly or indirectly. We thus create n binary causal relations R.

C) Adjacency matrix A

The adjacency matrix A (of size $n \times n$ where n is the number of nodes in the MB) is based directly on the binary relation R and is populated according to Eqn. 3.11 below.

$$A_{ij} = \begin{Bmatrix} 1, & \text{i} \rightarrow \text{j regulation is positive} \\ 0, & \text{otherwise} \\ -1, & \text{i} \rightarrow \text{j regulation is negative} \end{Bmatrix} \tag{3.12}$$

For example, if a causal relation exists in R that a gene X affects gene Y, then the value of element (i, j) corresponding to row i and column j in the adjacency matrix A is set to 1, i.e. A(i, j) = 1. For example, in Fig. 3.8, the shaded cell of the adjacency matrix A shows a direct relationship between gene X and gene Y.

D) Skeleton matrix S

A skeleton matrix S is developed from the adjacency matrix A to include both, the direct and indirect effects observed in a putative MB. While matrix element $A(i, j)$ represents only the direct relationship, the corresponding element of skeleton matrix S also includes the indirect causal relationship between genes corresponding to X and Y. For example, consider an indirect relationship between gene X and Y via gene Z (where Z is any gene other than gene X or gene Y) such that $A(i, k)$ and $A(j, k)$ are both equal to 1. Then *S(i, j)* is evaluated from Binary *{A(i, k) AND A(j, k)}* (for *k* = 1, . . , n). For example, in Fig. 3.8, the shaded cell of the skeleton matrix S shows the indirect relationship between gene *Y* and gene *W*. In this manner, all indirect effects are also captured in the skeleton matrix *S*.

E) Constraints set C

From skeleton matrix S, the direct and indirect effects are respectively converted as conditional dependence (CD) and conditional independence (CI) constraints. These constraints are of the type, *"IF model and data are known, THEN how well the model fits the data"*. The zero order constraint are obtained from the direct interactions while higher order constraints are obtained from the indirect interactions via a condition set. For example, if the condition set contains a single gene, it results in a first order constraint. For our analysis, we have considered up to 2nd order constraints. The CI and CD constraints evaluate to either true (constraint fits the data) or false otherwise.

F) Reduced Constraint set C'

Some tests are not necessary to be implemented and can be eliminated from the constraint set *C*. For example, in Fig. 3.8, we note that gene *Y* and *W* are conditionally independent being conditioned on *X* (i.e. connected via *X*) and further, gene *Z* and *W* are also conditionally independent (conditioned on gene *X*). Hence, gene *Y* and *Z* become conditionally dependent (conditioned on gene *X*). Hence the constraint involving gene *Y* and gene *Z* can be eliminated providing a reduction in the constraint set. Further, instead of conditioning on a single gene *X*, the constraint reduction is also done considering a condition set with more than one gene [67].

G) Constraints Evaluation:

The consistency of the constraints with respect to data is evaluated using the MB fitness scores of Eqn. 3.5 – Eqn. 3.7) explained in the following step viii). Statistical significance test, namely F-test is conducted to check if the correlation coefficients differ significantly from zero value. These tests apply an appropriate threshold p-value to produce satisfactory correlations. Conducting Bonferroni-corrected p-value on few genes, it is thus possible to select a-priori the required threshold p-value.

H) Fitness Score:

As mentioned earlier, the putative network is decomposed into sub models (i.e. the Markov blanket graphs of the type illustrated in Fig. 3.8). Each sub-model is scored for quality of MB structure, direction of causality and sign of regulation.

The score of the overall putative network is obtained as a weighted linear combination of the MBG consistency scores as shown in Eqn. 3.9 and repeated below for easy reference.

$$Fitness\ Score = \sum_i (w_1 * Score_{1_i} + w_2 * Score_{2_i} + w_3 * Score_{3_i})$$

The algorithm is computationally efficient as the order of the number of tests is *O(n log n)*. However, it is possible to further improve its efficiency by applying optimization techniques. An opportunity for computational pay-off exists, stemming from the idea of Constraint Logic Minimization (CLM) which is a form of digital logic circuit. It is believed that conditional independence tests follow a logical pattern [17]. Using this technique, we obtain a lesser number of tests which results in higher accuracy due the presence of noise in data and reduced computation power involved. This is explained in the next section below.

3.5 Constraint Logic Minimization

Although the above presented algorithm is efficient enough, due to the huge size of network search space and the limited amount of microarray data, it is impractical to test each and every constraint. Moreover, with the increase in the condition set needed for causal discovery, more and more CI tests had to be performed, resulting eventually in lower accuracy. By simplifying the complex logic involved with the constraints in the Markov blanket algorithm, the computational efficiency of the MB algorithm can be further enhanced by applying optimization.

The technique to model the Bayesian network accurately by Markov blanket (MB) graph was first proposed by Sprites *et al.* [77] who stated that MB can adequately represent all connections and interactions in a network. Since then, the work on MB has been rapidly expanding with a focus on the study of causality which plays an important role in modeling, analysis and designs of GRNs. Learning any Markov blanket Bayesian network structure and inferring gene networks involves application of constraints. These constraints are typically conditional independence statements. The conditional independence tests used in practice are statistical tests such as partial correlation, mutual information, and conditional probabilities etc. that indicate a causal influence. In order to use the conditional independence tests to reconstruct the structure, several assumptions have to be made, e.g. causal sufficiency, causal Markov and faithfulness [67]. With these assumptions, we can ascertain the existence of an edge, its direction and whether it is positive or negative. The Sprites-Glymour-Scheines (SGS) algorithm [77], used for obtaining a causal DAG from a dataset, assumes that graphs are acyclic. It is formulated using the concept of d-separation in which all possible combinations are tried before determining the existence of an edge between every pair of variables in the dataset. However, the SGS algorithm fails to always assign directions to each of the edges. This limitation of SGS algorithm is overcome by the inductive causation (IC) algorithm, which is capable of assigning directions. Some algorithms do not make use of independence tests but take into account d-separation in order to discover structure from data. For example, Cheng *et al.* [63] applied mutual information instead of conditional independence tests. All these algorithms are referred as constraint based algorithms. Constraint-based algorithms have certain limitations such as poor robustness or computation time which increases exponentially with the number of constraints. These limitations make these approaches impractical for large datasets of tens or even hundreds of variables.

In our proposed causal model approach for constructing GRN explained above and also reported elsewhere [98, 99], the network was inferred by applying the following three sequential steps to identify the sub-structures of a larger network:

i) Perform conditional independence (CI) tests for each node's Markov blanket

ii) Assign direction to the edges and

iii) Assign sign of regulation to the edges. However, due to the huge size of network search space and the limited amount of microarray data, it was impractical to test each and every constraint.

Moreover, with the increase in the condition set needed for causal discovery, more and more CI tests had to be performed, resulting eventually in lower accuracy. By simplifying the complex logic involved with the constraints in the Markov blanket algorithm, the computational efficiency of the MB algorithm (see Section 3.4) can be enhanced thereby resulting in improved accuracy for network reconstruction. In this chapter, we propose a technique for minimizing the constraints and hence the condition set needed for testing the structure with respect to data. The statistical tests following the logic are translated into a Boolean function after which a logic gate minimization technique such as K-map [100] is applied and the minimized logic is translated back to the constraints and used on the data. We have achieved this by a novel independence based algorithm which we refer here as the Markov blanket-Constraint Logic Minimisation (MB-CLM) algorithm. The MB-CLM algorithm heuristically uses Markov Blanket neighborhood of a node and makes model evaluation simple. In order to evaluate and validate a Markov Blanket, there is invariably a need for checking a set of conditions. However, from the available set of alternatives, it is possible to have a potentially smaller set of conditions that can establish the desired conclusion for the given network but with a faster computation speed and increased reliability. This is because a conditioning set S splits the data set into 2 partitions. With a smaller conditioning set, the data set is split into larger partitions thereby making dependence tests more reliable. This smaller or minimal set will fulfill the necessary and sufficient conditions required for GRN reconstruction.

By decomposing the investigation of a large putative network into problem of investigation of smaller MBs, the recovery of the local structure around each node gets greatly facilitated due to the knowledge of the nodes' Markov blankets. Hence, what would otherwise have been a daunting task of employing dependence tests conditioned on an exponentially large number of subsets of large sets of variables (even though most of their members may be irrelevant), we now focus only on the Markov blankets of the nodes involved, making structure discovery faster. We present below the plain version of the MB algorithm that utilizes blanket information for inducing the structure of a Bayesian network.

K-MAP Minimization

The proposed CLM algorithm explained above is based on the well known K-map technique used for logic gate minimization. To illustrate the minimization technique, let us consider an arbitrarily chosen four input (Boolean) network as shown in Fig. 3.9.

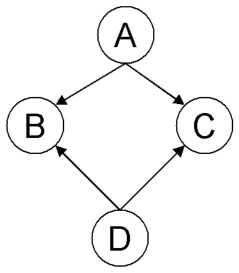


Fig. 3.9 Boolean Network

Let the network have four independent inputs *a, b, c* and *d* are characterized by, say, the following Boolean function to give an output of logic:

$$f(a, b, c, d) = \sum m\ (0, 3, 4, 7, 8, 11, 15) \qquad (3.13)$$

Here, f is the boolean function. The numbers on the RHS are minterms (i.e. decimal value equivalent of the 4 bit inputs). For example, the value 3 on RHS, means that the four bit input combination 0011 (i.e. input *a'b'cd* or the value 3) results in logical 1. The term m indicates that all the values within bracket are minterms.

The above equation, give Eqn 3.12, indicates that any combination of the inputs with values of 0, 3, 4, 7, 8, 11 or 15 would result in an output. Noting that *a=1* and *a'=0*, the above function in Eqn. 3.12 can be expanded as

$$f=a'b'c'd'+a'b'cd+a'bc'd'+ a'bcd+abc'd'+ab'cd+abcd \qquad (3.14)$$

The above Eqn.3.13 is known as a Sum of Product (SOP) of Eqn. 3.12 and the products are the minterms mentioned above.

cd

ab

	00	01	11	10
00	1		1	
01	1		1	
11			1	
10	1		1	

Fig. 3.10 K-Map for the function given by Eqn. 3.14

The K-map for the above function is shown in Fig. 3.10. All rows and columns in the K-map above are unique since only one variable changes its value within its square. The K-map elements are given a value of 1 in such a way that all possible network outputs included in the matrix. The first row, for example has input *a=0 (i.e. a')* and input b equals 0 (i.e. *b'*). Similarly, column 3 for example has both *c= 1* and *d =1*. Thus, an element, for example in row 1, column 3 corresponds to input *a'b'cd =1*. This is the second term on RHS in Eqn. 3.14 above. It can be further noted that between two adjacent elements, only one of the variables changes its value. For example, in Fig. 3.10, the order for input *cd* given as 00, 01, 11, 10 ensures that there is a change of only one input between the adjacent columns. Now let us consider grouping the common terms and minimization of the function using the K-map shown in Fig. 3.10. By grouping:

Four 1s in column 3 for all rows, we get the common term *cd*

Two 1s in column 1 for row 1 and row 2, we get the common term *a'c'd'*

Two 1s in column 1 for row 1 and row 4, we get the common term *b'c'd'*

Considering the above groupings, we can rearrange the RHS terms from Eqn.3.14 appropriately to facilitate logic minimization.
Further, noting that

$$a+a'=1 \tag{3.15}$$

We can simplify Eqn. 3.12 as follows.

$$\begin{aligned} f(a,b,c,d) &= (a'b'cd+abcd+a'bcd+ab'cd)+(a'b'c'd'+a'bc'd')+ab'c'd' \\ &= cd(a'b'+ab+a'b+ab')+a'c'd'(b+b')+ab'c'd' \\ &= cd+a'c'd'+ab'c'd' \end{aligned} \tag{3.16}$$

Since the constraint minimization when applied to the Markov blanket scoring for GRN reconstruction will result in an outcome which is either true or false, the principles of logic gate minimization presented in this section are easily extended and applied to GRN modeling. The variables *a, b, c, d* in Eqn. 3.15 above correspond to constraints that can be either CI tests or tests involving delays and directions.

In the GRN reconstruction method presented earlier (Section 3.4), the network is evaluated at the Markov Blanket (MB) of every node with respect to data resulting in a set of constraints to be satisfied per MB. In general, all these constraints should always be satisfied to validate a true MB with respect to data. Since the dataset under consideration is noisy and high dimensional, it is acceptable if all the constraints are not necessarily satisfied MB validation. For example, consider a MB having, say three constraints. A combination of say two constraints may leave the evaluation of third constraint (don't care value) unnecessary. However, if the two constraints fail, only then the third constraint may need to be evaluated. Since the Markov blanket scoring can be viewed as a logic circuit minimization, we can get a function similar to Eqn. 3.11 and the underlying logic constraints can thus be represented as a logic diagram explained in the previous section resulting in optimizing the computations.

The Constraint minimization version is given as follows:

i. Obtain the Markov blanket *H(X)*. Let the set of constraints be *C*.
ii. Get the constraint set *C* from step (v) of MB algorithm in Section 3.4
iii. Assign binary codes for constraints in constraint set *C*. Use the constraint evaluation table to generate a truth table and logic diagram.
iv. Perform minimization with the help of K-map.
v. Remove the unnecessary constraints from the constraint evaluation step.
vi. Execute the minimized logic on the dataset *D*.

In Fig. 3.11, C represents the set of constraints. Initially the constraints are obtained from the MB algorithm. The above mentioned CLM approach is shown as a CLM phase which takes the constraint set *C* as input and returns a minimized set C_{min} back to the MB algorithm for evaluation and validation.

```
1. C $\leftarrow$ 0
2. LEARN_MB Step F $\rightarrow$ C
3. CLM phase $\leftarrow$ C
4. LEARN_MB Step G $\leftarrow$ $C_{min}$
```

Fig. 3.11 MB Algorithm and CLM Algorithm integration step

The order of complexity for each conditional dependence/independence test taken is *O(nD)*, where *D* is the dataset of input to the algorithm. The computations are required for constructing the table of constraints and for each combination of the variables (genes) included in the constraint test that exists in the data set. As a worst case scenario, each dependence test uses *O(D)* space to store each variable combination of the conditioning constraint set that appears in the data.

The number of constraints tested is usually reported as a measure of the performance of Bayesian net reconstruction algorithms [63, 67, 77, 79]. To determine the number of tests in this algorithm, we assume that the steps 2 and 3 go through MB variables (parents, children, spouses) in an unspecified but fixed order. Therefore, the order of the entire algorithm is *O(n)* in the number of independence tests. The algorithm benefits by further computational optimizations from constraint minimization using the proposed K-map technique.

The Eqn. 3.8 giving the fitness score maximizes true positives. However, it penalizes false positive interactions only to a limited extent due to the presence of α_i and β_i factors. This is because the partial correlations used in the modeling approach, in spite of providing excellent capabilities for entailing meaningful relationships, cause high false discovery rate [46]. As a result, spurious edges may still be present in the final gene network obtained from GA. Hence, an analysis and post processing of the network is necessary to not only eliminate spurious edges for getting a minimal network but also to account for signal transition time delays. This post processing method is presented next.

3.6 Path Analysis – Pruning of false positive edges

In this section, we propose path analysis approach for incorporating missing d-separation, multiple paths, alternative causal explanation, and effect of path time delay. The background on d-seperation and paths in a Markov blanket are provided in Section 2.3.1. The d-separation path analysis algorithm is extended to handle signal transition time delays, and to propagate their effects in the circuit using the *'If..Then'* time functions. This algorithm has four phases: searching, marking for arc deletion, thinning (actual deletion) and finalizing the network. In the first phase, this algorithm finds the sets of paths for each node using its Markov blanket. In Phase 2, each of the paths is analyzed for missing d-separation and d-separation for compliance with important property. This is done using the functions described below. Following that, each of the paths is analyzed for variance conformance and the arcs that are non-conforming are marked for deletion. The result of Phase 2 is a list of arcs marked to be deleted under various conditions. The Phase 3 performs the actual deletion of those arcs which actually affect the fitness of the network. The result of Phase 3 is the final network and Phase 4 finalizes the network and if any errors are identified, the network is sent back to Phase 3.

3.6.1 The d-separation Algorithm

The selection of Markov blanket is based on the d-separation rule of the Bayesian network. When a specific node in the Bayesian network is given, Markov blanket for the attribute is the set of nodes composed of the attribute's parents, its children, and its children's parents. Theoretically, given a Bayesian network structure of the training data set, those nodes that are identified by the Markov blanket indeed block all the influence of the other nodes in the network. This helps to identify the d-separation condition set. The problem is to obtain a network that is minimal in the number of links, or representation size, necessary to fit the data. The properties of Markov blanket and d-separation are combined for this reason.

Concept of missing d-Separation: D-separation property in a DAG implies conditional statistical independence, and missing d-separation implies missing conditional independence D-Connection: X and Y are d-connected if and only if either (i) there is a causal path between them or (ii) there is evidence that renders the two nodes correlated with each other.

Missing d-Separation

Let G be a complete network (path diagram)

1. Initialize counter $k=0$ indicating the maximum number of nodes in the steps below.
2. For each pair of nodes X,Y, connected in G by an egde and possessing more than k neighbors in Markov blanket each, check if for any subset of neighbors of X with cardinality (size) exactly k, the variables X,Y are *conditionally independent*. If so, mark the arc (X,Y) for DELETION from G.
3. $k=k+1$. if more than k neighbors each, go to step 2. Otherwise go to step 4.
4. Repeat for 3 variables X,Y, Z paths, where the end nodes of the path (unconnected edges) are checked for D-separation with their respective neighbors
5. Repeat for each four variables X,Y, Z, T

Similarly, missing D-Connection algorithm can be formulated by replacing the word "separated" with "connected" in the above algorithm.

3.6.2 Pruning Algorithm

The false positive pruning algorithm has the following four phases.

Phase-1 (Searching):

Starting with a fully connected network G from the GA output, search for paths of the type (X, Z, Y) for each gene pair $(X, Y) \in G$ from gene's Markov blanket. The term Z denotes a gene or a set of genes which provide connection between the two genes X and Y under consideration. In Bayesian terminology, the set Z is commonly referred as the d-separating set. A non-zero Z indicates that, *"X is dependent on Y given Z"*.

From the search, list all the d-separation rules for each of the paths in the network which need to be tested. For each of the paths identified in Phase-1 above, list all the edges appearing in these paths.

Phase-2 (Identifying edges for deletion)

We apply the following tests (called beliefs in BN terminology) and identify edges for deletion.

Test 1: Test for missing d-separation and missing d-connection [67] on the condition set obtained in Phase-1. By comparing the d-separation rules (step 2 of Phase-1) and actual reconstructed model, we then evaluate Test1 as either a *pass* or a *fail*.

Test 2: Test alternative path hypothesis by considering various alternative paths possible between pair of genes *X* and *Y*. By comparing each of these alternative paths with the path in the actual model, we evaluate Test2 as either a *pass* or a *fail*. This information will be used in Phase-3 to identify those edges for deletion which are contradictory in the alternative paths.

Test 3: Test for alternative explanation hypothesis (see step viii) of Section 3.4 for clarification). As we apply partial correlation up to 2nd order, we can test paths having a maximum of 4 nodes. The pre-calculated partial coefficients are used to develop a hypothesis for the different possible paths. From this, we identify the path which is statistically most significant. This path is then compared with the corresponding path of the actual model. Test 3 is evaluated as either a *pass* or *fail*.

Test 4: Test for the time delay propagation in all the paths identified in step 1 of Phase-1. This is then converted to time dependent "*If... Then*" statements. The analysis is again limited up to 4 nodes. This limitation allows us to restrict the path delay calculations as computation of max and min delays. If the path delays match the real network, we evaluate the Test4 as a *pass*, otherwise it is *fail*. All the non-conforming edges are identified for deletion.

Phase 3 (Thinning edges by deletion):

It is necessary to avoid contradictory deletions based on four tests given above. For this, with the edges marked for deletion, we perform the following checks.

i) Check whether the deletion of an edge results in the network acquiring:
ii) Island property which causes a section of the network being isolated
iii) Sink property which causes a node to become isolated
iv) Acyclic property which introduces a directed cycle.

If the deletion is acceptable, then we proceed to step-2. Otherwise, the edge is unmarked for deletion. The outcomes of the four tests result in 2^4 (=16) combinations which provide a final decision on deletion of an edge. For example, if outcome of Test 1 and Test 3 are *pass* and the edge is marked for deletion by both models, then permanently delete the edge.

Phase-4 (Validating the network)

For validating the network, we perform the following steps.

Simulate the reconstructed network model in entirety by implementing all the valid arcs and genes with their expression values at a time *t*.

Test and validate the network at a time $(t+\Delta t)$ by examining the pattern of the output responses of various genes.

If errors present, carry out the necessary corrections to fine tune the network followed by revalidation.

3.7 Experiments and Results

Next, using the real life yeast cell cycle data set (see Appendix 1 for details), we carry out following experiments.

3.7.1 MB Method without path analysis

As the MB approach performed better compared to the other two with the artificial dataset (see Section 3.3.4), we further applied this algorithm to cell cycle expression data of Spellman *et al.* [49]. The dataset contains 76 gene arrays of 6177 S. cerevisiae ORFs. Gene expression levels are taken as continuous values. All the 76 samples from cdc15, alpha-factor and cdc28 datasets were used to determine the Markov Blanket structure between genes.

Even without expert knowledge, visual inspection of these sub-networks provides us with ready hypotheses as to why their genes are related. Particularly, inspection of the results from Markov blanket specific to CLB1 (YGR108W ORF) (see Fig. 3.12) is presented in a combination of three transcription factors CLB6, MCM1 and SFF (Swi five factor) and children genes CLB2, CLN2 and SWI5. The parents of children genes, CLN1 (cyclin) and CDC6 (DNA replication initiator) are also interesting to note. The gene regulatory interactions described above find support in the literature [90, 101]. Hence, the study shows that we can recover intricate structures with more accuracy using MB method.

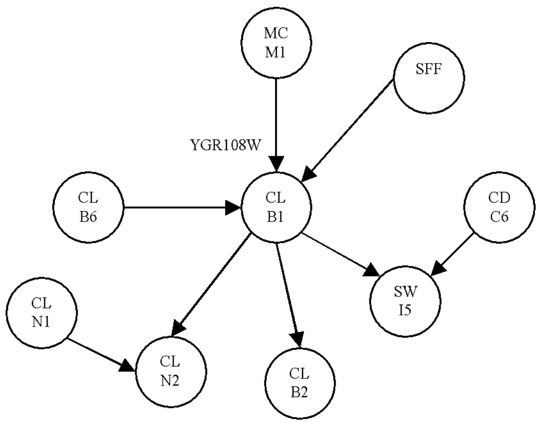
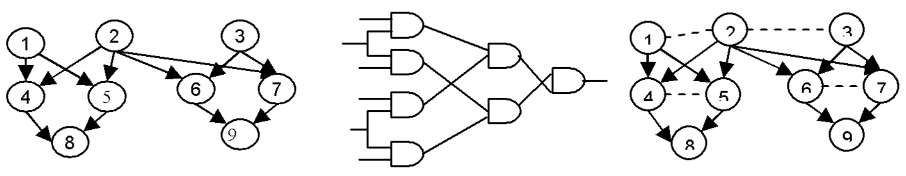


Fig. 3.12 Markov Blanket of gene CLB1

3.7.2 Effect of CLM on MB algorithm

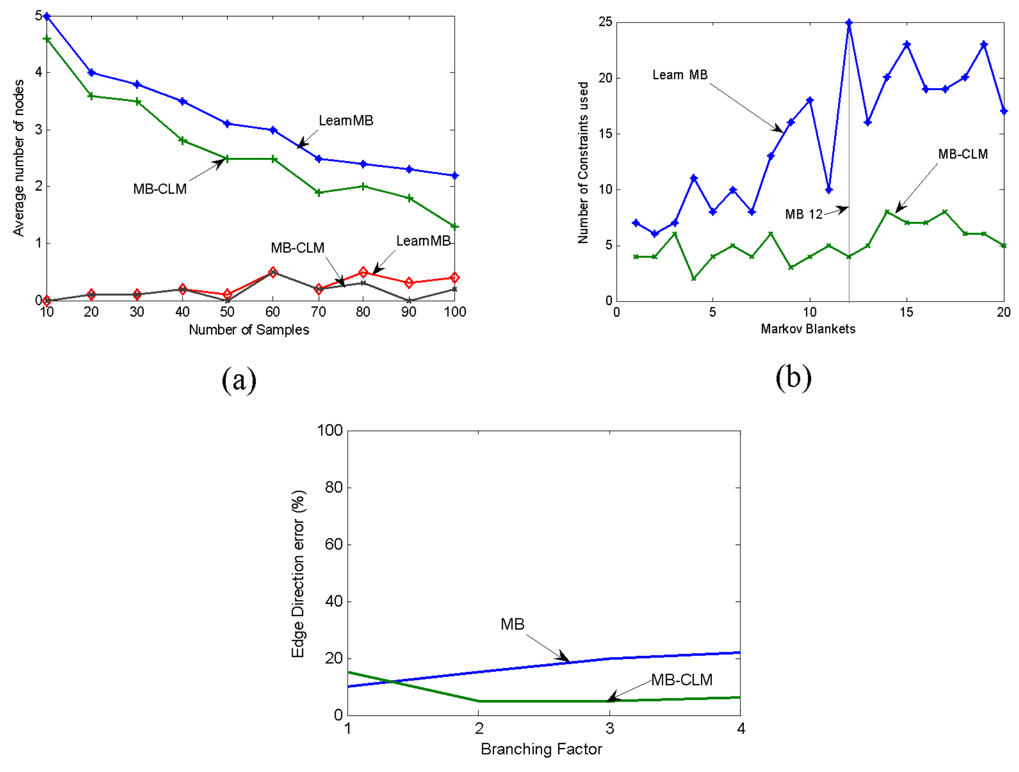
Fig. 3.13 shows an example reconstruction of an artificially constructed synthetic network using MB-CLM technique. Fig. 3.13 (a) shows the original synthetic network, then Fig. 3.13 (b) is the logic circuit corresponding to the constraints involved and Fig. 3.13 (c) shows the reconstructed network using MB-CLM algorithm. Amongst various network architectures possible, we have generated a type referred as random network. The generated network (Fig. 3.13 (a)) is of 3x3 dimensions with an up/down branching factor of 2. The branching factor refers to the number of parents, children and spouses connected to a node. The up branching factor specifies the number of parents of each node directly above it, excluding nodes near the left and right border of the grid, and on the top row.



(a) Synthetic network (b) Logic circuit (c)Reconstructed Network

Fig. 3.13 Synthetic network and minimized constraint logic

In our simulations, we used plain MB algorithm and MB-CLM algorithm with a MB threshold value of 0.90 in both cases and tested the algorithms using synthetic network 5x4 nodes and corresponding synthetic data of upto 100 samples. Fig. 3.14 (a) shows a plot of the number of nodes of the MB incorrectly included or excluded for plain MB algorithm and MB-CLM algorithm, averaged over all nodes in the domain. It can be observed that due to the constraint minimization, the accuracy of results have increased, as a result the number of nodes incorrectly included is less for the MB-CLM algorithm compared to the MB algorithm.



(a) (b)

(c)

Fig. 3.14 Simulation results

Fig. 3.14 (a) shows the number of nodes incorrectly included and incorrectly excluded during the Markov blanket. Fig. 3.14 (b) shows results for a 20 separate Markov blankets with branching factor 3 (in all three (upward, downward and sideways) directions, corresponding blanket size 9). Fig. 3.14 (c) shows the results from using a 5 x 5 network which generated 100 samples that are used for edge direction reconstruction. The branching factor has a threshold value of 0.90

Hence, there is better accuracy and reliability with the MB-CLM algorithm. On the other hand, as can be seen from Fig. 3.14 (a) the use of MB algorithm resulted in a slightly higher number of missing nodes. Although the nodes incorrectly included are very low for both MB and MB-CLM algorithm, the nodes incorrectly excluded fall more rapidly with increasing sample size in the case of MB-CLM algorithm compared to MB algorithm. From Fig. 3.14 (b), it can be observed that MB has very high constraints which are minimized by MB-CLM algorithm. The CLM algorithm thus helps with large reduction of constraints in certain circumstances. The effect on percentage Direction Error (DE) by increasing MB (via branching factor increase) is shown in Fig. 3.14 (c). DE for the MB and the MB-CLM algorithm remains close for lower branching factors but decreases slightly for MB-CLM algorithm with increase in branching factor. The decrease is due to the large number of parents for each node (i.e. more V structures) which provides greater opportunities to recover the directionality of an edge with increased number of tests.

3.7.3 Experimental Results of Path Analysis

Fig.3.15 is the section of the actual yeast network structure and the network after post processing step is carried out where the thick dark lines indicate the barrier and arcs cutting through the barrier were deleted after post processing step. When the fitness measure was re-computed after post processing was carried out, nearly 20% accuracy improvement was noticed in result. This shows that the algorithm delivers more plausible networks close to the actual network.

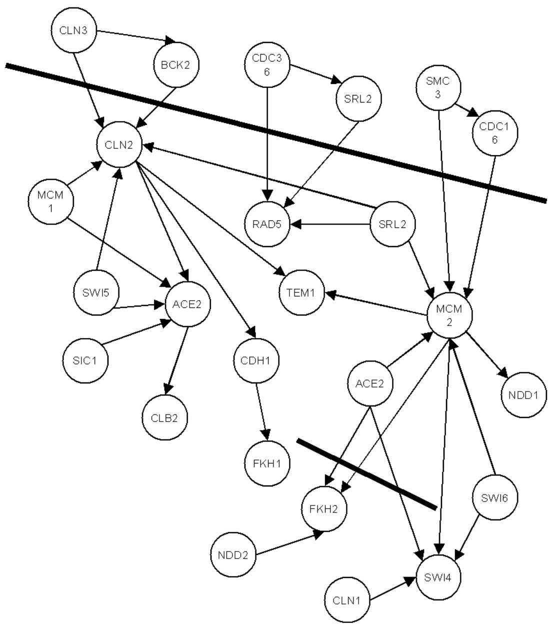


Fig. 3.15 Results from pruning the yeast network

3.8 Summary

In this chapter, we present a novel Markov blanket based approach to learning gene regulatory network which decomposes the network into Markov blankets of each gene (comprising of parents, children and parents of children). To minimize the computational overhead, a constraint minimization technique to speed up in the case of large datasets is also proposed. Further, a novel post processing path analysis technique to prune the network of spurious interactions. The preliminary results using real yeast dataset test the modeling technique. The results are promising as they not only identify selected biological interactions reported in the literature but were also able to detect spurious regulatory relationships predicted by the model. For more rigorous experiments, it is necessary to develop realistic synthetic datasets which can enable variation of parameters. In the next chapter, we propose techniques for generation of these datasets as well as its application for conducting detailed experiments.

Chapter 4

4 Synthetic Dataset and Model Analysis

4.1 Introduction

In the previous chapter, the development of causal model and associated algorithms were studied with the aid of real life dataset. However, with the documentation of real life dataset not always complete; the underlying network that produces the data remains unknown. This makes any validation, robustness analysis of models and algorithms or their comparison with other existing techniques quite difficult. A synthetically reconstructed GRN, while preserving the characteristics of the underlying data generation system, allows experiments to be performed using any new method to investigate the effect of parametric variations. These artificial but realistic GRN networks provide a simulation environment similar to a real-life laboratory microarray experiment and a mechanism for robustness studies reconstruction methods to individual and combination of parametric changes. Studies involving complicated interactions as well as parametric variations such as topology, noise (background and experimental noise) and time delays or number of samples can be carried out by the proposed synthetic GRN networks.

Limited literature for generating synthetic data for GRN reconstruction is available. Mendez *et al.* [2] proposed a method based on differential equations for generating synthetic microarray data. The method allowed variation of only noise and topology parameters and did not include the flexibility of varying single or combination of parameters for validating individual features of the GRN methods. Eisen *et al.* [38] generated synthetic dataset and applied for studying hierarchical clustering for gene expression data. As the method suffered from the lack of knowledge about the GRN under study, any conclusion vis-à-vis the underlying biology became uncertain. Further, because the data sets were different in each of the studies carried out, it was not possible to make any comparisons amongst studies that employed this approach. Friedman *et al.* [17] generated a Boolean synthetic data to validate the robustness of their Bayesian methods. Although useful for generating synthetic datasets, none of these techniques were suitable to examine model specific features such as time-delays, feedback loops, dynamic behavior, etc. Furthermore, all these techniques were limited in their ability to generate a variety of synthetic networks at different stages of refinement of GRN reconstruction methods.

In this chapter, in Section 4.2 we present a new approach for synthetically generating gene networks using causal relationships. The generated synthetic networks presented in Section 4.3 are realistic have varying topologies such as small world, random, scale free, or hierarchical topologies based on the well-defined GRN properties. The proposed method for generation of synthetic networks allows for various parametric variations, such as, network topology, varying levels of complexity of interaction, time delays, number of samples and amount of noise in the data. In Section 4.4, the datasets are also applied for validation of the robustness of the causal GRN modeling method presented in Chapter 3. Section 4.5 provides a discussion of the results and Section 4.6 gives the summary of the chapter.

4.2 Generation of Synthetic Data

The synthetic network generator, written in MATLAB, offers an option for choice of topologies that determines the structure of the network and specifies interactions between the genes. With this option, we can generate any number of networks having different topologies. In the next step, by choosing interactions and setting equation parameters, the full dynamics of the gene network (such as feedback loops, oscillations and so on) is described and can be implemented in specified pre-defined ways to produce a required level of complexity of gene interactions. Next, for generating discrete samples, the continuous responses of the genes in the synthetic network are sampled at different time instants which produce a noiseless time course data. Next, to make the sampled data realistic, time delays are added to the samples in a specified manner. Following this, noise is added to the data according to the Gaussian or gamma distributions. Finally, gene expression ratios are calculated which realistically represent the real-life microarray data set. The flow chart of the mechanism of proposed system for synthetic network generation is shown in Fig. 4.1. The entire process of generating the network topology and corresponding gene interactions is described in detail in following sub sections.

4.2.1 Network Topology

As mentioned earlier, the first step of synthetic data generation is to define a network topology. A topology is chosen by setting following three parameters:

i. Total number of genes in the network,
ii. Distribution of the in-degree of connectivity (i.e. the distribution of the number of parents per gene) and
iii. Distribution of the outgoing degree of connectivity (i.e. the distribution of the number of children per gene).

Based on the incoming and outgoing degree distribution parameters mentioned above, four different topologies are available for selection (with corresponding distribution provided in parenthesis):

- Random topology (Poisson distribution)
- Scale Free topology (power law distribution)
- Small World topology (power law distribution with small average distance between genes)
- Hierarchical topology (power law distribution with inherent modular structure)

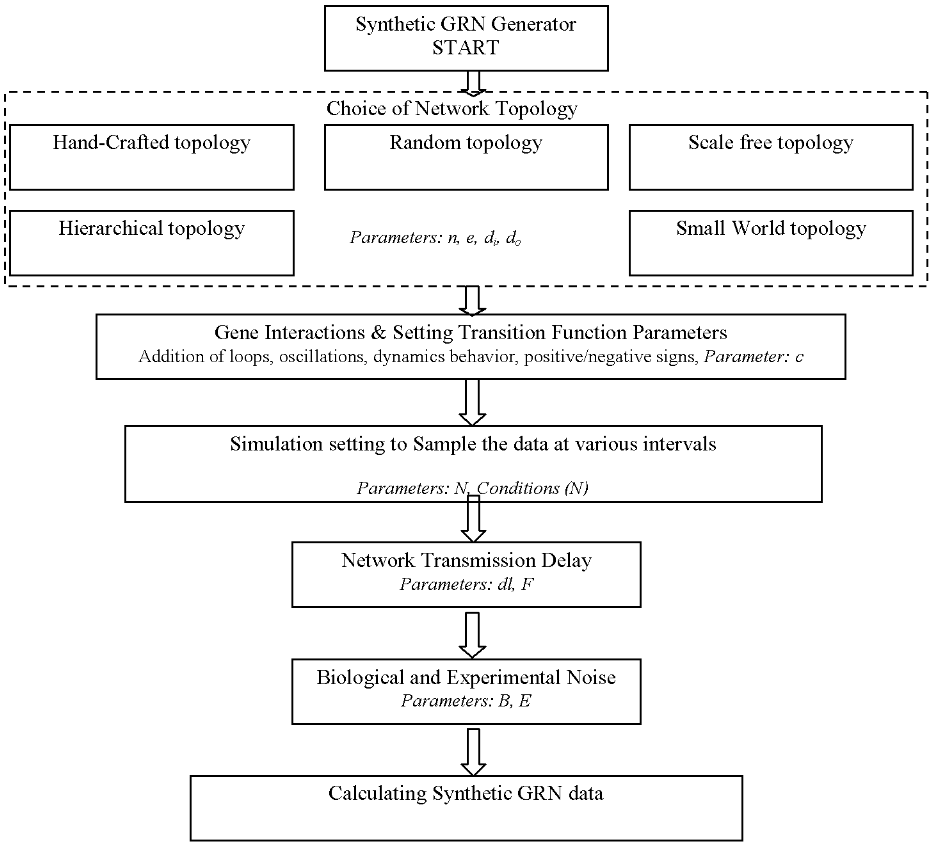

Synthetic GRN Generator START
Choice of Network Topology
Hand-Crafted topology
Random topology
Scale free topology
Hierarchical topology
Parameters: n, e, d_i, d_o
Small World topology
Gene Interactions & Setting Transition Function Parameters
Addition of loops, oscillations, dynamics behavior, positive/negative signs, Parameter: c
Simulation setting to Sample the data at various intervals
Parameters: N, Conditions (N)
Network Transmission Delay
Parameters: dl, F
Biological and Experimental Noise
Parameters: B, E
Calculating Synthetic GRN data

Fig. 4.1 Proposed methodology of synthetic gene expression data generation, The symbols used are: n - number of genes, e – number of edges, di – incoming degree distribution, do – outgoing degree distribution, c – percentage of complex interactions, N – number of samples, Condition (N) – specifies experimental conditions for each sample as in real each sample is an experiment, dl – delay levels, F – probability distribution of delays, B – percentage of biological noise in terms of hidden nodes, E – percentage of experimental noise

In random topology (RND), the connectivity degree follows a Poisson distribution. The nodes that deviate from the average are rare and decrease exponentially and the clustering coefficient is independent of a node's degree of connectivity [102]. In Scale Free (SF) topology [103], the connectivity degree follows a power law distribution, i.e. the behavior of a network system is controlled by few important nodes. Majority of nodes have only a few connections, while some special nodes connect with many other nodes forming a hub, i.e., most nodes are poorly connected, while a few are highly connected (Hubs). In a Small World networks (SW) [104], the mean shortest path is $l \sim log(N)$ indicating that most nodes are connected by a short path. The SW networks are characterized by large Clustering Coefficient and small Average Path Length. The Hierarchical network (HR) [105] integrates a scale-free topology with an inherent modular structure by generating a network that has a power-law degree distribution with degree exponent $\gamma = 1 + ln4/ln3 = 2.26$. In cases where the aforementioned topological types are not appropriate due to the uncertainty of GRN topology, we propose another topology, which we will refer henceforth as, 'handcrafted topology'(HC).

The choice of any of the network topology is user-definable and can be used for checking robustness of algorithm against topology. To generate a network topology close to real life GRN, network structures previously described in biological literature such as E. coli [25] and S. cerevisiae [51] were taken into account. These networks are partially random and partially scale free i.e. the distribution of the incoming degree of connectivity follows a Poisson distribution (random topology) while the distribution of the outgoing degree of connectivity follows a power-law (scale free topology). A single topology or combinations of two or more topologies to generate the gene network structure is user definable.

At this stage, the network structure is without any complex interactions, such as self loops, oscillations and dynamic behaviour. In the next section, we present the inclusion of these features to the network topology.

4.2.2 Gene Interactions and Transition Function Parameters

After generating the topology, transition functions representing the regulatory interactions between the genes are assigned to the edges in the network as follows:

i. Choosing the regulatory interactions
ii. Setting the transition function parameters

The entire synthetic modeling of gene networks essentially considers a causal interaction of genetic regulation. It considers each gene to be directly affected by number of other genes and represents the interaction as directed edges. A transition function defines the relationship between gene and its parent genes. The genes are represented as continuous variables rather than discrete variables, i.e. synthetic gene expression values are continuous rather than 0 or 1. First, while choosing the regulatory interactions, the genes are represented as activators or repressors. Our proposed method of network modeling allows for this positive or negative linear causal relationship between the input (i.e. parent) genes and the gene under consideration. Mathematically, these network models are based on set of linear causal equations. Each equation corresponds to gene expression which is a function of a positive (activation) and negative (repression) terms. When a given gene interacts with more than one regulator, different regulators can either act independently or in a more complex manner (such as complex combinational, short term co-activation, co-repression or a combination) on the target genes resulting in different interactions such as feedback loops, oscillations and dynamic behavior.

To incorporate such complexities, for each combination of a gene and its regulators, appropriate equation is selected, depending on the number of activators and repressors and on the user-defined settings that control the fraction of complex interactions. For genes involved in cycles, it is possible that not all inputs of their transition function are known during loop propagation. To model these loops, an approximation compatible with the steady-state transition functions is chosen. This approximation is represented by a parameter to represent complex interactions. It is an extremely useful parameter because it allows initial performance evaluation of a method to be done on relatively easy problems (e.g. small noiseless networks without complex interactions between regulators). Increasingly difficult data sets can subsequently be generated as the GRN inference method is improved or refined. Again, setting transition function parameters involves choosing appropriate correlation parameter settings of the transition function equations. The strength of correlation is an important parameter and is chosen from a distribution that allows a large variation of interaction that are likely to occur in true networks (including linear activation functions, sigmoid functions, sinusoidal functions, etc.), while avoiding very steep transition functions. To explain a simple chain interaction in the network considers, for example, that x causes y and y causes z. That is, $x \to y \to z$

$$x(t) = A\sin(Bt)\ ;\ y(t) = x(t)\ ;\ z(t) = y(t) \tag{4.1}$$

The expression $x(t)$ is a sinusoid with amplitude A, time period $2\pi/B$ where B is angular frequency. In this case, the strength of correlation between x and y is 1, so the signals are equal, but varied based on parametric specification.

4.2.3 Data Samples

Using the continuous gene expression output (resulting from the equations written for each node of the synthetic network), data is sampled at either fixed or irregular time spacing between gene expressions. The number of samples and the time step for sampling can be chosen either randomly or it can also be user defined. The sampled data represents the temporal state of synthetic network under different experimental conditions. This is similar to real microarray experiments where each sample of the dataset is an experiment that is repeated at fixed or irregular intervals of time. At this stage, various settings needed for simulation of the network per each sample (simulating a real experiment setup) for *N* samples are complete. However, note that the data representing real life conditions is not yet generated as time delay and noise component are yet to be added.

4.2.4 Network Transmission Delays

A delay in transmission of signals emitted by genes, being an important characteristic of all gene networks; it is important to realistically implement this feature in synthetic datasets. In the proposed modeling approach, we implement the delay levels as a user defined parameter which is nothing but the maximum number of samples on which the delay can be experienced. Further, to make the modeling more realistic, we have also made it possible to specify the fraction of interactions which have delays. Based on the choice of this parameter, a delay distribution is obtained for the links between the genes. Delays are implemented by simply reassigning a new simulation setting for a particular sample explained in Section 4.2.4 based on the delays assigned. This simulates the delay in the real microarray dataset. The fraction of links involved in time delay is determined using a known probability density in case it is not user defined. Investigations involving time delay parameter variation can thus be carried out on the datasets by incorporating/eliminating time delays.

4.2.5 Biological and Experimental Noise

A real life microarray data contains two types of noises, namely biological and experimental. The biological noise corresponds to stochastic variations in gene expression, and this noise is unrelated to the applied experimental procedures. It is present due to, for example, environmental conditions such as temperature, pressure, etc. While experimental noise is the noise due to the technique used to extract the data. Both these noises also should be appropriately included in the simulated data.

Briefly, biological noise is added by the presence of hidden background nodes which are either genes or conditions and experimental noise is added as Gaussian white noise. First, the background hidden node (for incorporating biological noise), which is a parameter to choose the amount of background noise, is user defined. The equations of the background noise nodes are generally uncorrelated to the genes on which they are acting. A limited number of input nodes are selected that mimic the external conditions and consider the genes not linked to these input genes act as background nodes. These are now part of the simulation set up while the data is not generated.

As the real microarray data also has experimental noise, three user defined choices for addition of experimental noise are made available:

i) Log normal
ii) Gaussian
iii) Gamma distributions

All these distributions take a percentage of the amount of noise as input which is then applied to make the final output data noisy. However, this experimental noise is added only after the simulated microarray data is generated. This is explained in Section 4.2.7.

4.2.6 Synthetic Network Generator Parameters

The entire flow chart for the generation of synthetic data is given in Fig. 4.1 which also shows the system and the parameters controlling the synthetic data generation at every step of the process. These parameters which are listed below can each be varied independently either before or during the simulation process for conducting simulated experiments with synthetic data:

i) Choice of source network

ii) Size of the network in number of nodes

iii) Number of background nodes

iv) Number of available experiments and samples for each condition

v) Level of stochastic and experimental noise

vi) Fraction of complex interactions

4.2.7 Calculating Synthetic GRN data

Using the synthetic network generator described earlier, simulations are next performed to generate the synthetic microarray data. The genes without regulatory inputs are assigned an arbitrary expression level which can be changed during an experiment (sample). The expression levels of the genes in the network are calculated, as specified by their transition functions, starting from the input genes. After these noise-free expression values are computed, noise is then appropriately incorporated in the data to reflect noise present in the real microarray data. These computed noisy expression values can be used for analyzing the noise which a GRN reconstruction method under investigation can handle. This feature of adding noise enables the comparison of level of noise in dataset on the reconstruction algorithms. A gene expression profile experiment for different time t corresponds to a vector $[x1(t) \dots xn(t)]$. For a set of N samples, a $n \times N$ matrix is constructed which is the final synthetically generated microarray dataset. This dataset can be used for investigation and evaluation of various GRN reconstruction algorithms. In the next section we present the synthetic datasets generated for the testing of the model.

4.3 Synthetic Datasets

In order to conduct tests using synthetic data set, several datasets are created by varying network generator parameters (one or two at a time). The groups of data sets which have similar variations are categorized into one of the four groups *A, B, C* or *D* (see Table 4.1). Although the experiments involved significantly large number of data sets to test robustness of GRN methods, due to space restriction, only a limited number of important models have been included in the paper and shown in Table 4.1.

The Group *A* consists of a set of synthetic network models which are used for investigating methods for their robustness against network topology. With this group, we carry out an initial level of testing since it contains no complex interactions and also because the effect of the noise is kept low. Different sample sizes help determine accuracy of reconstruction as generally most methods require higher sample size data to make accurate estimations.

The Group *B* networks compare two different network topologies, namely SF and RND. Compared to Group *A*, these are large sized networks of 500 genes and 500 interactions. Fig. 4.2 (a) shows networks that follow a random topology (RND) while the network shown Fig. 4.2 (b) is a scale-free (SF) network. From the figure, we can observe the differences resulting due to two differing topologies. The random topology has arbitrary arrangement of links throughout the network while the scale free network has hubs with large proportion of links in the top right corner of the figure while lesser number of links in the rest of the figure. Note that the number of genes and gene interactions is the same for the two cases under consideration. Since scalability is an important feature of GRN algorithms, this group enables to justify if the algorithm is robust in terms of size.

In Group *C*, the number of genes in the networks is kept fixed at 50 and the topology chosen for study is Scale Free. The number of links is varied as 50, 100, 200. This group is useful for checking robustness of methods with respect to density of connectivity (i.e. no. of parents per gene) along with accuracy with respect to number of samples. The Group *D* is designed to test the combinational effect of density of connectivity and also to include varying delays and noise intensity parameters resulting in an increasing average number of connections per gene.

Table 4.1 The Synthetic data sets are organized in four groups A, B, C, D. Column 2 gives different network topologies: Scale Free (SF), Small World (SW), Random (RND) and Handcrafted (HC). For each group, column 3 shows the number of repeated models for a given experiment. Column 4 and column 5 respectively give the number of genes and the edges in a given model. Column 6 gives the % fraction of complex interactions. Column 7 gives the network transmission delay. Column 8 gives the number of parents while column 9 gives the %ge noise of each model. Column 10 gives number of samples for each condition.

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Group	Topology	No. of Models	Genes	Edges	% Complexity	Delay	No. of parents	% Noise	Samples
A	SF	50	100	200	20	0	2	1	20, 50, 100
	SW	50	100	200	20	0	2	1	20, 50, 100
	RND	50	100	200	20	0	2	1	20, 50, 100
	HC	50	100	200	20	0	2	1	20, 50, 100
B	SF	5	500	500	40	2	5	5	50
	RND	5	500	500	40	2	5	5	50
C	SF	50	50	50	20	1	3	5	50
	SF	50	50	100	20	1	4	5	50
	SF	50	50	200	20	1	7	5	50
D	SF	10	100	200	40	1	1	5	20, 50
	SF	10	100	200	30	2	2	1	20, 50
	SF	10	100	200	50	-2	3	5	20, 50
	SF	10	100	200	10	3	4	1	20, 50
	SF	10	100	200	40	-3	4	5	20, 50
	SF	10	100	200	30	4	5	1	20, 50
	SF	10	100	200	50	0	3	10	20, 50
	SF	10	100	200	10	0	2	1	20, 50

The *D* group tests are for advanced level testing of GRN algorithms as the data generated is from a complex complicated network of interactions. Because these gene networks are generated with random connectivity for each of the experiment rows in Table 4.1, we repeated the generation of models for specified number of times (see column 3) and took the average results from each row to get a synthetic dataset which is close to real dataset.

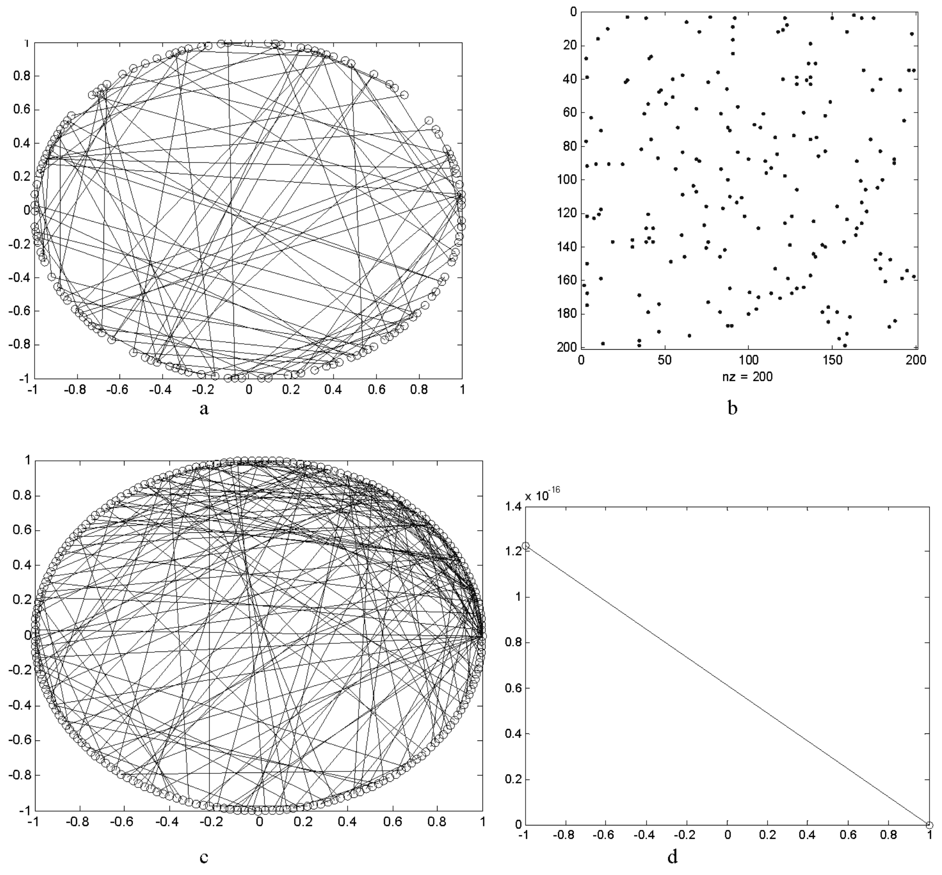


Fig. 4.2 Two illustrative network topologies of B Group: a- RND Network Topology *(n=200, e = 100, di = dp = Poisson pdf)*, b – Adjacency matrix *(n x n)* of the random topology, c - Scale Free Network Topology *(n=200, di = dp = Power law pdf)*, d – Power Law function, f(x) on y-axis

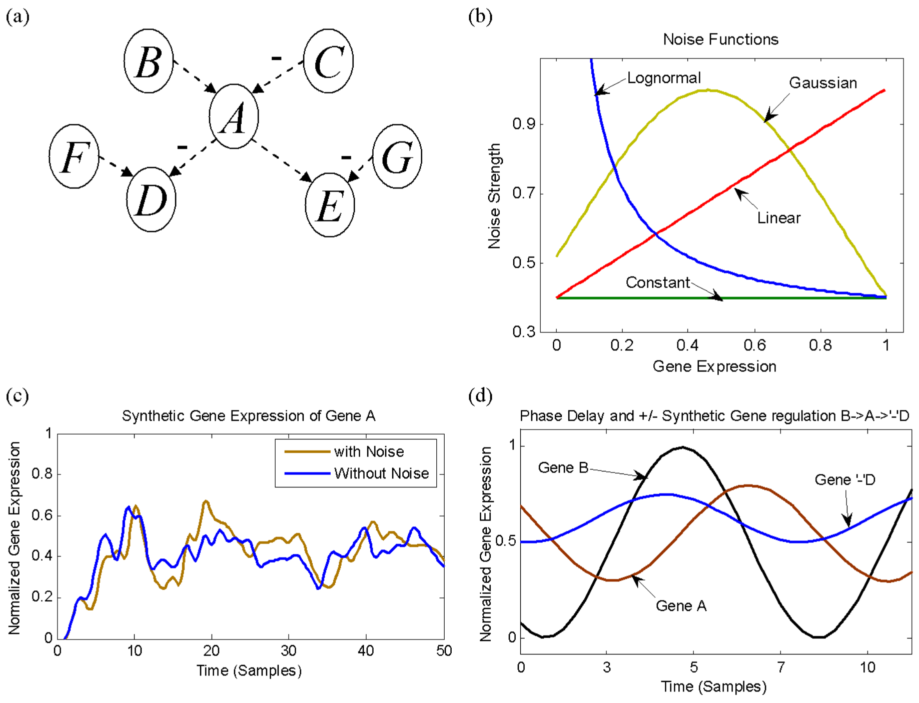


Fig. 4.3 Simulation results: (a) A subset of genes of the example network *(labeled A to G)*. This sub-network has two input genes and contains repressor gene *D*. (b) The different noise functions used in the simulation. (c) The expression gene *A* with and without addition of noise. (d) Shows phase shift (time delay) and plus/ minus regulation between *B*→*A*→*D*

The simulation results are provided above in Fig. 4.3.

4.4 Analysis of Proposed Causal Model

We note that the synthetic datasets are useful as they facilitate evaluation of robustness by variation of parameters to generate large number of related synthetic data. For our investigations on the robustness of the proposed Markov blanket based causal model presented in Chapter 3, the synthetic data generation is performed using the technique presented in the previous section. A 40 gene artificial network containing 200 edges, 5 hidden nodes and 2 hidden node combinations is generated.

It is built with a topology referred to as hand crafted in which its parameters are randomly assigned. The dataset generated has 150 samples. For evaluating the proposed method and its robustness to parametric variations, number of artificial networks and corresponding datasets are also generated using the parameter settings from Table 4.2. Since the networks are significantly different from each other, the error measured is normalized to enable comparison.

Table 4.2 Simulation Setting: Synthetic Dataset Variations

Synthetic	delay	No. of	Noise%	% Edges	
Network		parents		Add	Remove
0	-	2	1%	-	-
1	1	2	1%	5%	5%
2	2	3	1%-	-	10%
3	-2	3	5%	10%	-
4	3	4	1%	5%	10%
5	-3	4	5%	10%	5%
6	4	5	1%	5%	5%

The search for optimal network structure was carried out by implementing following parameter settings for the GA.

Table 4.3 Genetic Algorithm Settings

Parameter	Value
Crossover probability	0.1
Mutation probability	0.8 (evenly distributed over 4 types of mutation)
Population Size	200 - 600
Iterations	50 - 100 / based on improvement

These structures are independently evaluated by the Receiver Operating Characteristic (ROC) and the Area Under Curve (AUC). The ROC curve is obtained by plotting the *sensitivity* versus *(1-specificity)* for different values of the error term and describes the trade off between sensitivity and specificity. From the inferred model, each pair of genes that contain an edge and relative delay time can be represented by a relationship consisting of a leading gene, a lagging gene, a relative lag time, a co-regulation/causal value (which we call as influence score), and the direction of association (positive or negative). When the relative lag time is zero, the leading gene and the lagging gene are interchangeable except for causal faithfulness. For scoring, the input genes receive higher scores compared to output genes and are selected according to the MB cut-off parameter. Next, genes that are highly and consistently expressed over the conditions identified in the first step are selected according to a second cutoff parameter (the path analysis threshold). The algorithm ensures a robust identification of relevant conditions (and hence of the output genes) in spite of unrelated 'noise' genes to the input. This results in a model which consists of a set of genes along with the regulating conditions. However, due to the difficulties posed in estimating delay and direction of orientation, it is possible that two genes that appear to be associated with each other, in reality, may be lacking such an association. It is also probable that a time ordering (observed for two associated genes) is misleading, either because they are not associated or because they are associated without a lag in time.

For this reason, following classification is applied in calculating the error.

i) *Correct:* the edge is present and oriented in the same direction in both the graphs (reference and observed) or it is absent in both the networks
ii) *Committed:* the edge is absent in the reference network but present in the selected graph
iii) *Omitted:* the edge is present in the reference graph but absent in the selected graph

iv) *Reversed:* the edge is present in both network but points in opposite directions (referred negative lag)

v) *Unresolved:* the edge is oriented in the reference graph and although present in the selected graph, it cannot be oriented (referred zero lag)

vi) *Over-determined:* the edge cannot be oriented in the reference graph but is oriented in the selected graph (referred non zero lag)

vii) *Lag error:* the edges are oriented similarly in the two networks but their lags are different.

These error properties fall into the following three groups.

i) *Errors of commission:* Outcome (ii) can occur only if a edge is missing in the true (and therefore, reference) graph.

ii) *Edge errors:* Outcomes from (iii) till (vi) can occur only if an edge is present in the reference graph.

iii) *Lag or delay error:* Outcomes (iv), (v), (vi) and (vii) can occur only if the delay is greater than 0.

For GRN modeling, the sensitivity can be considered as a measure of the proportion of true regulations that are correctly predicted with their sign and lag. Similarly, the false positive rate (Table 4.4) is a measure of the proportion of non-regulations that are wrongly predicted as regulations. Some false positives are more informative than others, since they link genes that are not in a direct parent–child relationship but are still nearby in the pathway. Hence, we categorize false positives as those appearing either from relatives (i.e. informative) or from strangers (i.e. uninformative). The relatives consist of siblings, uncles and children while strangers consist of all those genes that are not relatives. The most informative false positive edges of a gene are from its grandparents, since they are upstream in the pathway and only one step removed from the true parent.

Table 4.4 Equations: Sensitivity and Specificity

FN = $\{X \rightarrow Y \mid X \rightarrow Y$ is in the original graph and $X \rightarrow Y$ is not in the obtained graph$\}$
TP = $\{X \rightarrow Y \mid X \rightarrow Y$ is in the original graph and also in the obtained graph$\}$
TN = $\{X \rightarrow Y \mid X \rightarrow Y$ is not in the original graph and X → Y is not in the obtained graph$\}$
FP = $\{X \rightarrow Y \mid X \rightarrow Y$ is not in the original graph and X → Y is in the obtained graph$\}$

To evaluate the accuracy of a recovered network, we use two general measures defined as follows:

$$Sensitivity = TP/TP+FN$$

$$Specificity = FP/TN+FP$$

$$Precision = TP/TP+FP$$

For ROC curves, AUC (ROC) close to 0.5 corresponds to a random forecast, $AUC(ROC) < 0.7$ is considered poor, $AUC(ROC) < 0.8$ is fair and $AUC(ROC) > 0.8$ is good. Robustness analysis is carried out by considering following perturbations in the system parameters:

i) Length of time series, i.e. delay, structure and vertex in-degree
ii) Noise
iii) Network size and topology.

We have divided the synthetic data experiments into two parts, based on the two datasets given in Table 4.2 and Table 4.6 respectively. Comparison of the proposed method with two well known methods [17, 54] is also carried out. These perturbation analyses are presented next.

4.4.1 Effect of number of time points in the time series

An artificial network of 40 genes and 200 connections having arbitrary number of parents per gene, arbitrary delays and noise, and without any cycles is investigated (Table 4.2). A time series data with the number of samples $N >= 50$ is usually not available from a wet lab experiment. Hence simulations are performed for datasets with $N =< 50$ for three different length of time series $N = 10$, $N = 25$, and $N = 50$. The TP and FP values for the 3 settings are given below.

Table 4.5 Results: Effect of Samples

Samples/Accuracy	TP	FP
50	200	0
25	153	0
10	95	52

For the dataset with $N = 50$ generated with a noisy regulation, it is observed that all true edges can be recovered without incurring any false spurious edges. With $N=25$, we were able to recover 75% of the true edges with a zero FP rate. With N decreased further to a very low $N=10$, there were 25% FPs observed. The tabulated results given in Table 4.5 clearly show that these models perform satisfactorily.

4.4.2 Effect of delay

The effect of delay is determined by computing the Root Mean Square Error (RMSE) as follows.

$$\text{RMSE}\left(\text{edge delay}\right) = \frac{\sqrt{\sum\left(\text{observed delay} - \text{actual delay}\right)^2}}{n_e} \tag{4.2}$$

where n is total number of edges. The RMSE values shown in Fig. 4.4 are the averaged values obtained from 20 runs. The figure shows that the edge delay error increases linearly with increase in complexity of dataset but it is robust with increase/decrease in the number of samples. However, if there are fewer samples, it is difficult to estimate delays accurately.

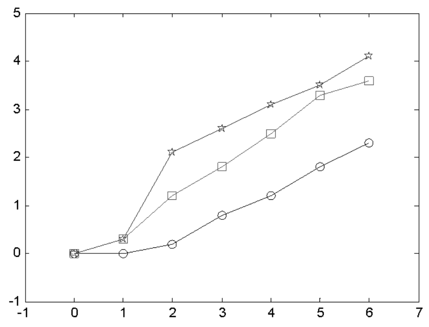


Fig. 4.4 Variation of RMSE with delay settings. Legend: Axes: X-axis - delay settings, Y-axis - RMSE. Plots: ○ – 50 samples, □ – 25 samples and * –10 samples

4.4.3 Effect on structure

The ability of the method to provide structural correctness is evaluated by considering two types of errors, i.e. error due to extra edges (*EE*) and error due to missing edges (*ME*). The total structural error, accounting for both the errors, is evaluated in a manner similar to RMSE as follows:

$$SE = \sqrt{EE^2 + ME^2} \tag{4.3}$$

This structural error metric, SE determines the error on the entire topology of the network. The results are shown in Fig. 4.5 and are plots of *SE* vs synthetic dataset variations in Table 4.4. (Note: x-axis refers to the index of the synthetic networks). From the figure, we can see that the structure error increases linearly with an increase in the complexity of dataset but it is robust and remains constant with an increase/decrease in the number of samples.

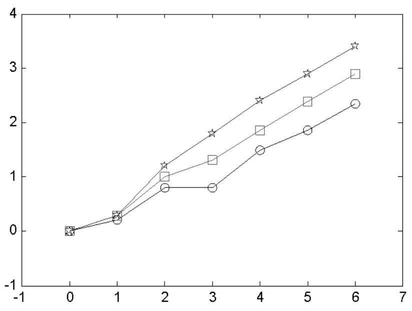


Fig. 4.5 Variation of Structure Error (SE) with the structure Legend: X-axis: Structure settings, Y-axis: Structure Error Plots: ○ – 50 samples, □ – 25 samples and * –10 samples

4.4.4 Effect of vertex in-degree

To determine errors in the number of parent genes recovered, following metric is considered.

$$\text{RMSE}\left(\text{parents / gene}\right) = \frac{\sqrt{\sum\left(\text{observed parents} - \text{actual parents}\right)^2}}{n_e} \qquad (4.4)$$

Fig. 4.6 shows that the error due to the vertex in-degree increases with an increase in the complexity of dataset. Since some datasets have same in-degree, these results are not shown in the figure. The model is still robust and remains equal with an increase/decrease of number of samples with the variation in in-degree.

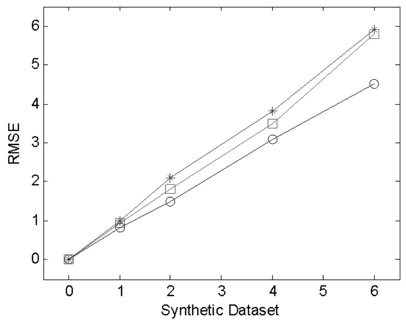


Fig. 4.6 Variation of samples with Vertex in-degree settings Legend: X-axis: vertex in-degree settings, Y-axis: RMSE Plots: ○ – 50 samples, □ – 25 samples and *–10 samples

4.4.5 Effect of noise

Noise plays a major role during inferring the structure of gene regulatory network. While generating synthetic dataset described earlier, a Poisson distribution was used to vary the amplitude of the gene expression. The proposed approach for model reconstruction is based on the structure of the sub-model which usually contains around 10 nodes. For the studies performed here, for the sub-model, we keep the numbers of nodes = 10, edges = 20, parents/node = 3, delay = 2 and samples = 25. The noise level is increased from 1% to 40% on the average amplitude (expression) of the genes by varying the mean of the Poisson distribution function as shown in the Fig. 4.7.

$$\text{Noise measure } (\text{SNR}) = \log_{10} \frac{\text{observed}}{e} \tag{4.5}$$

Where e is the deviation (of structure and parameters) between the observed network and actual network

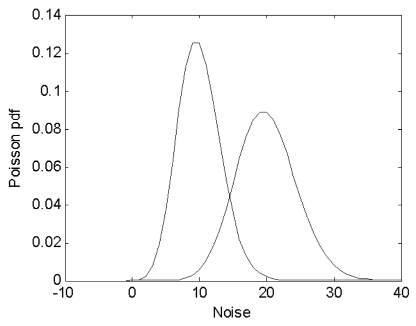


Fig. 4.7 Noise Variation of Poisson probability distributions with noise inclusion of 10%(steep curve) and 20% (rounded curve)

In the experiments carried out, noise is added in steps of 1% for noise levels from 1% till 5%. For higher noise levels, namely 15%, 25%, 30% and 40%, the step size is made larger. Again, the number of runs for each noisy data was maintained at 20. Fig. 4.8 plot shows the min and max values of SNR for the 20 runs carried out for each noise level. As expected, the measure of signal to noise ratio of Eqn. 4.5

indicates a decrease in performance with the increase in noise level. However, we observe that the decrease in performance with the increase in noise is not constant. The performance deterioration is relatively less at higher noise levels compared to the condition when the noise is less. This indicates some robustness of the method against noise variations.

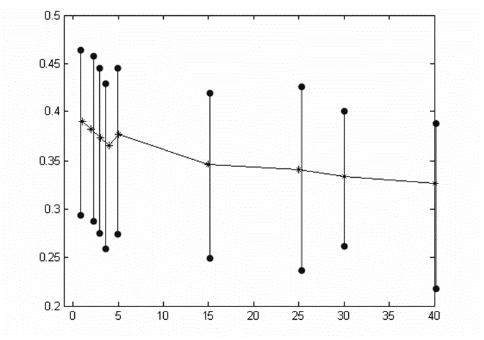


Fig. 4.8 Average SNR at each noise % value. Legend: X-axis: Noise in %, Y-Axis: *SNR*

4.4.6 Effect of network size and topology

To evaluate the method for structural variation, two sets of networks are generated. The first set consists of five random networks with 100 nodes and 200 edges and the other set has 200 nodes and 400 edges. Table 4.6 below shows the five 100 node and four (2-5) 200 node networks configurations used.

Table 4.6 Simulation Settings: Synthetic Network Structure

Network	Path lengths	% arcs removed	% arcs added	5% Noise
1	3	No	No	Yes
2	3	10%	5%	No
3	4	5%	10%	No
4	4	10%	5%	Yes
5	5	5%	5%	Yes

The threshold of ROC curve is chosen to be the significance measure (Bonferroni-corrected p-value) which was presented in Section 3.4 (step vii). We emphasize that this significance measure is not arbitrarily chosen, but the choice was based on few initial simulation results.

All values of the threshold with increment at the rate of 1% are used to plot the ROC curve in each of the experiments. The ROC curves are plotted for the two sets of randomly generated artificial networks [106]. For computing the significance of proposed method, all simulations are repeated 3 times. It can be clearly seen that all the ROC curves lie above the diagonal which is the ROC curve of a random model. In Fig. 4.9, the ROC curves for reconstruction of random and scale free networks of 100 genes are shown. A powerless method has an ROC score close to 0.5. The mean of the ROC curves gives a ROC score close to 0.8 which shows that the model is robust with respect to size of network and irrespective of the topology used. The 200 gene network with a scale-free topology yields AUC (see Fig. 4.10) that is comparable to the 100 gene network. The errors (i.e. TP/FP/TN/FN) have been computed by taking the final solution and comparing its structural difference with the target network. The results obtained for a 100-gene network are similar to those for larger networks. Thus, we observe that the performance of the model does not deteriorate with increase in number of genes.

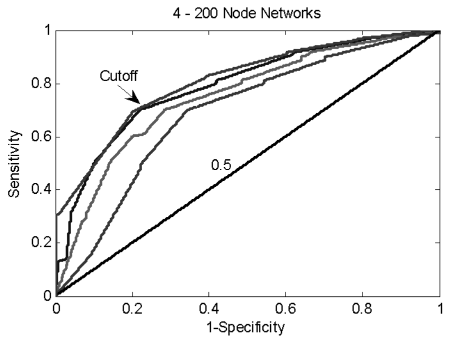


Fig. 4.9 ROC curves for 200 gene network. Legend, X axis: (1-specificity) and Y axis: sensitivity

The average TP/FP for all such networks for the 10 runs is calculated which best specifies error for the network generated with the highest fitness score among all 10 runs. For gene networks, as adjacency matrix is generally sparse, the ROC curve is susceptible to the high number of false positives as these edges might be getting included to improve the final score. Fixing a cut-off threshold only alters the tail of the ROC curves.

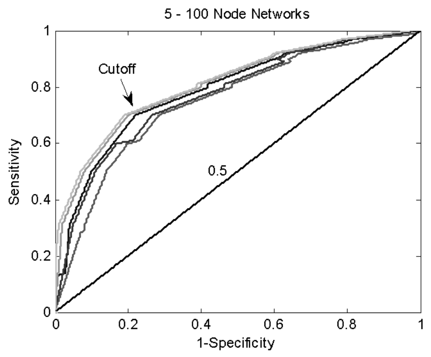


Fig. 4.10 ROC Curves for 100 node network. X axis: is (1-specificity) and Y axis: sensitivity

4.4.7 Comparison

Using the synthetic data, the proposed method is next compared to two known existing methods namely, the Graphical Gaussian model (GGM) proposed by Toh *et al.* [54] and the Bayesian network method proposed by Friedman *et al.* [17]. For comparison, three sets of synthetic data are generated. As in previous experiments, we chose the sum of FP rate and FN rate as the measure of error. The three ROC curves shown in Fig. 4.11 provide the comparison of reconstruction ability for a small sample size (compared to the number of network genes). For a small sample size, both the proposed approach and the GGM approach are able to recover the true network topology with high accuracy. However, for higher cut off values, the GGM approach degrades in quality compared to our approach. For the BN method, although the AUC is within the acceptable range, it has considerable lower accuracy.

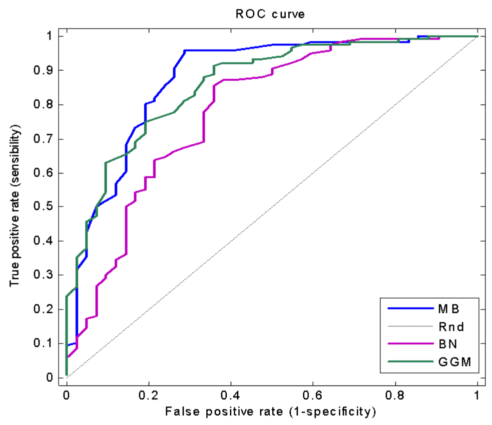


Fig. 4.11 Comparisons based on 100 gene network with 40 samples and without noise.

The three ROC curves shown in Fig. 4.12 compare the robustness of approaches towards noise. The well known GGM method is found to reconstruct the network with moderate accuracy compared to the proposed method due to its underestimating the true noise. It appears that GGM method requires more than second-order dependence (partial correlation) for elucidation of the relationship between the genes. However, the proposed method effectively copes with the noise variations at different rates as shown in Fig. 4.7.

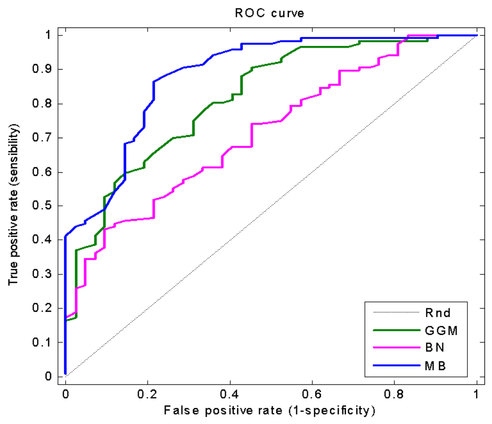


Fig. 4.12 Comparison based on 100 gene network with 40 samples with 10% noise

The BN method is not completely accurate and showed a moderate reconstruction in the presence of noise. Robustness against the scale of network is presented in Fig. 4.13 based on a network of the order of 200 genes. The BN method tends to provide very poor reconstruction. We also observe the GGM method's failure to capture many interactions as the number of genes in the network increases, because the order of partial correlations used becomes very complex and the accuracy of the dependencies identified becomes very less. With a threshold value of 0.62, the FPs produced are exceeding half the total number of edges and the reconstruction by GGM becomes less acceptable. The proposed method again shows a superior capability in capturing gene interactions and not only copes with small samples and noise but also shows suitability for larger networks. The approach can predict regulatory networks with significantly improved accuracy and reduced computational time compared with the two existing approaches.

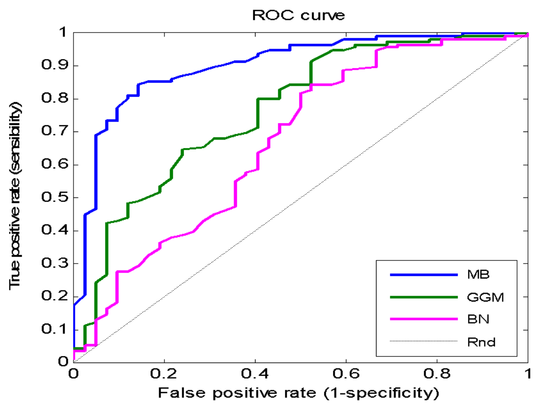


Fig. 4.13 Comparisons with a large 500 gene network, 100 samples and without noise

4.5 Discussion

With any GRN reconstruction techniques, the validation of results and predicted interactions using the real microarray dataset for inferring the underlying real network is restricted due to the limited availability of data for validations [17]. Hence, the synthetic networks which capture crucial elements of transcriptional regulation have been used for validating the GRN reconstruction of complex biological networks generate networks. Although simple, they thus provide realistic test beds for our new algorithm and enable us to conduct experiments by parametric variations for assessing effects of small perturbations of gene expressions. From these investigations, we were able to clearly establish that the proposed method is not only robust but is also able to discover connections with relatively low false discovery rate.

All the accurately inferred interactions and the predicted edges having causal influence value, direction, sign and delay are significant since the synthetic data used for testing the model simulates a real microarray experiment by including for example, noise, delays etc. The edges are identified with high confidence even with the test being carried out for a small proportion of all interactions O(*nxn*). The use of existing knowledge in the Markov blanket algorithm reduces the Markov blanket condition set which also results in low false discovery rate.

While complex GRN models which fit micro-arrays can represent a wide range of relations, e.g. thresholds or combinatorial interactions, there is always a risk of over-fitting with small sample sizes. For example, Graphical Gaussian models (GGM) [54] suffer from unreliable estimates of the full partial correlation coefficients if the number of samples is relatively small in comparison with the number of genes because the dependency between two genes is controlled by all other genes (full partial correlation coefficients). Linear causal models provide a vital middle path between quantitative models requiring many observations to fit parameters, and typical Bayesian networks which generally rely on discrete variables (excluding certain Bayesian methods [17] involving continuous responses). The assumption of linearity also enables many techniques [7, 17, 107] in accurately finding independence in gene expression data. Due to simplifications, we also eliminate statistically unreliable and computationally costly search for conditional independence in large subsets since only a subset of conditional independence models need be considered to enable us to study the effect of the sample size, number of parents (or children/spouses) per node, noise level, accuracy in estimating delay and the level of conditional independencies. The experiments carried out have also been able to establish this lack of benefit from increased order of independence testing. Although the causal model is a simplification of real biological networks, it captures complex interactions including elements of transcriptional regulation accurately and efficiently thus showing it to be suitable for reconstructing GRN.

Compared to the existing Bayesian and GGM approaches, the proposed method based on causal relationships is simple, reliable and flexible for scoring. It differs from other existing approaches both in the application of novel scoring technique as well as the learning algorithm and provides the flexibility to perform on-the-fly modifications to improve the structural accuracy with less computational effort. The qualitative scoring method is effective since higher-order statistical tests become unreliable due to the usually small sample size in functional genomics. It shows promise even with limited number of samples because the approach inherently requires a small number of parameters to represent relationships between genes. This is also confirmed from studies which involved small to medium sample sizes, in which the causal models proved to be better estimators compared to the results from GGM modeling. In contrast to de la Fuente approach [46, 54, 67, 76, 93, 108], in which the undirected graph (UDG) is inferred first by the brute-force search method, the proposed GA estimates the direction and sign of regulations after the UDG.

4.6 Summary

The network generator system presented in this chapter generates synthetic GRN datasets which are used for validation of the methods and techniques proposed in this thesis. Investigations using the network generator show the significance of the application of system for synthetic data generation. The proposed system can generate four different network topologies, namely scale free, small world, random and hierarchical. Further, the generated synthetic network is made realistic by incorporating complex network characteristics such as transmission delays, biological and experimental noise. These datasets are generated for evaluation of methodologies based on these synthetic datasets. The system will help other similar methods to computationally determine the robustness and also establish comparisons between the methods. In comparison to other existing methods, the proposed system is useful in carrying out rigorous studies about the GRN methods. The generated synthetic but realistic datasets was applied in validating the robustness of the proposed method for GRN reconstruction by varying topology, time-series size, delay effect, noise, vertex degree, and presence of hidden nodes. The experiment results show that the proposed approach has excellent inferential power and also low specificity even in the presence of noise.

Apart from mathematical representation of GRN and the synthetic dataset generation, another important aspect of network reconstruction is the computational time which depends not just on the size of the dataset but also on the design of the suitable search algorithm. In next chapter, we present two novel search techniques, namely guided GA and FOMBGA techniques, to improve the performance of simple GA which was applied in this chapter.

Chapter 5

5. Guided Genetic Algorithm

5.1 Introduction

A Markov blanket based approach for constructing a Bayesian gene regulatory network inference and its application to noisy high dimensional microarray data was presented. As explained earlier, the structure search, in general, can be stated as, *"Given a data set, a score metric, and a set of possible structures, find the network structure with maximal score"*. As the number of genes *n* for GRN inference is of the order of thousands in a gene expression data, the number of possible structures explodes to a large astronomical value. For this reason, search for biologically significant investigations was usually limited to small subsets of selected genes, i.e. on a small scale mainly to as the search strategy to find the best candidate network structure were not very effective. Since exhaustive search in the structure space can be impractical and exact inference with BNs is known to be NP-hard [81], stochastic approximation to the search for high-scoring network structure is often necessary to obtain results. Many search strategies are available for learning general Bayesian networks in various domains including GRN [30, 34, 59, 64, 69, 77, 109, 110]. However, attempts to obtain an optimal skeletal structure (the essential edges in a network) that accurately reproduces the continuous time-course microarray data have mostly remained unsuccessful.

In Chapter 3, we presented a simple genetic algorithm for the structure search. The chosen approach, i.e. GA, performs a global search and can simultaneously estimate many (causal) sub-model parameters [62]. Although GA has been successfully applied in many cases, its implementation is often challenging. For example, designing an effective mutation operator in order to ensure a correct neighborhood search becomes difficult because most operators usually search in the local neighborhood and do not take into account the global neighborhood information. When GA is applied to learning static Bayesian networks, the application of various operators is required. Although randomness prevalent in GA provides it with the ability to escape local maxima [77], in practice, an appropriate tuning of the GA by incorporating domain knowledge is often necessary to enhance its performance. For example, application of a guided mutation operation [111] could successfully generate new individuals which were observed to be close to the best solution. In [112], Larranaga *et al.*, used a genetic algorithm to evolve BN structures and a 'repair' operator is applied to remove cycles because BNs are acyclic. Lam *et al.* [113] used the Minimum description length (MDL) principle with a GA to evolve BNs with the help of three operators: freeze, defrost and a Knowledge Guided Mutation (KGM) to improve the scalability and speed of convergence as well as remove any cycles. The freeze and defrost operators are used to engage and disengage the KGM operator. Using a KGM involves generating a list of all single edges, ordered on their description length (DL). This list guided the mutation within GA by adding edges which appear in the higher ranks of the list and removing edges which appear in the lower ranks. Sahami [114] used the mutual information between a node and its parents as an operator to select networks.

In this chapter, we propose to enhance the performance of simple GA presented in the previous chapter. In Section 5.2 we briefly study the simple GA scheme. We investigate a novel technique to guide the simple GA by including the knowledge of execution history for making the search effective in finding a high-scoring network structure. The knowledge acquisition process provides information to make decisions while performing guided crossover and mutation operations. As the proposed approach can have ambiguity as to whether or not an edge can be added or deleted, a technique is further applied where the randomness in carrying out an operation is varied according to the level of ambiguity in the knowledge acquired for guidance. These details are given in the Section 5.3. The advantage of the proposed strategy is that it offers the possibility of determining more efficient structure learning with necessary tradeoffs between the reduction in false positives, diversity, random/guided operations, multiple paths and best solution. The process is based on a ranking schema and standard Gaussian function which is also subsequently explained (see Section 5.3). Subsequently, in Section 5.4 we further refine the algorithm to a probabilistic model based GA called as *Frequently occurring Markov Blanket Genetic Algorithm* or simply *FOMBGA*. Finally Section 5.5 gives the summary of the chapter.

The simple GA on which the proposed guided GA and the FOMBGA are based is explained next.

5.2 Simple Genetic Algorithm

Finding an optimal causal structure for gene regulatory network using GA can have several problems. Some of these are given below.

- *Lack of data problem*: It is well known that BN learning algorithms perform better with larger quantities of data. However, with microarrays, the quantity of data is often limited to a few samples resulting in the network having false positives. Yu *et al.* [115] infer DBN models of gene expression networks where they use influence score to help improve the relevance of edges from recovered networks, and thus reduce false positive (extraneous) edges. Although reduction of false positives is important, increasing true positives is equally important.

- *Diversity problem*: Maintaining high diversity is particularly important for optimization of NP-hard problems because high diversity increases the probability of relocating the peak after a change in the landscape because the population covers a larger part of the search space. Diversity is undoubtedly closely related to the performance of evolutionary algorithms, especially when attempts are made to overcome the problems of escaping local optima. As our fitness works on sub-networks, the diversification of the search process is important in order to escape from local optima and the simplest mechanism to diversify the search is to consider false positives during the process. In order to make a trade off between the reduction in false positives and diversity, the average quality of the population can be made adaptive by alternating between guided and random genetic operations using diversity measures. Diversity measures have been traditionally used to analyze the performance of GA's. Although, diversity measures such as Hamming distance [116, 117] have been used in the literature for controlling the EA's, these are not suitable for our problem because we are dealing with networks. We therefore propose a novel diversity measure where a skeleton network is obtained for each generation and the average deviation of the individuals from the skeleton forms the diversity measure.

- *Dead Node problem:* The occurrence of dead nodes, defined as a node which has no connectivity to the network, is also important issue to be considered. The occurrence of dead nodes can be attributed to randomness in the deletion of edges during the evolution process. In this case, we have to look at the connectivity, or topology, of the network. Each node has the potential to be in contact with every other node in the network either directly or indirectly. If one edge is deleted from the network during a genetic operation, and if the network is still connected, then the deleted edge neither plays a vital role nor has little effect in the network. If the deleted edge leads to a dead node, the edge may have a greater effect in the network. A path in network is defined as a sequence of edges or nodes without any repetition of nodes. Our proposed guided strategy overcomes this problem by checking alternative paths before deleting an edge.

- *Ambiguity problem:* Finally, the knowledge acquired for performing a guided operation can be ambiguous. For example, the KGM operator uses a rank list of edges. The topmost and bottom most edges in the list have absolutely no ambiguity, while rest of the edges have some percentage of ambiguity in providing guidance to either adding or deleting an edge leading the search to get trapped in a local optima.

In the previous chapter, due to the sparse nature of GRNs [71, 93, 118], we reduced a bigger network into smaller networks, called sub-networks. The sub-networks, as mentioned, are nothing but the Markov Blanket (MB) networks of genes in the network. A fitness score is assigned to each of these MB networks. This is done by performing a series of Conditional Independence (CI) tests in order to detect whether or not connections are direct or indirect according (see Section 3.4) and thereby assigning a score for each node. Further, direction and sign of the edges are analyzed for each MB. The summation of the scores yields the fitness of the total network. The fitness equation, given in Eqn. 3.7, is repeated below for easy reference.

$$Fitness\ Score = \sum_i (w_1 * Score_{1_i} + w_2 * Score_{2_i} + w_3 * Score_{3_i}) \tag{5.1}$$

Here w_1, w_2 and w_3 are fractional weights assigned to each score such that $w_1+w_2+w_3=1$. This weighting is incorporated to adjust the growth rate of the individual scores as a function of the number of edges. Score_1 ascertains the structure of the network given the structure, direction and sign of regulation. Score_2 and Score_3 ascertain the direction and sign respectively. Though for any given graph there is only one possible direction and sign for each edge, it does not hold good for our model in selection of best individual among the population as $Scores_2$ and $Score_3$ give additional flexibility towards the search for optimal solution.

The candidate network structures are constructed with each gene having a set of *M* parents, where the value of *M* ranges between 2 and 7 on an average as specified by Hertz *et al.* [16, 119]. The feedback loops and auto-regulation (gene regulating themselves) connections are not constructed in the putative structure since their presence have been ignored for the sake of simplicity. The *nxn* chromosome matrix, encodes the network structure with each row corresponding to a tail of a edge and each column corresponding to the head. The chromosome encodes the presence of a directed edge between two genes, its direction and sign of regulation using values {1, 0,-1}, where 1 indicates positive regulation, -1 indicates negative regulation and 0 indicated no regulation as shown in Eqn. 5.2. If for example, there is a edge between gene *X* to gene *Y*, with a negative sign of regulation, the chromosome encodes Chromosome $(X, Y) = -1$.

$$m_{ij} = \left\{ \begin{array}{ll} 1, & \text{i} \rightarrow \text{j regulation is positive} \\ 0, & \text{otherwise} \\ -1, & \text{i} \rightarrow \text{j regulation is negative} \end{array} \right\} \tag{5.2}$$

A simple genetic algorithm (GA), applied to explore this structure space, begins with a sample population of randomly selected network structures and their fitness calculated. Iteratively, random crossovers and mutations of networks within a population are performed and the best fitting individuals of the population are kept for future generations. As generations pass, the population evolves leaving the fitter structures while those performing poorly become extinct. Due to the stochastic nature of the GA, it is repeated for a number of times and the resulting network structures are combined to reconstruct the final gene network.

The details of the overall method are available in Section 3.3. However, the results of the method are given below. These results will be used to compare with our proposed GGA and FOMBGA method.

5.2.1 Experiments and Results – Simple Genetic Algorithm

To investigate the impact of simple GA in the causal modeling process, we studied the well known *Saccharomyces cerevisiae* microarray dataset [120]. In DeRisi's experiment [120] with this dataset, DNA microarrays containing almost all genes of *S. cerevisiae* were used to monitor the temporal changes of gene expression levels accompanying the metabolic shift from fermentation to respiration upon glucose exhaustion. Data consist of 7 time point profiles of logarithmic expression level ratios. The full data set is available at the Gene Expression Omnibus website [121] (Omnibus ID: GSE28; PMID: 9351177). The UNF_VALUE column in the dataset contains the expression values of each of the genes at all seven time points measured. The data preparation step involves creation of a table with rows corresponding to genes and columns corresponding to time points and UNF_VALUE as the data.

Since regulatory and signal relationships are currently not sufficiently known and because of the size limitation imposed by the relatively small number of time points, we restricted our analysis to 77 genes involved in the major metabolic energy pathways given in [120] (i.e. glycolysis, gluconeogenesis, pentose phosphate pathway, TCA cycle, glyoxylate shunt and fermentation). The experiment involved transcriptions of the glycolytic genes for glucose metabolism and the response of the cells to the stress of nutritional starvation toward time point 6. Amongst others, the

77 genes include ACO1, ACS1, FRDS1, GPM3, ICL1, IDP2, MLS1, PCK1, PDA1, PYC1, PYC2, YOR283w, YOR297c, FBP1, CIT2, IDP2, YJL045w, PCK1, HXK1, HXK2, GLK1, PGI1, PFK1, PFK2, FBA1, TPI1, TDH1, TDH2, TDH3, PGK1, GPM1, ENO1, ENO2, PYK1, GCR1, GCR2, RAP1, MSN2, MSN4, GCR3, MSN1, MSN5, etc....

The raw data used in this experiment was transformed by taking the log of the expression levels. Table 5.1 shows ranges of the GA parameters used in this experiment. The mutation probability was evenly distributed over the four mutation techniques. The genetic algorithm was run 5 times. Fig. 5.1 shows the increase in the best score of the networks in the population as the population evolves for the first GA run. Fig. 5.2 shows the average number of edges over the generations during the first GA run. The GA is stopped when there is no improvement in the average score, which on an average occurred at the 45th generation.

Table 5.1 Genetic Algorithm Parameter Settings

Parameter	**Value**
Crossover probability	0.1
Mutation probability	0.8 (evenly distributed over 4 types of mutation)
Population Size	200 – 600
Iterations	50 - 100 / based on improvement

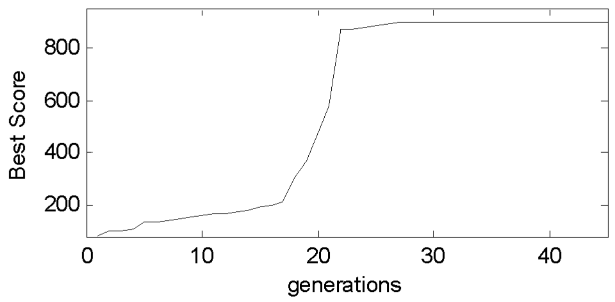


Fig. 5.1 The plot shows the improvement in the best scores. There is a large improvement in score between generations 18 and 23. The GA was terminated when there was no improvement in the average score

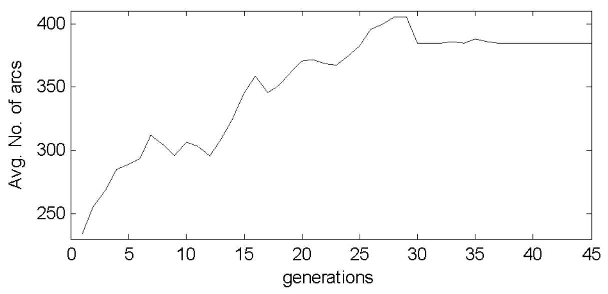


Fig. 5.2 Variation of average of edges with generation. The average number of arcs increase up to generation 30 beyond which there is no further increase

There were 385 connections (edges) inferred on an average over all GA runs. As diagrammatic representation of the entire network with large number of edges is difficult to visualize; in Fig. 5.3, we plot only a part of the genetic network compiled from the results obtained from the networks with highest scores in the population of the last GA generation for each of the five GA runs. It was also observed that in a single GA run, the highest scoring network differs from other networks by only a few edges. Hence we quantify the significance of an edge based on its number of occurrences within each GA run as well as over all five GA runs. The edges with high frequency are thus seen to comprise the skeleton of the reconstructed network. There were a total of about 178 highly significant connections compiled from the results obtained from repeated GA runs.

In Fig. 5.3, the reconstruction of the edges in relation to genes involved in the Glycolysis/Gluconeogenesis pathways, for example, GCR2→HXK1 and RAP1→PGK1 are consistent with experimental findings [122]. In relation to genes involved in the Glyoxylate Shunt, repression PCK1 → IDP2 is also consistent with the literature. Many highly significant edges such as ENO2→ GCR2, PFK1→GCR1, etc…, which were identified by computational methods [123] and have also been verified. Other remaining edges could not be verified as there is little evidence in literature supporting these remaining edges.

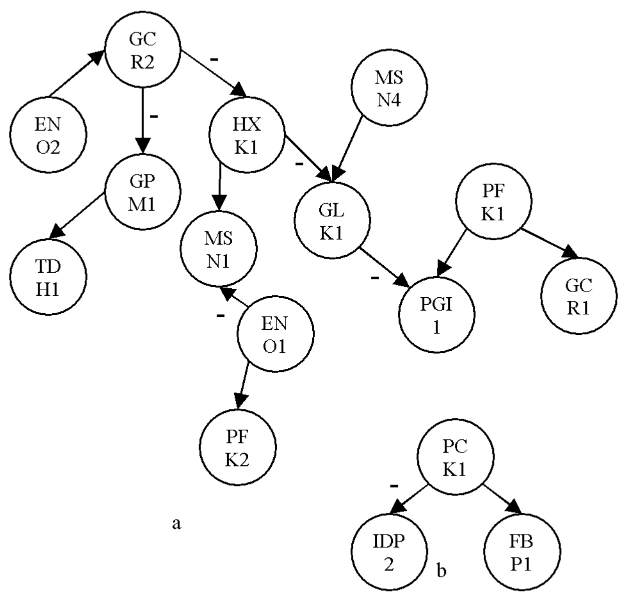


Fig. 5.3 A section of the reconstructed genetic network of S. cerevisiae during metabolic shift from fermentation to respiration upon glucose exhaustion. (a) Reconstructed Glycolysis / Gluconeogenesis & Pentose Phosphate Pathway (b) glyoxylate shunt

In Fig. 5.3, the reconstruction of the edges in relation to genes involved in the Glycolysis/Gluconeogenesis pathways, for example, GCR2→HXK1 and RAP1→PGK1 are consistent with experimental findings [122]. In relation to genes involved in the Glyoxylate Shunt, repression PCK1 → IDP2 is also consistent with the literature. Many highly significant edges such as ENO2→ GCR2, PFK1→GCR1, etc…, which were identified by computational methods [123] and have also been verified. Other remaining edges could not be verified as there is little evidence in literature supporting these remaining edges.

Although a simple GA above is seen to be effective in finding useful networks, it does have some limitations, such as

1. The random creation of initial populations and the randomness of subsequent exploration.
2. The tendency to focus too closely on a single, high-quality solution.

3. The tendency to carry out redundant computation, as it re-evaluates large populations after making small changes to the individuals.

The next Section 5.3 describes the Guided GA method to eliminate the above mentioned limitations.

5.3 Guided GA

In this section, instead of the simple genetic algorithm, a guided GA (GGA) is used to perform a heuristic search through the space of gene regulatory networks. The fitness used is the same as given by Eqn. 5.1. The proposed algorithm works as follows.

i. Initialization and Fitness Function

The gene regulatory network consists of a set of nodes, which represent genes; and a set of directed edges between nodes, which describe dependencies involved. The nodes and edges form a Directed Acyclic Graph (DAG). To represent a gene network as a GA individual, an edge matrix or adjacency matrix is needed. The set of network structures characterized by n variables can be represented by an $n \times n$ connectivity matrix M. Each element in M, m_{ij} where $i, j \in \{1, 2, \ldots, n\}$, represents the edge between two nodes such that it satisfies Eqn. 5.2. It can be noted that values of m_{ij} for $i=j$ can be ignored during the search process because it presents an edge between a node and itself.

With the representation of individuals worked out, we next devise the generation of the initial population. Amongst several ways for generation, in this approach, we generate individual matrices randomly. For each individual, a node and set of parent nodes are selected (with an average number of parents per node as 4 and a variance of 3) based on a random Poisson distribution and edges are created accordingly. The simple approach can create cyclic sub networks. As these are illegal DAGs, an algorithm is also proposed to remove these cycles.

Now that the initial population is generated, a fitness function is defined to determine the fitness of the individual. We use the fitness function that is specified in Eqn. 5.1. The network is decomposed into nodes and their corresponding Markov blankets. The Markov blanket, as explained in Section 3.3.3, comprises of the parents, the children, and the parents of the children of the node of interest. The Markov blanket of a node can be easily identified from the network topology. These putative sub-networks are evaluated for the fitness of their structure, and also the fitness of the direction and the sign of their regulation. Hence, a fitness score is assigned for each node and simple summation of scores on all nodes in the dataset will determine the fitness of the entire DAG, i.e. an individual.

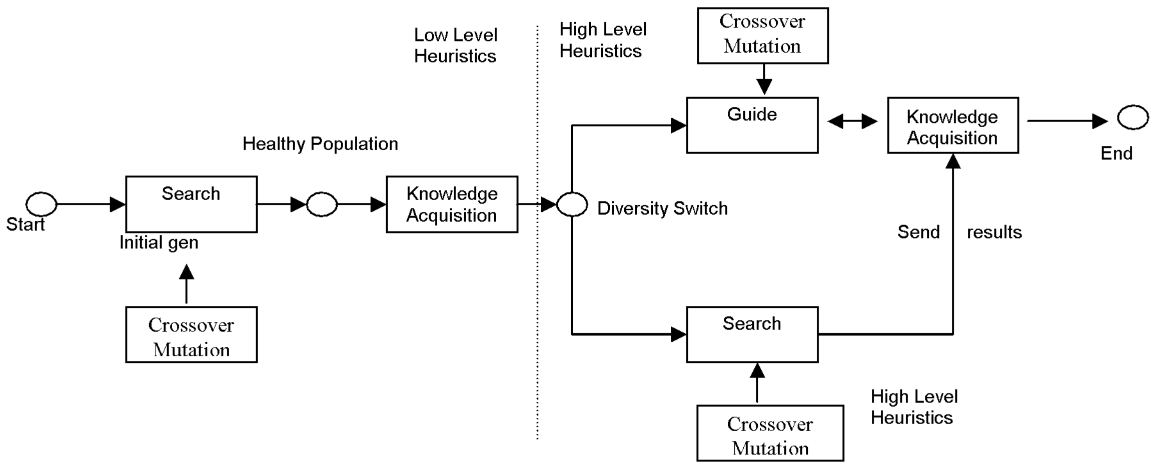


Fig. 5.4 Novel Guided GA strategy

With the initial population is evaluated, the evolution (the iteration process involving crossover and mutation) is carried out. This is explained next with the aid of Fig. 5.4.

ii. Low level Search heuristics

The low level heuristics refer to random crossover and random mutation operations performed during the initial stages of the GA run. The crossover operation between two networks involves taking two random individuals from the population, randomly selecting two gene edges and then swapping these between the pair of networks. Mutation is applied on individual networks by randomly adding a new edge or deleting an existing edge.

The low level heuristics (see Chapter 3 Fig. 3.2), when performed later in the guided algorithm, acts as heuristics needed to increase the diversity of the population because of their randomness.

iii. Knowledge acquisition process

This part of the algorithm is responsible for collecting and storing information that is intended to provide knowledge to guide the search operation. For a n-node gene network, at any i^{th} generation $G(i)$, and with a population size of N, fitness of $n \times N$ nodes and their corresponding Markov blanket structures are to be evaluated. Hence, there are $1 \times N$ set of Markov blanket structures for each node at a given generation. These are rank ordered over the period of execution to form a rank list of Markov Blankets for each node. The methodology for ranking is done by a simple algorithm illustrated in Table 5.2. The algorithm accepts the population of individuals as input and produces an array of ranks and corresponding MB structures as output. As the generations are incremented, both the ranks and MB structure arrays evolve in such a way that information needed to perform a guided operation is latest up to the present generation.

Though the execution is altered between low level heuristics and guided heuristics (explained later), the knowledge acquisition process is kept informed of the changes and the rank tables are constantly updated. This is shown in Fig.5.4 as “Send results”.

Table 5.2 Algorithm to Rank MB’s

```
Algorithm Rank_MB
Input: Population
Output: rank(), MBstructures()
Curr_Rank = 1
N = num_nodes
m = Population_size
While N ~= 0
    For i = 1:m  % Add new rank to table
        If MBi is nondominated
            rank(MBi) = Curr Rank
        End If
    End For
    For i = 1:m
        if rank(MBi) = Curr Rank
            Remove MBi from array rank()
            N = N-1
        End If
    End For
    Curr_Rank = Curr_Rank + 1
    m = N
End While
End of algorithm
```

iv. Diversity Switch

The action of the diversity switch is based on a diversity measure to switch between low level heuristics for improving diversity and guided heuristics for improving the best solution. The diversity measure is defined as follows:

$$D_m(G(i)) = \frac{1}{N}\sum_{1}^{N}\sqrt{(Indv - skel)^2} \tag{5.3}$$

where *N* is the population size, *Indv* is the individual adjacency matrix and *skel* is the adjacency matrix of the skeleton network for generation *G(i)*. The diversity measure involves estimating the skeleton network. A skeleton network is defined as a network containing edges that are present significantly in the population of a given generation. In other words, for a population of *N* individuals, an edge is part of its skeleton network if it occurs in *N/2* or more individuals.

The deviation of the skeleton network from an individual *(Indv-skel)* is modeled as a score, which is assigned 0 initially and incremented for every edge that is not in common or is extra and not present in the skeleton whilst being present in the individual. The average of this deviation is equal to the diversity measure. If the population loses diversity, the diversity measure approaches zero, as the individuals become similar to the skeleton network. The diversity score is normalized within a range 0 to 1, in order to set a threshold over diversity as a GA parameter. When the diversity measure drops back below this threshold value, the operators responsible for increasing diversity are used, and above the threshold, the guided operators which tend to decrease diversity are activated. The adaptation and guidance is suitable only at a later stage of the evolution and hence the diversity switch comes into action only after completing initial GA runs.

v. High level Search heuristics

The operators used in high level search heuristics are similar to the low level operators as explained in Fig. 3.2. The main difference is that, the high level operators perform a series of algorithm steps before addition or deletion of an edge during crossover and mutation as detailed in Fig. 5.5 instead of the randomness.

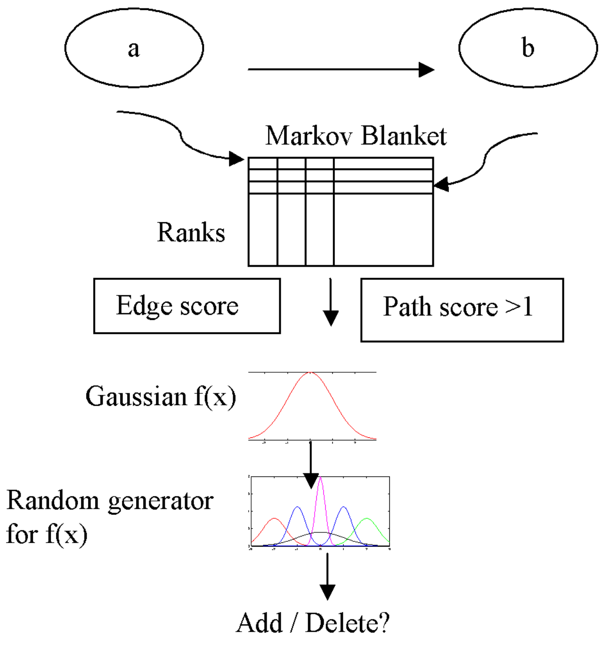


Fig. 5.5 High level heuristics

Heuristic algorithm steps shown in Fig. 5.5 are as follows:

Select two nodes, say *'a'* and *'b'*. Once the two nodes are selected for an edge addition or deletion during crossover or mutation, the list (array) of nodes Markov blankets are extracted along with their ranks. This information is used in the computation of the edge score and path score as explained below.

- *Edge Score*: The top half of the rank ordered MBs is considered as best while the MBs in lower half are considered worst. For an edge which is a member of an MB in the top half, we add 1 to the edge score and for the edge which is a member of lower half, we subtract 1. The resulting score is the edge score assigned to a variable x (which is used in the subsequent steps).
- *Path score*: From the two individuals (DAG) from whom nodes *'a'* and *'b'* are selected for crossover or mutation, the topology of the two networks are searched to determine the number of direct and indirect paths between them. The number of path equals the path score. If the path score is greater than 1, then continue, otherwise add the edge between *'a'* and *'b'* and exit this

algorithm without proceeding to the next step (as otherwise this could lead to a dead node[66]).

The resulting edge score x is normalized and given as a input to Gaussian function *f(x)*.The Gaussian function is defined as

$$f(x) = ae^{-(x-b)^2/c^2} \tag{5.4}$$

Where b is the mean value which is 0 and c is the deviation on either side of the bell-shape curve shown in Fig. 5.6. The returned value lies in the range 0-1 which represents the amount of ambiguity in the knowledge.

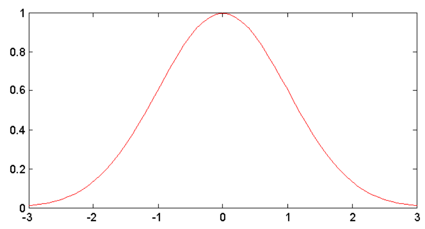


Fig. 5.6 Gaussian function

A random normal variable generates a random number according to a normal distribution

X=N (mean, variance)

The density function for 0 mean and 1 variance is given as

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \tag{5.5}$$

The density function for different mean and variance are plotted in Fig. 5.7. The random function's mean is shifted according to amount of ambiguity determined by the Gaussian function. A detailed explanation of working of this algorithm is given below.

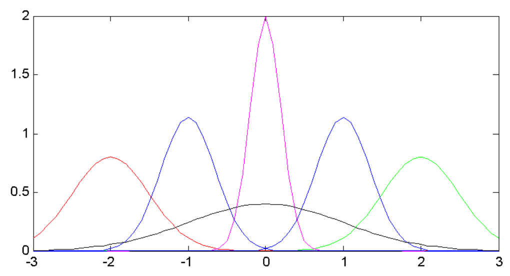


Fig. 5.7 Probability Distribution Functions of Normal Random variable

v. Detailed Explanation of the Algorithm

The edge score mentioned in the algorithm above is a counter that increments the score every time the edge is encountered in the first half (best) of individuals and penalized every time the edge is encountered in the next half (worst). This score, when normalized (with maximum and minimum scores that can reach $-N$ and $+N$ respectively), will lie in the range of [-3, 3] with 0 mean and unit standard deviation. In Eqn. 5.4, mean is set to 0 (i.e. $b = 0$) and standard deviation is set 1 (i.e. $c = 1$) and the value of 'a' is adjusted to produce a bell curve as shown in Fig. 5.7. The normalized edge score (i.e. x) is given to Gaussian function $f(x)$ of Eqn. 5.4. The value of $f(x)$ will lie in the range [0, 1]. Here, for $f(x) = 1$, the knowledge is 100% ambiguous as the edge score is zero and the decision to add or delete the edge is completely a random choice. The edge score of zero can either mean the edge has occurred equal number of times in the best half as well as in the worst half or the edge never occurred at all. The path score, which works on the connectivity of the network, will result in a 0 for the case where the edge never occurred in both individuals involved in either the crossover or mutation, thus preventing a completely random decision leading to dead nodes and ensuring complete connectivity of the network. Similarly, for $f(x) = 0$, the knowledge is 100% unambiguous and the decision as to add or delete the edge is known for certain and there is no randomness involved. This is because the Gaussian function returns zero for values of x at the edges of the bell. For, $f(x) = (0,1)$, exclusive of 0 and 1, the knowledge is partially ambiguous. In this situation, a standard normal random variable generator is implemented. This random variable is restricted to generate non-zero random numbers within the range (-3, 3). The probability density function of the random variable (given by Eqn. 5.5) for 0 mean and unit standard deviation gives equally likely chances for the generated random number to lie on either sides of the bell curve. This is suitable for a 100% random decision (i.e. $f(x) = 1$) because, for the value of random number $r < 0$, the edge can be deleted and for $r > 0$, the edge can be added. For a 100% unambiguous decision, the mean of the random generator can be placed at +3 or -3 based on the (+/-) sign of the edge score, and 0 standard deviation. For the rest of the values of $f(x)$, the mean and variance are varied appropriately to make a partially ambiguous decision as to whether or not add or delete the edge. As for an example, $f(x) = 0.8$ and edge score $x > 0$, the partial ambiguousness can be defined as, there is 80% that edge be added and there is 20% chance that edge be deleted. So the random variable should have 80% chance to generate a random number $r>0$ and 20% chance to generate $r<0$. Here, the generated random number should lie between +2.4 (= 80/100*3) and -0.6 (= 20/100*(-3)), with mean placed at 0.9 (= (2.4-0.6)/2) and unit standard deviation. Similarly, the random number generator function is modified to take a partially ambiguous decision.

Following are several possible operations during crossover and mutation (Search/Guide mode):

1. Add-random (Search)
2. Remove-random (Search)
3. Add-swap-random (Search)
4. Add-remove-random (Search)
5. Add-Add-best(Guide)
6. Add-remove-worst(Guide)
7. Remove-add-best(Guide)
8. remove-remove-worst(Guide)
9. Add-swap-remove-worst (Guided crossover) and so on….

The best solution found through the overall search is presented as the final solution when the next stopping criterion is reached. The stopping criterions are:

1. *Maximum Limit*: When the iteration reaches a predefined maximum number of generations or
2. *No Improvement*: When the difference between the current fitness average and previous fitness total is less than 0.0001, the GA is stopped.

5.3.1 Experiments and Results

We now study the effectiveness of the guided genetic algorithm with the aid of both the synthetic data and real data. The underlining structures and parameters of the synthetic dataset are generated using the method discussed in Section 4.2. The details of the dataset used are presented next.

A. Synthetic Dataset

The expression data for the 40 gene network is generated by assuming that 6 genes have 3 regulatory inputs, 10 genes have 2 regulatory inputs, while the remaining genes have a single regulatory input. 33 interactions are designed to have a time delay of zero, 21 have a time delay of one and 9 have a time delay of two time points.

Table 5.3 GA parameter setting

Parameter	**Value**
Crossover probability	0.4
Elite Rate	0.1
Mutation probability	0.8 (evenly distributed over 4 types of mutation)
Population Size	150 – 200
Iterations	100
Diversity	0.7 (70%) (not applicable for ordinary GA)

Given this topology of the regulatory network, gene expression values are computed for each one of the 40 genes at 10 time points. The derivatives are computed by employing forward difference. The starting value for the bound for each gene is set to 1.0 and a bound increment value $\delta = 0.1$ is employed for computation. The synthetic network constituted 63 interactions with known regulatory weights and time delays associated with these interactions.

To evaluate the algorithm, we also executed the simple genetic algorithm which incorporates only the low level heuristic operators. Table 5.3 shows the parameter settings of the GA. The Fig. 5.8 shows the plot of the best fitness value for the whole evolution process of 100 generations for guided GA and simple GA. From Fig. 5.8, it can be seen that GA with guided strategy performs better than simple GA after the initial 0- 20 generations. For the first 20 generations, both the SGA and the GGA perform in a similar manner as GGA adaptation can only be performed in the later stages of the evolution. It can be noted that, between generations 40 and 50, the best value of fitness of the guided GA was lower than the ordinary GA and later followed by an increase in the fitness of guided GA. This increase is attributed to the diversity switch which indicates the previous generations between 40 and 50 had less diverse individuals. This is further witnessed in Fig. 5.9 which shows the diversity measures throughout the evolution from generations 20 to 100, and a diversity measure of 0.98 was recorded during generations 40 and 50.

As seen in nature, many genes of an organism stay inactive through its lifetime and are passed to further generations for later mutations or crossovers to activate, which is very well performed by the diversity switch and so this is seen as safeguarding diversity of the population. Both algorithms were restricted to runs of 100 generations. The guided GA converged at the top score at the 81st generation when all edges were recovered while the simple GA improvement had stopped prematurely at the 55th generation.

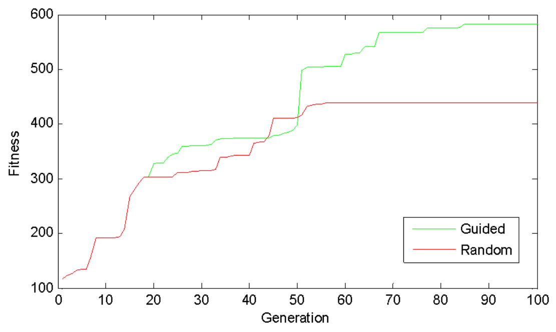


Fig. 5.8 Plot of best fitness values for SGA and GGA

From Fig. 5.9, it can be noted that diversity switch has alternated from low level heuristics to high level heuristics a total of 9 instances falling above 0.7 (threshold) during the entire evolution of 100 generations.

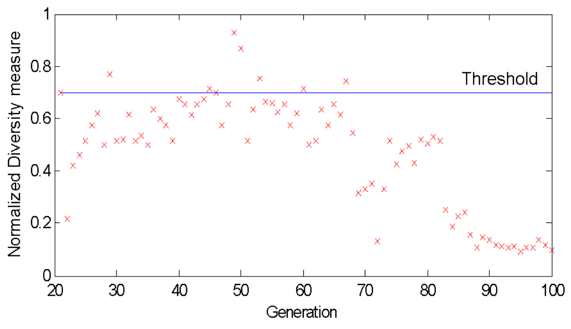


Fig. 5.9 Diversity measures from generations 20 to 100

Fig. 5.10 shows the histogram of the normalized fitness values for the individual Markov blanket structures at the end of 100 generations. Normalization is done for the range from 0 to 1 based on the maximum and minimum value of fitness calculated during the evolution. The fitness in the range of 0.8 to 1 are alone taken into consideration as they constitute the best scores and it can be seen that most of the Markov Blankets inferred lie in the range of 0.99 and 0.96. This shows most of the sub-networks have achieved their maximum fitness values.

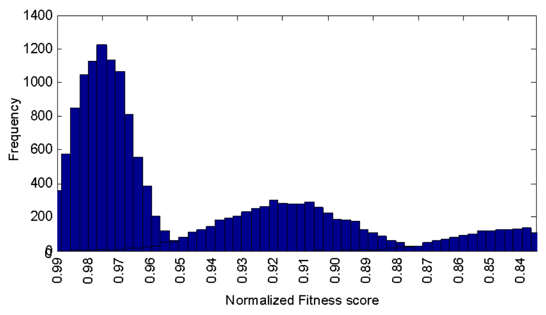


Fig. 5.10 Histogram of fitness values in the final generation

These results show that guided GA discovers causal GRN structures with a greater accuracy than the simple genetic algorithms. The accuracy improvement does not require any increase of search space. In all experiments carried out, only 150 to 200 individuals were considered during each of the 100 generations. Thus, 15,000 to 20,000 networks are totally searched to learn the causal structure. Considering the exhaustive search space of networks, the algorithm needs only a small percentage of the entire search space to learn the causal structure.

B. Yeast cell cycle dataset

After establishing the improvement in performance of the guided GA approach, we next applied this approach to the cell cycle expression data of Spellman *et al.* [49], containing 76 gene arrays of 6177 S. cerevisiae ORFs. The gene expression levels are taken as continuous values. We make use of all 76 samples from cdc15, alpha-factor and cdc28 datasets to determine the gene network structure. Since regulatory and signal relationships are currently not sufficiently known and because of the size limitation imposed by the relatively small number of time points, the analysis is restricted to 55 genes involved in the major cell cycle pathways given in [49]. The 55 genes include STE2, SWI5, CLN1, CDC6, YBL032W, STE6, STE3, AGA1, MFALPHA2, STE3, MFA1, STE6, FAR1, CLB1, CLB6, MCM1, SFF, CLB2, CLN2, SWI5, CLN1, AGA2, FAR1, TEC1, CDC6, SAG1, SST2, YBL032W, YDL165W, YIL010W, YPR066W, YOL129W, YEL071W, YOR045W, YCL064C, YIL066C, YMR226C, YCL021W, YGL139W, YKL141W, etc....

As is commonly done, the raw data used in this experiment was transformed by taking the log of the expression levels. The GA parameter settings are shown in Table 5.3. The GA was stopped at the 100th generation. The resulting Markov blanket substructures were of equal maximal score. There were a total of 281 interactions recovered from the final results. Out of the list of edges recovered are as follows.

STE2→STE6,STE3→SAG1,CLB6→CLB1,SST2→AGA1,YPL256C→YIL066C,MFALPHA2→STE3,MCM1→CLB1,CDC6→SWI5,MFA1→AGA2,FAR1→TEC1,STE6→FAR1,TUP1→MCM1.

The existence of above interactions is validated from the literature [49, 124].

Although performance of GGA is superior to SGA, its success is still limited since the search is based on the use of adjacency matrix in which the network interactions are simply recorded as binary elements (1= edge present and 0 = edge absent). Due to this, the context of each bit (i.e. occurrence of edges *in relation* to other edges or the nodes as is the case in a network structure) is igored while the search is being performed. Since both SGA and GGA are unable to treat the constraints explicitly and both these algorithms ignore the *context* in which the search (GRN) is being carried out, there is need for further enhancement of the search technique. To deal with this problem, we next propose a probabilistic model based GA which we will refer as the *Frequently Occurring Markov Blanket Genetic Algorithm* (FOMBGA) approach.

5.4 Frequently Occurring Markov Blanket Genetic Algorithm

We see that, although GAs are capable of efficient search, they are inherently limited due to the randomness and intuitive manner of parameter setting, namely crossover, mutation and selection rate. Further, when using GA for Bayesian structural optimization (search) problem, we have other major issues to be taken into consideration:

(i) Several types of constraints, e.g. structure, sign, time delay, feedback

(ii) Large number of design variables, and

(iii) Due to (i) and (ii), a narrow feasible region compared to search landscape.

The proposed *Frequently Occurring Markov Blanket Genetic Algorithm* (FOMBGA) replaces the process of offspring generation (i.e. crossover and mutation) in GA by

(i) construction of a probabilistic model based on the estimated distribution of the selected individuals

(ii) generation of offspring according to the probabilistic model.

The FOMBGA approach continues to apply the GGA technique presented in the previous section. However, to further increase the effectiveness of the method, it eliminates the cross over and mutation operation resulting in a probabilistic model in which the children are generated stochastically. The MB based proposed FOMBGA technique finds local maxima for the MB of each node in the network. In a particular GA iteration, all the MBs of a given node obtained from different individuals of GA population form a set of MBs or a set of graph transactions. Such a set of graph transactions for a node under consideration can typically be of the order of 200,000 for a GA of population size 200. The frequency of occurrence of a particular MB sub-graph i.e. an edge or a V-structure (V-structure is one of the few standard structures existing within a GRN), is determined by the number of graph transactions in which the MB sub-graph occurs. In order to count and compute the frequency, a probabilistic method is used. We present the method next.

5.4.1 The FOMBGA method

Briefly, the proposed FOMBGA technique completely avoids the necessity of specifying set of standard GA parameters (crossover, mutation and selection) by replacing these operations with a process of probability estimation and sampling. The FOMBGA search uses a probability vector from a population of selected fit solutions for creating additional new solutions (offspring) for the next generation.

A. Schematic

A schematic showing probabilistic model GA is given in Fig. 5.11 below. A probability vector is applied to learn the probabilistic model from the distribution of promising solutions in each generation (i.e. iteration). This vector is updated by applying global statistical information extracted from the current population. Then, the offspring are generated by sampling this vector. From this perspective, the proposed approach is seen to be a probabilistic GA.

Each generation of the GA consists of the following steps:

1. Select promising individuals
2. Estimate the probability distribution
3. Based on this distribution, generate new individuals

The proposed stochastic search method, FOMBGA, employs the probabilistic model based on the above schematic. The algorithm examines various available fit models of a specific generation having different network connectivity around the nodes' Markov Blanket. To fully understand the operation of FOMBGA, we next consider the construction of a probability vector. Although the algorithm is based on Markov blankets (MBs) of the nodes in the GRN, for the sake of simplicity, we will initially explain the approach with the aid of individual edges in the network below. It will be extended below (in Section C) to the use of MBs leading to the FOMBGA algorithm.

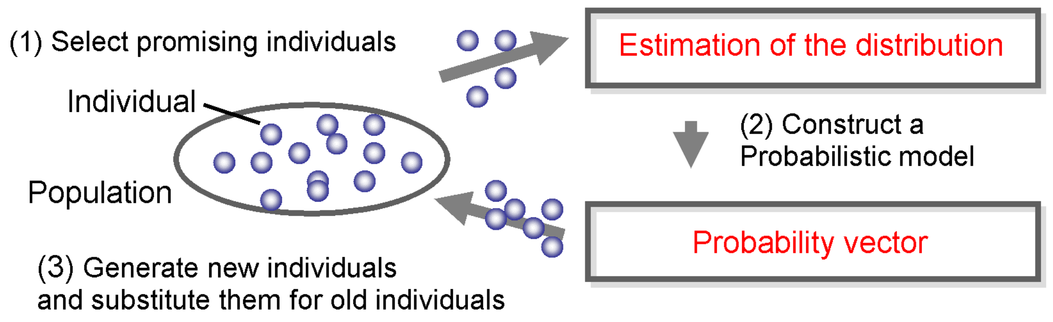

(1) Select promising individuals
Estimation of the distribution
Individual
(2) Construct a Probabilistic model
Population
Probability vector
(3) Generate new individuals and substitute them for old individuals

Fig. 5.11 Schematic representation

B. The Probability Vector

The GRN modeling tasks usually involves the generation of an adjacency matrix (A) (see Eqn. 3.12). As a result, each network (individual) in the GA population can be represented as a string of 1s and 0s. To elaborate further, consider an illustrative example of a six-edge gene network shown in Fig. 5.12 below. When the algorithm begins, the current population of binary strings is initialized randomly. In this example, the current population is shown to consist of four randomly generated members (110011, 111011, 010111, and 110001) whose elements are either 1 indicating presence of an edge and 0 otherwise, representing four small gene networks. The individuals are sorted according to the fitness score with highest scoring member ranked first.

The selection is limited to first 50%, i.e. top 50% of the best fitting population is chosen to be carried forward to the next generation. These selected individuals of the current population are also used as parent set for estimating and updating the probability distribution (vector). In Fig. 5.12, these samples are shown as 110011 and 101011.

With the single bit probability estimation that we are currently considering, given a n-bit gene network (n= total number of edges in the network), the approach involves creation of a n-bit probability vector $p = (p_1, p_2, p_3, \ldots, p_i, \ldots, p_n)$. The probability p_i is the probability of occurrence of 1 (existence of an edge) in the i^{th} bit position. The probability p_i is learnt by finding the number (i.e. the percentage) of 1s' in position i from the selected population. For example, we can see in Fig. 5.12 that 1 exists in position 2 only for sample 1 but sample 2 has a 0 in that position. Hence, the probability of having a 1 (i.e. finding an edge) in position 2 is 0.5. Similarly, in position 4, both the samples indicate the non-existence of an edge. This is represented in the probability vector by a 0 in the fourth position. Since we are considering only two samples, the probability vector consists of 1, 0.5 or 0 in this case. However, if the sample numbers is large, the probability vector would be distributed between values 0 and 1. The probability vector is learned and updated at each GA's iteration for modeling the distribution of promising solutions.

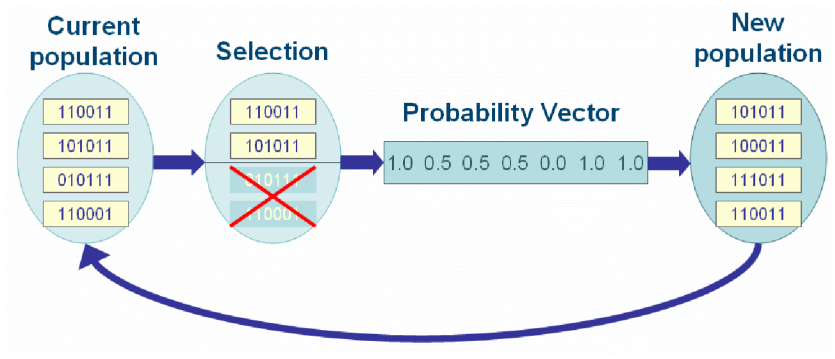


Fig. 5.12 Probability vector and generation of new population for a toy 6 gene network

After completing the first step of learning the probabilities, the next step is sampling. Since half (i.e. two samples in Fig. 5.12) of the samples were removed during the selection process, we need to additionally create the same number of samples for the new population. The general idea is to sample the vector in position i with probability p_i. For example, consider sampling the bit for position 2 of a new individual. Since the probability of an edge being present in position 2 is p_2=0.5 (from the probability vector), 1's and 0's are equally sampled in position 2. Similarly, we can sample the probability vector for other positions. From the illustrative example of Fig. 5.12, the newly added samples shown are samples 2 and 3. It may be noted that since the elements of the offspring are sampled from the probability vector, these are expected to be highly fit (i.e. being close to the promising solutions).

The probability vector discussed so far has been explained in the context of considering one bit at a time (single edge) in the network. It is applied to an overall putative network. With this, the bits that perform better get more copies and are combined in new ways with other bits of an individual. This technique may work in some cases, especially in a scenario where on an average, bit 1 (i.e. edge present) will outperform bit 0 (edge absent). However in many cases, the technique may not work because the context of each bit (i.e. occurrence of edge in relation to other edges or the nodes Markov blanket) which has significant effect on GRN construction gets ignored. Hence, working with single bits can produce misleading results. However, if in the approach, the context for each position (bit) could be learnt and utilized, then it can be very effective in the generation of probability vector and help solve the decomposable problem with greater computational speed and accuracy. The context of each bit can be obtained by the decomposition of the network based on the Markov blanket of each node and grouping the n-bit strings based on these Markov blankets. In the next section, we discuss this is in more detail and explain how the probabilistic approach can be easily extended and applied to the Markov blankets of the GRN.

C. Markov Blankets and Probability Vector

Referring to Fig. 5.13, we restate that the solutions are represented as the n-bit binary strings and appear as input to the probability vector. Each individual solution is actually representing a putative network. However, since all these putative networks are Bayesian networks, each of its nodes is independent of all other nodes given its MB. Hence, if we decompose the putative networks into smaller manageable networks which are MB for each gene, it will be equivalent to grouping the set of edges belonging to each MB together into a sub-string. Markov Blanket sub-networks are potentially more suitable for local search, as these networks are treated as independent of each other and hence it can be expected that the effectiveness of FOMBGA search will also increase significantly.

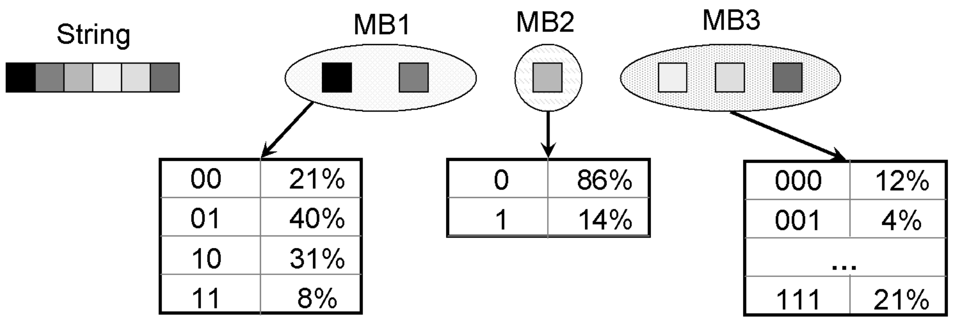


Fig. 5.13 Markov Blanket Probability vector

To clearly understand the application of Markov blankets for probabilistic modeling over the individual edge (1-bit) explained in Section B above, consider any randomly generated 6-bit binary string representing a 6 edges gene network. If we learn the probability vector based on network edges, we are computing probabilities such as *p(000000), p(000001), ..., p(111111).* However, if all the bits are considered together, we have to sample the vector considering all the 6 bits together, i.e. we are generating the entire network (i.e. string 000000) together with a probability of p(000000). Similarly, string 000001 is generated with a probability *p(000001),* and so on.

As we have seen before, the representation of an individual within a population is obtained by a n-bit binary strings. These individual can also be shown as a string of shaded blocks (see Fig. 5.13). Each block is representing one bit (i.e. 1 or 0) of the string. In Fig. 5.13, we show a randomly chosen solution represented as a string of six shaded blocks. Let us now consider the development of MBs from this six block string. Consider, the grouping together (i.e. generate a MB) of the edges as follows: (i) Group1 (or MB1) consisting of first two edges, (ii) Group2 (or MB2) has only edge 3 and (iii) Group3 (MB3) comprising of last three edges. For the two edges of MB 1, there are altogether four structures (combinations) possible: 00, 01, 10 and 00. These combinations can occur throughout the entire GA population with their respective probabilities. For example, considering MB1 of the entire population, we may find that in 16% of population, MB1 has both these edges absent. Thus, we have p(00) = 0.16. Similarly, for the other combinations, we may get, *p(01) = 0.45, p(10)= 0.35* and *p(.11) = 0.04.* Again for MB3 containing 3 bits, we can see that it will have 9 structures or combinations (such as 000, 001 … 111) possible. In this case, the probability of occurrence can be, say *p(000) =0.17, p(001) =2%.*

It is important that the essential edges in a network remain intact. Using GA, this is easily done by obtaining various skeletal structures by repeated trials with different initial values. From this, we select the high ranking individuals for estimating the probability distribution as explained earlier. In our experience, we have observed that all these high ranking individuals essentially have common edges. With the skeletal structure containing essential edges obtained, a knowledge acquisition process is then carried out as follows. Since it is difficult to simultaneously optimize all the MBs, we work with individual MBs within a network and determine their relative significance. From the highly fit and selected GRNs obtained above, each of the MBs are ranked individually. Similar to the single edge method explained in Section B earlier, we then apply the probability vector method to the highly ranked MBs (which form the initial individual set) and generate the probability vector from which we then obtain the highly fit offspring. Working with MB eliminates the optimization difficulty of the single-edge method occurring due to the n-bit string representing an individual may sometime contain many 0 values due to presence of futile edges. Moreover, working with MBs, a given problem gets divided into several simple problems, and the importance of each edge gradually becomes clear.

D. Guided FOMBGA search

To further enhance the effectiveness of the FOMBGA method, we next incorporate the heuristics to guide the search process. For this, the GGA technique presented in the previous section is extended for application to FOMBGA. It is schematically shown in Fig. 5.14 and works as follows.

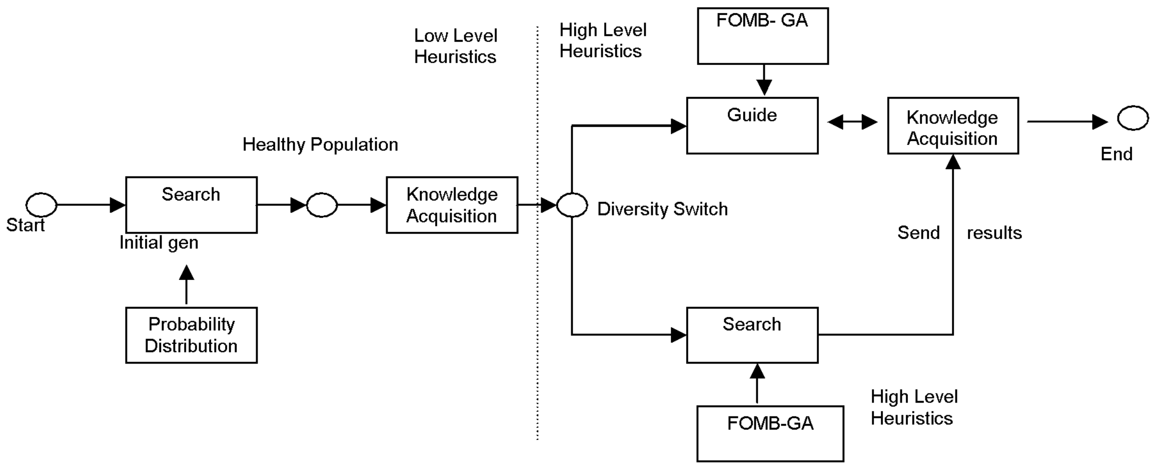


Fig. 5.14 The FOMGA strategy

Like GGA, the FOMBGA allows simple GA to initially control the first 20 generations to obtain a population containing relatively fit individuals. As shown in the Fig. 5.14, the main loop involves only the low level heuristic operators, namely random crossover and mutation for generating population. To implement crossover, gene edges from two random members of the population are randomly selected and swapped between the pair of networks. Mutation is applied on individual networks by random deletion of an edge or random addition ensuring that a directed cycle is not created. Usually, the individuals whose fitness values are high are chosen. Using these individuals, the knowledge acquisition is performed. Briefly, an ongoing knowledge acquisition process keeps track of each node's Markov blanket that passes through the GA. These MBs are ranked according to their dominance (by means of individual fitness score) to appear in the final network. A table containing

MB of each node in the network is prepared. During the high level GA heuristic operations (after first 20 iterations), the probability vector is learned from the ranked MB's and are sampled to generate the new population. The diversity switch selects between a low level and a high level GA heuristic operation. Diversity is a measure of variation between two individuals in a population. To calculate diversity, a mean skeleton network which is the basis for all networks for a population is obtained. The difference in the number of edges calculated between an individual network in the GA population and the skeleton gives the diversity value. The iteration repeats through the diversity switch and operators until the stopping criterion is reached. Since the GGA is stochastic in nature, the algorithm is repeated number of times and the resulting network structures are combined to reconstruct the final gene network.

5.4.2 Experiments and Results

The entire FOMBGA search technique described earlier is next evaluated by initially using the synthetic dataset. After validating the technique with synthetic data set, as an application of the technique, we also investigate the well understood yeast cell cycle data set and further examine various known and unknown gene interactions.

A. Synthetic Data

As stated earlier, since the synthetic network allows parametric variations[106], e.g. variation of sample size, they are appropriate for studying the robustness of the proposed method using large number of datasets obtained by parametric variation. For our experiments, we will apply the synthetic time series data obtained from synthetic networks to reconstruct GRN using the proposed probabilistic method. Then, the actual synthetic network is compared against the reconstructed network on an edge by edge basis with observations referred either as (i) positive (presence of an edge) i.e. P or (ii) negative (no edge present) i.e. N. Establishing whether the edge between node and rest of the network is positive P or negative N, is achieved by applying a statistical test on the node's conditional independence statement, *"Given a node X and its Markov Blanket, the node is independent of rest of the network T, if the test results in a value less than a specified threshold value γ."*

Put mathematically, this can be stated as:

$$\text{Test}\ (X;T|MB) < \gamma \qquad (5.6)$$

Since, multiple linear regressions are involved in computation of partial correlation coefficients, statistical tests such as t-test, chi-square test, F-test or p-value test can also be applied to determine the significance of a null hypothesis on the correlation coefficients. In our work, we used the stringent p value[71] and multiple-testing procedures for controlling the false discovery rate. To evaluate a realistic *p* value for the given data set, the target networks are initially chosen in a random manner. From this, the actual value of p was determined to be around a value of 0.4. Once a decision is made regarding the edge being positive or negative, it can be further classified into one of the following four categories of TP, TN, FP, and FN.

TP: True positive (edge exists in the original GRN and it is correctly identified to exist in the reconstructed GRN)
TN: True negative (edge does not exist in the original GRN and it is correctly identified not to exist)
FP: False Positive (edge does not exist in the original network and it incorrectly identified to exist)
FN: False Negative (edge exists in the original network and it incorrectly identified not to exist)

From these values, sensitivity and specificity is calculated as follows.

$$\text{sensitivity} = TP/(TP+FN)$$
$$(1\text{-specificity}) = FP/(TN+FP)$$

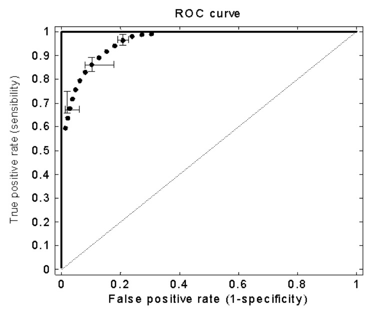
By varying the threshold value γ, we can change the values of TP, FN, FP, and TN with corresponding changes in sensitivity and specificity. ROC curves, which are plots of (sensitivity) versus (1-specificity), are then obtained.
The area under the curve (AUC) from the ROC plots is used as a metric for evaluating the reconstructed network and is a measure of test accuracy. Higher the AUC, better is the GRN reconstruction. As done conventionally, we apply the following classification for GRN reconstruction.

AUC =< 0.5	random
0.5 < AUC <= 0.7	poor
0.7 < AUC <=0.8	satisfactory
0.8 < AUC	good

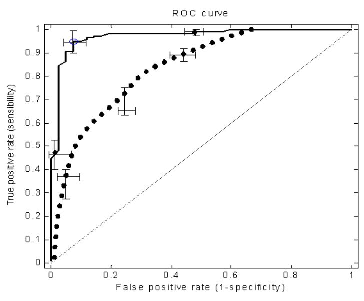
In the two different experiments utilizing the synthetic networks, we apply the ROC curves for establishing the improvements in accuracy of the reconstructed network using the FOMB GA method.

i) Effect of Markov blanket

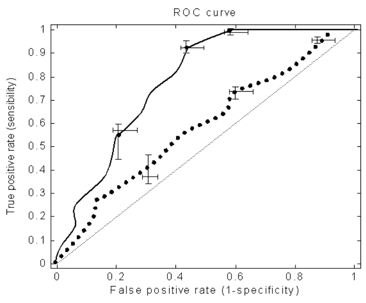
The FOMBGA method in which the probability vector is developed on a single edge will be referred as the single edge method. Similarly, when the method uses Markov blankets of the nodes for generation of probability vector, the method will be referred as the MB method. While single edge approach will consider single edge on its own, the MB approach will consider a set of edges within a MB will be grouped together. For conducting experiments, three 40 gene synthetic networks are arbitrarily generated with sample sizes of 200, 100 and 50. The parameters (sample size and number of genes) are chosen such which it will results in networks facilitating easy experimentation.



(a)



(b)



(c)

Fig. 5.15 ROC plots. Legend: X-axis -- (1-specificity), and Y-axis -- (sensitivity). The dotted line (1) indicates the ROC curve for the single edge approach while the dark continuous line (2) refers to ROC curve of MB method.

Fig. 5.15 summarizes the results for varying sample sizes. Fig. 5.15 (a), (b) and (c) are respectively for gene networks with samples sizes of 200, 100 and 50. To make the observations statistically meaningful, each data point in the Fig. 5.15 is obtained by averaging the values of 5 repetitions of data generation and network estimation. The dotted line (1) indicates the ROC curve for the single edge approach while the dark continuous line (2) refers to ROC curve of MB method. For sake of comparison, a ROC curve of a random network (diagonal line in the Fig. 5.15) is also shown. It can be clearly seen that both the ROC curves appear above the diagonal confirming that the reconstruction of GRN in both these cases is not random. The area under the curve (AUC) is a measure of accuracy of reconstruction. From Fig. 5.15 (a), we observe that with 200 samples, the curve for the single edge method is slightly above 0.75 (calculated by estimating the probability of presence and absence of edges in the network). However with the frequent MB method, we note that it produces a 100% accurate reconstruction of the synthetic network (AUC=1). Further, from the three figures, we can observe that as the number of samples is reduced from 200 to 100 or 50, the ROC curve corresponding to 'single edge' method degrades significantly in comparison to the MB method. These results are consistent with our above discussion of demerits with the use of single bits. The increase in reliability comes at the expense of increased computation cost however reliability might be of higher priority, considering the high complexity of gene network models and the noisy data. Hence, there is better accuracy and reliability with the FOMBGA algorithm with MB based probability vector.

The Fig. 5.16 below is a bar graph plot of the Markov Blanket scores for the 40 gene synthetic network with 200 samples. The plot is for the best fitting solutions of the three search approaches, namely SGA, GGA and FOMBGA. The fitness scores are computed using the scoring method stated in Chapter 3 Section 3.4. Briefly, a series of conditional independence tests are carried out to check the compatibility between the model and the data. The score is incremented by 1 if the test is passed with a certain degree of statistical significance and penalized by 1 if the test fails. The larger the Markov blanket, more the number of tests to be carried out and hence higher the score. From Fig. 5.16 it is seen that, in general, all larger MBs with high value of scores are inferred correctly by the FOMBGA method. However, for nodes with smaller MB, (e.g. node 23), the FOMBGA score is slightly lesser than GGA and SGA. This should not be construed as failure of algorithm for smaller MBs.

The reason for the smaller FOMBGA score for smaller nodes is because the working of the algorithm is based on the "frequency" of occurrence of MB in generating new offspring. This forces more frequent (and hence important) MBs of the network to get optimized first with a much higher score causing the less frequent MBs to become smaller in size leading to a significantly smaller score. However, since the overall fitness score of the network is formulated as a linear summation of individual MB scores, it causes the network fitness score obtained by FOMBGA to be higher SGA or GGA resulting in a more accurate reconstruction of the network.

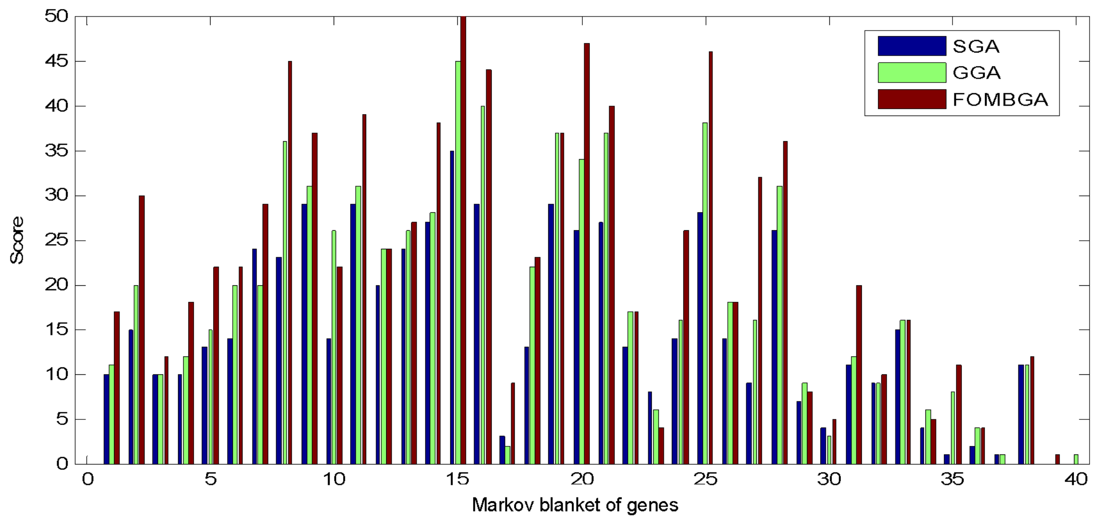


Fig. 5.16: Bar plot of the MB fitness score for the 40 gene synthetic network. MB fitness score of each node for the three search methods, SGA, GGA and FOMBGA are shown.

ii) Comparison with other GAs

The performance of FOMBGA is compared with simple GA and guided GA. The population and the probability vector are initialized for the search to a randomly generated 200 individuals (networks). The population is evaluated for fitness and ranked accordingly. The top 50% of the population is selected as the fittest strings to become parents and then update. The elite rate is initially set to 20% and then subsequently reduced to 10%. Thus, the similarity between an offspring and its parent is, to an extent, controllable. The resultant offspring networks join 10% of their parents to form the population of the next generation. The larger the percentage, the more elements are sampled from the probability vector. In other words, similar to the mutation rate in conventional mutation, the algorithm controls the similarity between offspring and the parent, while the parent can be chosen from the best solutions found so far. The migration interval is set to 10 generation. If all the members in the current population are identical, the search will migrate to a new area in the search space.

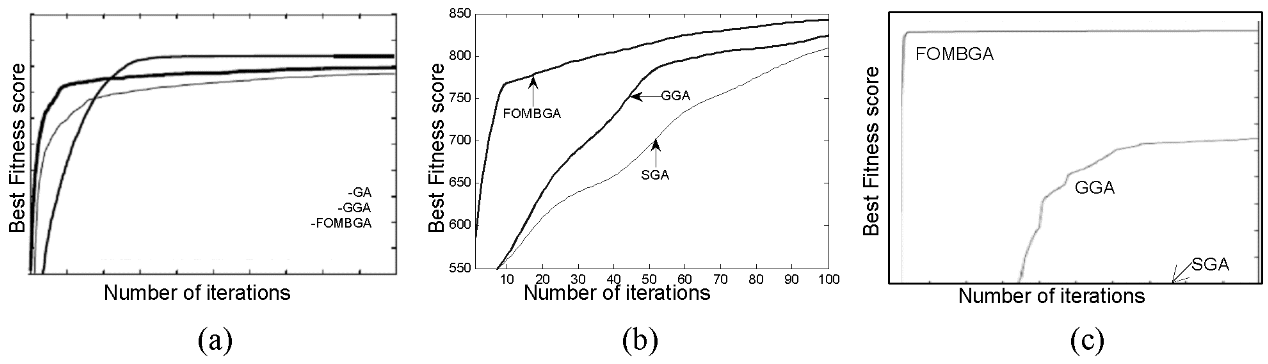


Fig. 5.17 Comparison between FOMBGA, GGA and SGA for the GRN modeling problem for 100, 200 and 500 gene networks. X-axis: represents generation and Y-axis represents the best fitness scores per generation of population.

In order to compare the GA, GGA and FOMBGA algorithms, we varied the network size (number of genes) as shown in Fig. 5.17 summarizing the results obtained from these experiments. Fig. 5.17 (a), (b) and (c) show the ROC curves respectively for a series of experiments on synthetic data of the gene networks of 100, 200 and 500 genes. First, because of the less number of genes in the data (Fig. 5.17 (a)), all GA's showed almost identical curves, making a more detailed analysis difficult. However, when the number of genes was increased to 200, for example in Fig. 5.17 (b), the FOMBGA converges faster than SGA and GGA. Similarly, with 500 genes (Fig. 5.17 (c)) the FOMBGA converges faster and results in even higher score. On the contrary, the GGA resulted in a suboptimal performance and the SGA resulted in a still poorer solution.

We have restricted the experiments to just varying the number of genes rather than including the variation of samples, as varying number of samples would have the same impact on all three algorithms since the same fitness metric is used in all three cases (SGA, GGA and FOMBGA). In other words, the differences between the optimal networks do not depend on the number of samples. Further, although there are several other techniques such as simulated annealing (SA) and ant colony optimization (ACO) [77] applied for synthetic GRN reconstruction, comparison with them is not possible as these models use Boolean variables instead of continuous values used in our model.

Since there is lack of a benchmark dataset to compare various GRN reconstruction methods[17, 46, 54], the comparison becomes difficult overall and painstaking. Hence we justify the methods performance with the use of synthetic datasets. Furthermore, most methods work with discrete data and perform experiments on small toy networks. Hence we are able to do a one to one comparison with other works.

B. Real life dataset

A real dataset is far more complex than the synthetic data studied earlier. To demonstrate the learning capabilities of FOMB GA approach and also as a practical application of the method on a real biological dataset, we consider the yeast dataset [49] containing 800 genes and 77 samples comprising of a comprehensive catalogue of cell cycle-regulated genes in the yeast Saccharomyces cerevisiae. The dataset includes three long time course expression values representing three different ways of synchronizing the normal cell cycle, and five shorter time courses representing the altered environmental conditions. These results were combined with those by [125]) to produce a more comprehensive collection of data. The test samples were synchronized so that all the cells would be at the same stage in their cell cycle. Using this data set, gene networks have already been reconstructed [17]. We also note that the Spellman dataset has classified the 800 genes in different phases of cell cycle such as G1, G2, S and M/G1 and G2/M.

The measurement noise level in the data is usually accounted by modeling it as Gaussian noise s which helps eliminate genes with a significantly small SNR (signal to noise ratio) thereby increasing the accuracy of our reconstruction. In our work, the Gaussian noise modeling is carried out by obtaining a histogram of the genome-wide expression ratios taken at the first time point when the cells are just about to grow. Normalized gene expressions with no noise have values are close to the mean values. To the Gaussian distribution obtained, we apply a standard deviation =0.19 to eliminate the noisy genes. Further, the data set is partitioned into categories referred as alpha-factor, elutriation, CDC15, and CDC28 temperature-sensitive mutants.

(i) Experiments and results

The S.cerevisiae gene expression data mentioned above was obtained by disrupting 100 genes containing the known transcription factors.

Table 5.4 Five significant genes under investigation

Gene	Description/Regulators from YPD database	Predicted interactions from the reconstructed GRN
MCM1	Transcription factor of the MADS box family CDC6, CDC5, SIC1, STE6, CLN2, STE2, ACE2, SWI5, CLB1, CLB2	YGR177C, YLR131C, YGR108W, YMR001C, YOL158C, YOR274W, YOL043C, YLR071C, YIL158W, YJL051W
SWI5	Transcription factor CDC6, SIC1, CLN2, PCL2, PCL9, EGT2, RME1, CTS1, HO	YBR158W, YLR295C, YKL185W, YOR264W, YNR067C, YDR512C, YDR516C, YOR31W
ACE2	Metallothionein expression activator CLN2, EGT2, HO, CTS1, RME1	YER124C, YGL028C
SNF2	Component of SWI/SNF global transcription activator complex CTS1, HO	
STE12	Transcriptional activator STE6, FAR1, KAR3, SST2, FUS1, STE2, BAR1, AGA1, AFR1, CIK1	

We now apply the FOMBGA method to micro array data of these 100 genes from which we estimate the 100 gene network. From this reconstructed GRN, based on the Markov blankets (which include other genes as parents, children etc.) of the 5 regulatory genes, we extract the combined 36 gene network. Fig. 5.18 shows the reconstructed regulation network. By comparing the relationships in the 36 gene network with the regulatory relationships from Yeast Proteome Database (YPD), we are able to validate the presence of 15 target genes (CLN1, CLN2, CLB5, CLB6, GIN4, SWE1, CLB4, CLB1, CLB2, TEM1, APC1, SPO12, CDC20, SIC1, FAR1) and 19 edges. These interactions are shown in Fig. 5.18. The Figure also shows 11 yeast transcription factors (SWI4, SWI6, STB1, MBP1, SKN7, NDD1, FKH1, FKH2, MCM1, SWI5, ACE2) and one cyclin gene (CLN3) which are known to activate the cell-cycle dependent genes [51].

Again, from the 800 genes dataset, it is also possible to obtain periodic interactions during the cell cycle [49]. However, only some of the activators-target pairs of these interactions are known interactions. For our studies, only the 15 target genes mentioned earlier whose transcription activators are known are analyzed further.

ii) Validation

From the reconstructed network shown in Fig. 5.18, we note the following interactions taking place.

- MBF (a complex of MBP1 and SWI6) and SBF (a complex of SWI4 and SWI6) controls the late G1 genes (e.g. CLN2 and NDD1).
- MCM1, together with FKH1 or FKH2, recruits the NDD1 protein in late G2 and controls the transcription of G2/M genes.
- SWI5 and ACE2 regulate genes at M/G1.
- MCM1 regulates SIC1 and ACE2.

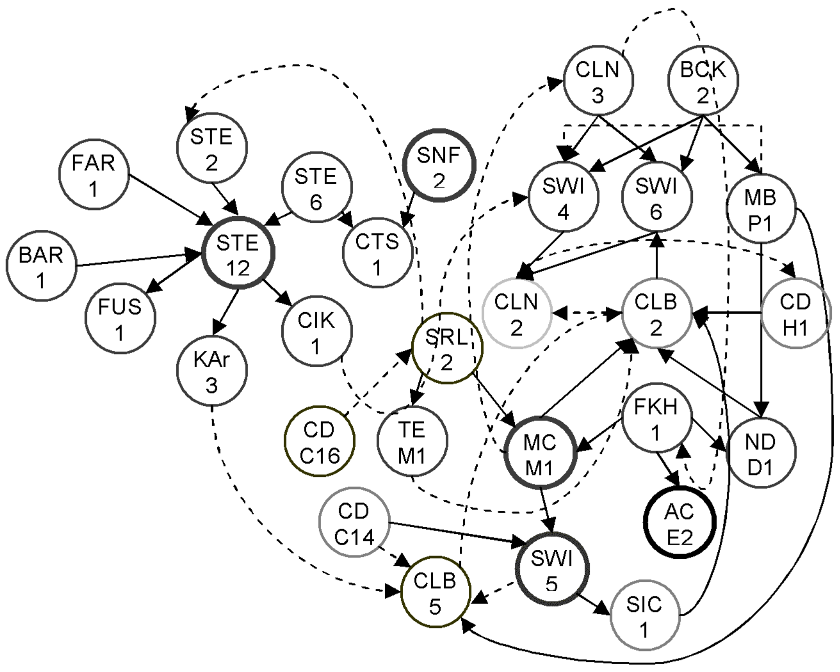


Fig. 5.18 Gene Regulatory network reconstruction

These observed interactions are in complete agreement with the information obtained from the Spellman's dataset. We note further that the relationships around MCM1 has improved significantly because FOMBGA has been able to model the MCM1 Markov blanket more accurately than the SGA or GGA techniques that we had applied earlier. Further, we find that 10 genes (Table 5.4, Column 2), out of 24 genes that are listed as co-regulated genes of MCM1 in the YPD database are also identified correctly along with their interactions.

A positive (negative) edge weight represents activation (repression) in the causal model. In Fig. 5.18, these are shown as continuous (for positive) and dotted (for negative) lines. We observe that repressions through some of the transcription factors are actually activating target genes in the cell cycle. For example, from Fig. 5.18, it is evident that the edge FKH2→CLB2 is a negative regulatory relationship indicating a dominating role of FKH2 on CLB2 compared to the positive influence of MCM1, FKH1, and NDD1 on CLB2. The influence of FKH2 on CLB2 that we have identified is dominating since the transcriptions of genes such as CLB2 are more likely to be effected by the joint binding of multiple transcription factors on the promoter region of the gene in the DNA.

Again, the relationships surrounding STE12 have become more accurate with the FOMBGA method. Before adding FOMBGA, the previously estimated networks in Chapter 3 incorrectly concluded that STE12 regulates genes from amongst FUS1, AFR1, KAR3, BAR1, MET4, MET16 and MCM1, and that STE12 is controlled by HO, STE6 and MET3. However, the network of Fig. 5.18 obtained by application of FOMBGA clearly shows STE12 regulating FUS1, AFR1, KAR3, CIK1, STE2, STE6, HO and MCM1. The correctness of this regulatory interaction is validated with the latest information available from YPD database thereby confirming further the superiority of the FOMBGA search approach over other techniques.

5.5 Summary

Since exhaustive search in the structure space is impractical and exact inference with BNs is NP-hard, stochastic approximation to the search for high-scoring network structure is necessary. A novel guided GA strategy for learning causal Bayesian network structures is presented. The approach employs a diversity switching to adaptively alternate between ordinary operators and guided operators. The algorithm also combines a ranking schema; multiple path constraint and standard Gaussian function to perform high level heuristic operations. We have conducted experiments using artificial data and real world microarray data which demonstrate the superior performance of our GGA with SGA. To further enhance GGA performance, we proposed a Frequently Occurring Markov Blanket Genetic Algorithm (FOMBGA). The FOMBGA replaces crossover and mutation operators with a probabilistic model on frequency of occurrence of fit Markov blankets (MBs). The experiments were carried on various synthetic datasets as well as a real life yeast cell cycle data. The studies on synthetic data show the superiority of FOMBGA over both SGA as well as the GGA. The results of yeast cell cycle regulatory network appear promising since 15 genes and 19 reconstructed regulatory interactions are found to be consistent with the known biological findings and also because of the new plausible interactions that have been predicted for the network.

Other than development of a suitable model and an associated search technique, it is also important to investigate the mechanism of parametric learning of the BN based GRN. This is the focus of next chapter.

6. Parameter Learning based on Markov Chain Monte Carlo approach

6.1 Introduction

At a qualitative level, the structure of a Bayesian network describes the relationships between these genes in the form of conditional independence relations. At a quantitative level, relationships between the interacting genes are described by conditional probability distributions (CPDs). The probabilistic nature of this approach is capable of handling both biological and technical noise and makes the inference scheme robust and allows the confidence in the inferred network structures to be estimated objectively. However, the application of BN learning to gene expression data in understanding the mechanism of GRN is particularly hard because the data sets are very sparse, typically containing only a few dozen samples but thousands of genes. Here, we devise computational methods that consistently identify causal and dependence relationships between expressions of different genes.

Estimation of parameters is the estimation of conditional probability distributions (CPD) of the given GRN. Due to high dimensional data, exact computation of the CPDs is infeasible and computationally expensive. Hence, the joint distribution can be approximated by stochastic simulation commonly referred as sampling. Using the well known Monte Carlo algorithm based on random sampling, we can fit a distribution to the data and retain the samples. However, random sampling from the GRN is not the best strategy since the state space is enormous with large number of samples needed to approximate the probabilities reasonably well. Hence, many times most representative samples are picked which increases efficiency and creates a 'Markov Chain'. This approach has resulted in Markov Chain Monte Carlo (MCMC) method and its variants [72, 73, 83, 92].

In this chapter, we propose a new MCMC approach to approximate the conditional probability distributions of complex GRN models. The proposed approach is essentially based on two novel concepts. The first is an efficient computation of CPDs based on the ordered ranking of Markov Blankets (MB). We choose MB for ranking, because it is aligned with our work for structure search, reported in previous chapters, produced promising results. The genes with high scoring MBs tend to be more accurate allowing much faster convergence compared with a stationary distribution of the Markov chain. The second novelty of the approach is progressively reducing the space by clamping those variables whose samples have converged to a fixed distribution thereby limiting convergence process to a increasingly narrower region. Empirical results are presented to illustrate the superiority of the approach over direct MCMC and random sampling. Furthermore, we extend our work and create networks at higher levels of detail by integrating the regulatory sequence analysis and the GO data. Although interacting genes usually have similar gene expression patterns, their profile similarity varies widely mainly due to their complexity of regulation. Studies are performed using not only the synthetic data sets but also the real life *Saccharomyces cerevisiae* (yeast) [49] microarray dataset. From the significant motifs identified by carrying out regulatory sequence analysis over all genes at different levels, the most significant genes which characterize the degree of relationship among the genes are selected. The resulting genetic networks are refined by combining the results with regulatory sequence analysis. Gene ontology information is next used to annotate these groups of genes.

The chapter is organized as follows. Section 6.2 gives a brief outline of probability distribution and sampling techniques. In Section 6.3 the concept of markov chain and gibbs sampling is explained. The proposed MCMC method is presented in Section 6.4. Section 6.5 presents the experiment and results using both synthetic and real data. Section 6.6 deals with integration of biological knowledge to the inferred network. Section 6.7 provided the summary of the chapter

6.2 Probability distribution and Sampling

As stated above, estimation of parameters is basically the estimation of conditional probability distributions (CPD). Estimating CPDs involve specifying $P(X \mid pa(X))$ for each of the gene (variable) X where the term $pa(X)$ refers to parents of variable X in the given structure. Assuming that the inferred structure G, is an Independence-map (I-map) of a probability distribution P, we note that $I(G) \subseteq I(P)$ where $I(G)$ represents independence assertions in graph G and $I(P)$ is the independence assertions in the probability distribution P. Since G is an I-map of P, P factorizes according to joint probability distribution (JPD) given by Eqn. 6.1.

$$P(X_1, \ldots. X_n) = \prod P(X_i | pa(X_i)) \tag{6.1}$$

The network is a pair (G, P) where G is specified in edges and P is specified in CPDs. With several optimal graphs G equally representing the distribution P, $I(G)$

becomes a subset of *I(P)* as shown in the Fig 6.1 below implying that we can obtain $P(X_1 \ldots X_n)$ from *G*. Once we obtain *P*, it is possible to deduce a unique minimal I-map *G*. Removing any edge from the minimal *G* then induces conditional independencies that do not hold in *P*.

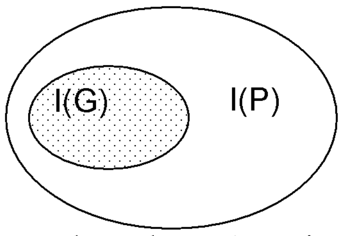


Fig. 6.1 Independence Assertions

Next, we briefly elaborate on the probability distribution and sampling for GRN with a focus on Gibbs sampling which is a type of Markov Chain Monte Carlo (MCMC) sampling.

6.2.1 Probability distribution

A GRN based on Bayesian network specifies a probability distribution through a directed acyclic graph (structure) and a collection of conditional probability distribution (parameters) for each gene X_i in the graph *G*. The graph *G* captures conditional independence relationships in its edges. A gene (node) is conditionally independent of all other genes (nodes) in network given its Markov Blanket (parents, children, and children's parents). The probabilities summarize a potentially infinite set of circumstances that are not explicit in the model but rather appear implicitly in the probability. If each gene (variable) is influenced by at most *k* others and we have *n* random genes (variables), then we only need to specify $n*2^k$ probabilities instead of 2^n. Succinctly, conditional probability distribution shows the probability distributions over all values of gene X given the values of its parent genes. Conditional probability distribution of $X=x$ given $Y=y$ is given by Eqn 6.2.

$$P(x \mid y) = \frac{p(x, y)}{p(y)} = \frac{p(y \mid x)p(x)}{p(y)} \tag{6.2}$$

If genes x and y are independent, then $P(x \mid y) = p(x)$ since $p(x, y) = p(x)\, p(y)$. The Eqn. 6.2 above is repeated to condition X on all parent genes of X. The parents of gene X_i are all those genes that directly influence gene X_i from the set of genes X_1, …,X_{i-1}. Since large GRN models will have more parameters, the exact computation is, therefore intractable and in such cases simulation (sampling) technique becomes suitable for approximating conditional distribution. The structure G, necessary for sampling, is obtained by applying a structure search over the entire space of all possible structures. Hence, given structure G with genes $X=\{X_1, X_2....X_n\}$, we can draw a sample from the joint probability distribution as follows:

i) Instantiate randomly all except one of the genes, X_i

ii) Compute the probability distribution over the states of X_i, i.e. $P(X_i|X_1...X_{i-1},X_{i+1},...X_n)$

iii) From the probability distribution, randomly select a state of X_i

If all genes in the network except the gene X_i are instantiated, then due to the factorization of the joint probability distribution, the full conditional for a given gene in the DAG involves only a subset of genes participating in its Markov blanket (i.e. the set of parents, children and other parents of the children for the gene).

$$P(X_i|X_1...X_{i-1},X_{i+1},...X_N) = P(X_i|MB(X_i)) \tag{6.3}$$

Here, $MB(X_i)$ is the Markov Blanket of gene X_i. Since gene X_i is independent of rest of the genes in the network (except its Markov blanket), it is necessary to consider only the partial conditional conditioning on the Markov blanket.

Furthermore,

$$P(X_i|MB(X_i)) = P(X_i|Pa(X_i)) \prod P(Y_i|Pa(Y_i)) \qquad (6.4)$$

Here, Y_i, $i = 1, \ldots k$ are the children of gene X_i.

6.2.2 Sampling

The sampling using Monte Carlo methods involve drawing *n* samples from the GRN with the instantiated genes fixed at their values as explained above. From these samples, the probability distributions are estimated based on frequency of occurrence of genes. Since our model involves continuous expression values, we plot these samples as a histogram and then *smooth* the histogram to give the probability density function of the genes. The instantiation of the genes is done using the distribution available from the data set. However, due to typically large number of genes in a GRN, random sampling methods are not suitable because they can be slow and the posterior distribution estimated may not be reliable. Markov Chain Monte Carlo (MCMC) approach is suitable in such cases for approximating the difficult high dimensional distributions. From amongst the many MCMC methods available, we choose Gibbs sampler which results in obtaining samples asymptotically from the posterior distribution and can provide convergence in reasonable computation time. The Markov chain and Gibbs sampler is discussed in detail in the next section.

6.3 Markov Chain and Gibbs Sampling

A MCMC method such as Gibbs sampler which is applied for sampling probability distributions is based on constructing a Markov chain. We briefly present the concept of Markov chain followed by the Gibbs sampling technique.

6.3.1 Markov Chain

The Markov chain includes the probability of transitioning the variables from their *current* state s to the *next* state s' based on the transition probability $q(s \rightarrow s')$. If the state distribution $\pi_t(s)$ describes the probability of genes being in state s at the t-th step of the Markov chain, then the stationary (equilibrium, invariant) distribution $\pi^*(s)$ will occur when $\pi_t = \pi_{t+1}$, i.e.

$$\forall s' \qquad \pi(s') = \sum_s \pi(s) q(s \rightarrow s') \tag{6.5}$$

We note that the stationary distribution also satisfies the detailed balance Eqn. 6.6 given below.

$$\forall s, s' \qquad \pi(s) q(s \rightarrow s') = \pi(s') q(s' \rightarrow s) \tag{6.6}$$

No matter what the initial state distribution is, a Markov chain converges to $\pi^*(s)$ if it fulfils the following conditions: uniqueness, aperiodicity and irreducibility. The condition of aperiodicity ensures that the chain can not get trapped in cycles or the state transition graph is connected. The irreducibility condition ensures that for any state, there is a positive probability to visit all other states. An aperiodic and irreducible Markov chain is called ergodic [126] and ensures that every state much be reachable from every other and there can be no strictly periodic cycles.

Using Gibbs sampling, we propose to design a Markov chain whose stationary distribution is the target (desired) distribution such that gene X_i quickly converges to the stationary distribution irrespective of the initial distribution. From this, we then run the chain to produce a sample; throwing away the initial (burn-in) samples as these is likely to be influenced by the initial distribution. The sampling method for the target distribution π^* on χ_v constructs a Markov chain S^0, S^1, . . . , S^k, . . . with *π*(s)* as equilibrium distribution. Since the distribution *π*(s)* is a unique equilibrium, and the Markov chain is ergodic, we have

$$\forall s \qquad \pi^*(s) = \lim_{n\to\infty} \pi_n^*(s) = \lim_{n\to\infty} \frac{1}{n} \sum_{i=m+1}^{m+n} \chi_v(S^i) \tag{6.7}$$

Where n is the number of iterations. The state of the chain obtained after a large number of steps is then used as a sample and its quality improves with the increase in the number of iterations. When a dynamic equilibrium is reached, the long-term fraction of time spent in each state is exactly its posterior probability for the given conditions. As number of iteration tends towards infinity, all statistically important regions of state space will be visited.

6.3.2 Gibbs Sampling

To perform a MCMC simulation on GRN where the target distribution is the joint probability distribution, we design a Markov chain where each state is a full joint instantiation of the distribution (i.e. values are assigned to all variables). Hence, a transition is a move from one joint instantiation to another. The target sampling distribution *π*(x)* of the GRN is the posterior joint distribution *P(x | e)* where *x* is the set of unknown variables and *e* is the set of evidence variables. It is typically the unknown we want to evaluate. Although sampling methods such as logic sampling [88], rejection sampling [127] and importance sampling [48] are available to sample *P(x|e)*, in the absence of evidence e or with the probability of evidence being small (i.e. if *P(e) ~=0*), these algorithms result in many wasted samples. The Gibbs sampling overcomes these limitations as it specifically uses conditional distribution *P(s' | s)* to define state transition rules.

In Fig 6.2, an example of Markov Chain for a 4 gene GRN is shown. We have specifically fixed the Gene *B* and *D* values and Gene *A* and *C* are varied to produce 4 states.

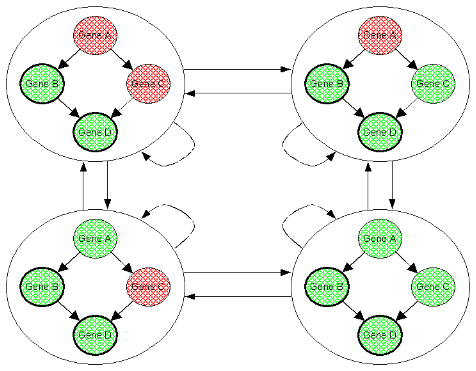


Fig. 6.2 Example Markov Chain for a toy 4-gene network. The genes B and D are instantiated as true while the genes A and C are false.

The working of the Gibbs sampling algorithm is shown by the flow chart in Fig. 6.3. Consider a GRN with n unknown variables $X_1, X_2, \dots X_n$ which appears as input to the algorithm. We now recall that a gene X_i is independent of rest of the network given the variables in the Markov blanket (MB) of X_i, i.e.

$$P(X_i \mid X_2, \dots X_n) = P(X_i \mid MB\,(X_i)) \tag{6.8}$$

The Markov condition that a variable is independent of all other variables (except its neighbors) reduces significant computational overhead especially for large scale problems. Calculating $P\,(Xi \mid MB\,(Xi))$ can be done using Eqn. 6.4 and Eqn.6.6. The initial states of all the variables can be chosen randomly or these can be chosen from the original small sample dataset. If the current state is $X_1 = x_1, X_2 = x_2, \dots, X_n = x_n$, then we can sample a new value x'_1 for X_1 from $P(X_1|X_2 = x_2, \dots, X_n = x_n)$. In similar manner, we can sample the remaining new values for $X_2, X_3 \dots X_n$ until we have a new state $X_1 = x'_1, X_2 = x'_2, \dots, X_n = x'_n$. The initial samples are influenced by the initial distribution. At every step, we weigh our selection towards the most probable sample using the transition probability so that the samples follow the most common states accurately. Moreover, as the process is ergodic (i.e. it is possible to reach every state), it will ensure convergence to the correct distribution if sufficient number of iterations are carried out.

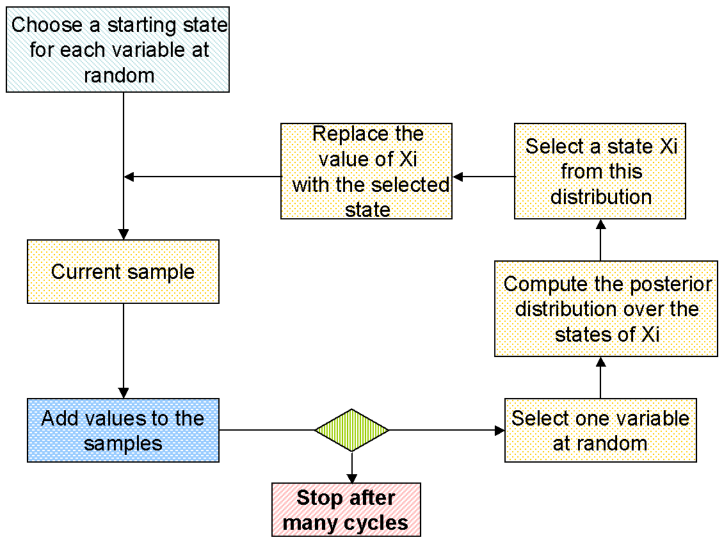

Choose a starting state for each variable at random
Replace the value of Xi with the selected state
Select a state Xi from this distribution
Current sample
Compute the posterior distribution over the states of Xi
Add values to the samples
Select one variable at random
Stop after many cycles

Fig. 6.3 Gibbs Sampling

However, the application of Gibbs sampling for GRN estimation is somewhat limited due to the high dimensional data where number of genes is significantly higher than the samples. This means that the variance in the values taken by the variable is high, and can increase dramatically for thousands of genes and may prohibit producing independent uniform samples during sampling. The proposed new methodology for based on novel Gibbs sampling for the GRN estimation problem can overcome this limitation.

6.4 Proposed MCMC Sampling Scheme

The proposed MCMC sampling scheme is shown in the Fig. 6.4 below. In our earlier work, we employed a guided GA [128] search strategy where we had obtained a set of 10 dissimilar high scoring network structures closely representing the probability distribution using the gene expression data. With the aid of proposed methodology, we will now calculate the Bayesian posterior probability distribution of all the variables (genes) of the ten gene network structures. From the samples drawn from a network structures, we can obtain the posteriors after convergence, and then determine the state sequence and probability estimates of the model in a straight forward manner. Although the inferred high scoring network structures are disjoint (i.e. cannot be combined into one network structure), they can all be combined independently to the underlying probability distribution. Hence, all these network structures are sampled to estimate the probability distribution accurately. The important feature of our approach is the use of high scoring initial networks and a rank ordering on the network genes using Markov blankets. The convergence is obtained by running several Markov chains in parallel. Let us briefly discuss the major 'constituents' of the proposed method as they occur in Fig. 6.4.

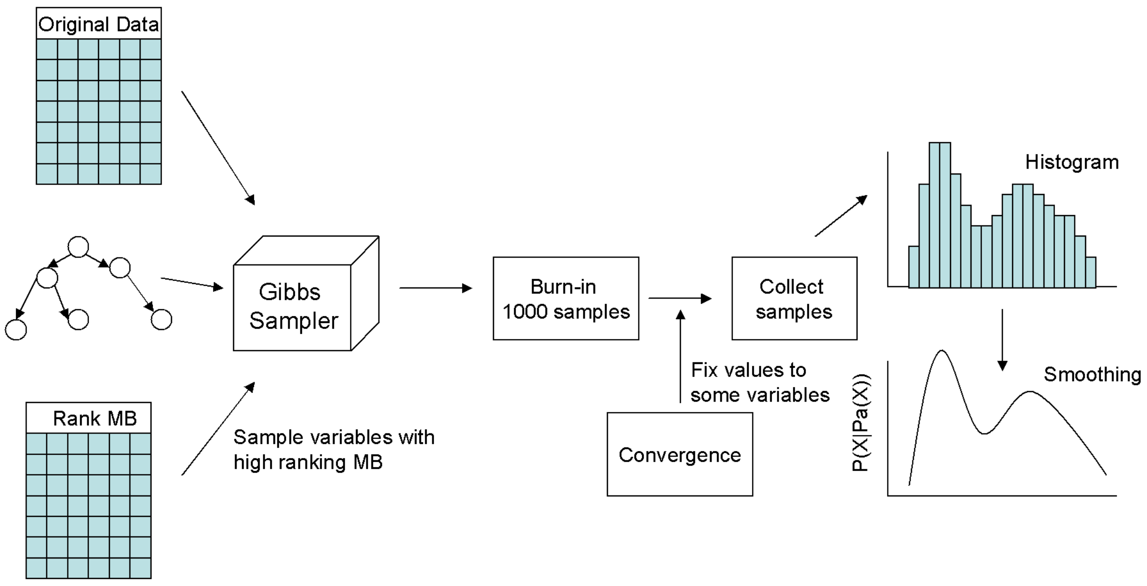


Fig. 6.4 Proposed MCMC Sampling Scheme

6.4.1 Rank ordering of the variables

As explained before, an ordinary Gibbs sampler (MCMC) chooses genes at random and then samples a new value from the estimated posterior of the neighbouring variables (i.e Markov Blanket variables). Friedman and Koller [64] argued that sampling from the space of (total) orders on variables rather than directly sampling DAGs was more efficient than application of ordinary MCMC directly in random manner. In our previous work [129], evaluating a network structure was based on the summing of scores of the individuals genes in Markov Blankets. Since the Gibbs sampler also samples the new value of a gene based on the MB variables, we will order the rank of the Markov Blankets based on their scores.

6.4.2 Gibbs Sampler

Before we proceed with the Gibbs sampling scheme, we need to specify a uniform prior distributions for all the genes in the domain. Rather than a random initial state of the network, we apply a standard prior which is a multivariate *Dirichlet* distribution [64]. This distribution is assigned to initial state distribution and also to the state transition distribution of the Markov chain. The initial distribution of the variables in the network (from which the initial state is sampled) is assigned using the density function estimated after smoothening of the histogram of normalized gene expression data. Sampling is straightforward as there is no evidence in the network and is done by sampling each variable in the specified rank order. For nodes without parents, sampling is done from their initial distributions while for nodes with parents, we sample from the conditional distribution of their MBs. Similarly, n independent and identically distributed samples are drawn from the target distribution P(x). Since the samples drawn are continuous (through the normal range of -3 and +3) rather than discrete, the sampling precision is restricted to two decimal places to reduce the space of complexity. The samples collected are plotted using a histogram with n bins as shown in Fig. 6.4 above. The probability density function P(x) of a continuous variable (gene expression) x is approximated by smoothening of the histogram of samples as shown in the Fig. 6.4. Similarly, the conditional probability distribution of all variables is estimated.

6.4.3 Burn-in and Convergence

The process of achieving stationary probability distribution is called as convergence while the initial phase of convergence is called the 'burn-in' phase. In the proposed method, the convergence is improved by running several parallel Markov chains each using a different network structure representing the probability distribution as the starting point. The idea of running multiple chain using different Bayesian network structures is mainly to obtain samples from the entire sample space of the probability distribution underlying all the structures. The chains are merged together at a certain stage of the iterations and made into a single chain.

During the process of multiple chain runs, samples are exchanged between the chains and the overall samples of a number of variables in the top of the specified order are monitored for autocorrelation and stationary distribution. A sample variation factor is introduced to determine the fraction of samples that go out of range. When the sample values do not go above a variation factor after significant number of iterations, we assume the samples have converged. From there onwards, the variable is clamped to the stationary value. This allows the sampling to be carried out on the variables that are in the lower in the rank order of the variables. In our experiments we find that the rank ordering of variables, multiple Markov chain runs and clamping also improves the mixing of samples for the unknown variables improves the mixing of samples more efficiently than an ordinary MCMC approach.

6.5 Experiments and Results

The validation of new techniques by comparing with other GRN reconstruction methods becomes difficult and painstaking due to non-availability of suitable benchmark dataset. Furthermore, most methods work with discrete data or perform experiments on small toy networks which also make comparisons difficult. For this reason, in this section, we validate the method's performance by investigations of synthetic datasets. The presented method is compared with a plain MCMC method which does not incorporate the improvements.

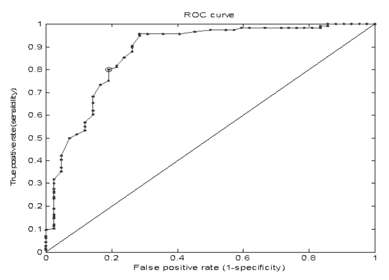
6.5.1 Experiments with Synthetic Dataset

For the work reported in this paper, three 40 gene synthetic networks were arbitrarily generated with sample size of 100. From the set of reconstructed networks using the guided GA [128, 130] approach, we choose the first 10 high scoring networks. The probability distribution is then estimated using the proposed MCMC method. For each of the 10 structures of the networks, samples from the probability distribution were obtained with MCMC, after discarding those from the *burn-in* phase. All simulations were repeated three times for different training data generated from synthetic networks. The results of experiments are summarized in Fig. 6.5. First we carry out single MCMC simulation runs instead of proposed multiple parallel MCMC runs.

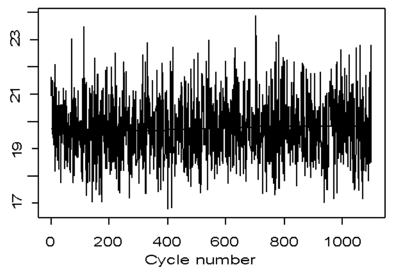
From the estimated probabilities, a set of all edges whose posterior probability exceeds a given threshold $\theta \in [0, 1]$ is taken for comparison with actual network. For a given threshold θ, we count the number of true positive (*TP*), false positive (*FP*), true negative (*TN*), and false negative (*FN*) edges. We then compute the *sensitivity* $= TP/(TP + FN)$, the *specificity* $= TN/(TN + FP)$, and the complementary *specificity* $= (1 - specificity) = FP/(TN + FP)$. Rather than selecting an arbitrary value for the threshold θ, we repeat this scoring procedure for several different values of $\theta \in [0, 1]$ and plot the ensuing sensitivity scores against the corresponding complementary specificity scores. This gives the *receiver operator characteristics* (ROC) curves of Fig. 6.5 (a). The diagonal dashed line indicates the expected ROC curve for a random predictor. The ROC curve of Fig 6.5 (a), top left, shows that we can recover more than 80% of the true edges at approximately zero FP rate. We note that the ROC curve corresponds to the network structure obtained based on the estimated probability distribution and not based on the network reconstructed by our earlier GGA causal modeling method. The MCMC trace plot between the objective function verses cycle number for 1000 cycles for the synthetic network of 40 genes is shown in Fig 6.5 (b). For this plot, the joint probability distribution is considered as the evaluation criterion after every run. The plot shows good mixing with a very low burn-in period. The same synthetic dataset is repeated on a plain MCMC simulation which does not incorporate the presented improvements. The trace plot of the plain MCMC simulation is shown in Fig 6.5 (c) for 1000 cycles. It is clearly evident that mixing is poor and has a longer burn-in period. Also the simulation is oscillating around sub-optimal values of the objective function while the proposed method quickly reaches the higher values of the objective function confirming that proposed method is better than the simple MCMC. The proposed method is repeated for 500 gene synthetic network dataset and its trace plot is shown in Fig 6.5 (d). This shows the method is easily scalable for thousands of gene as is the case of gene expression data at a comparatively feasible.

With sufficient improvements identified using single MCMC runs of the presented method over the plain MCMC method, we proceed to parallel MCMC runs as presented in Section 6.3. We obtained 3 different network structures for the same synthetic dataset using the GGA [128] search method and applied the network structures in a parallel MCMC runs with exchange of samples. Fig 6.5 (e) shows the trace plot of 3 parallel MCMC runs where each chain corresponds to an individual network. From the results, it was found that the auto-correlation between the samples produced from the 3 chains was far apart during the initial 1000 samples after which the correlation increased at 2000 cycles which is an indication of convergence. The parallel runs uncovered the entire probability distribution.

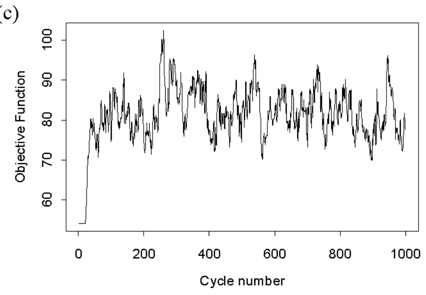
(a)



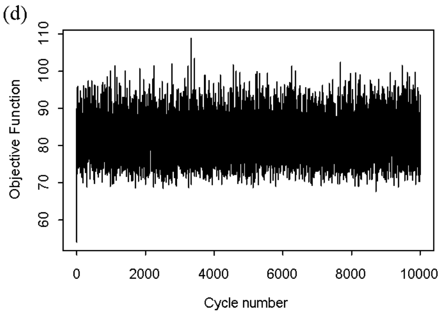
(b)



(c)



(d)



(e)

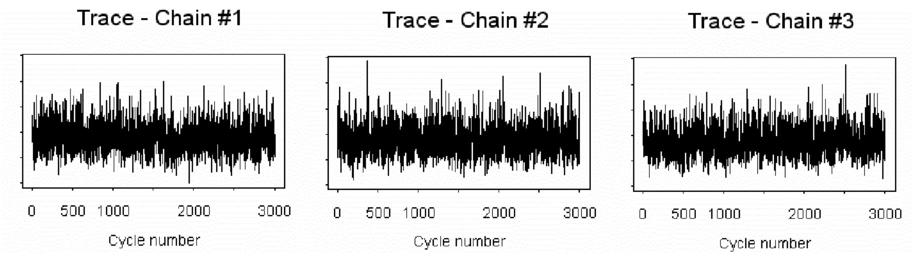


Fig. 6.5 Simulation results: (a) ROC plot of sensitivity versus (1-specificity) for a synthetic dataset MCMC simulation. (b) Trace plot of proposed MCMC method on synthetic network of 40 genes (c) Trace plot of plain MCMC method on synthetic network of 40 genes (d) Trace plot on synthetic network of 500 genes (e) Trace Plot of Parallel MCMC runs on 3 different network structures.

Although the experiments on synthetic data are successful, the time series of 100 gene expression measurements is significantly larger than what is usually available from real world wet lab experiments; hence we also test the approach using real yeast dataset [49].

6.5.2 Experiments with Real Dataset

To demonstrate the performance of the MCMC approach and also as a practical application of the method on a real biological dataset, we consider the yeast dataset [49] containing 800 genes and 77 samples comprising of a comprehensive catalogue of cell cycle-regulated genes in the yeast Saccharomyces cerevisiae. The dataset includes three long time course expression values representing three different ways of synchronizing the normal cell cycle, and five shorter time courses representing the altered environmental conditions. These results were combined with those by Cho *et al.* [125], to produce a more comprehensive collection of data. The test samples were synchronized so that all the cells would be at the same stage in their cell cycle. Using this data set, gene networks have already been reconstructed [25]. We also note that the Spellman dataset has classified the 800 genes in different phases of cell cycle such as G1, G2, S and M/G1 and G2/M. Using the MCMC based probability inference; the minimal I-map of the inferred yeast network is obtained. It is shown in Fig. 6.6.

From the reconstructed network shown in Fig. 6.6, we note the following interactions taking place which is in confirmation with the available literature [51].

- MBF (a complex of MBP1 and SWI6) and SBF (a complex of SWI4 and SWI6) controls the late G1 genes (e.g. CLN2 and NDD1).
- MCM1, together with FKH1 or FKH2, recruits the NDD1 protein in late G2 and controls the transcription of G2/M genes.
- SWI5 and ACE2 regulate genes at M/G1.
- MCM1 regulates SIC1 and ACE2.

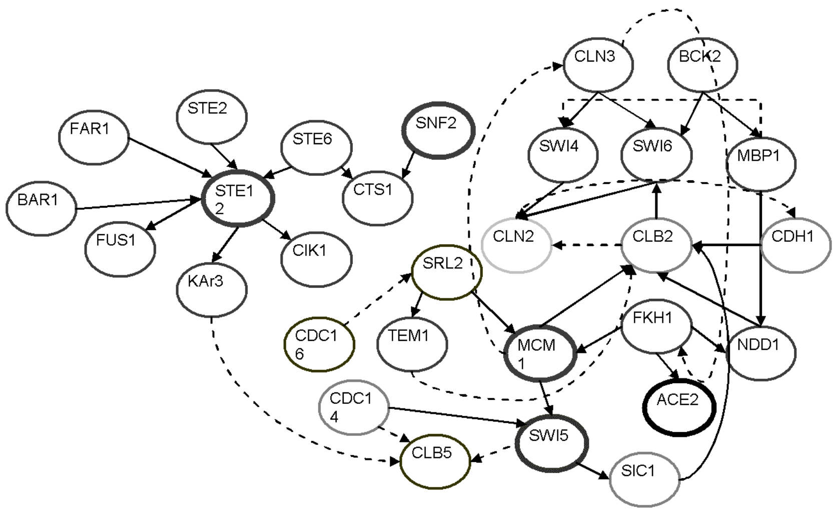


Fig. 6.6 Yeast Gene Regulatory network Minimal I-map.

6.6 Integration of Biological Data

From the yeast cell cycle data experiment, we observe that the minimal network derived using real life yeast dataset has more accurate reconstruction of regulatory interactions. However, due to the nature of the microarray data set, the resulting minimal GRN is not unique. Hence with integration of other related data such as sequence analysis in the form of prior probability, it is possible to recover unique minimal network which represents the underlying structure of gene expression. This work is presented in this section. Since biological integration is only possible on a real data, we have integrated information from sequence motifs and GO with the recovered yeast cell cycle network. Before we proceed to the experiments we briefly review the methods used to identify regulatory sequence motifs next.

6.6.1 Methods to identify regulatory sequence motifs

There are several methods to search over-represented motifs at the sequence upstream of co-regulated genes [27]. These approaches can roughly be categorized into two classes: word frequency based and probabilistic sequence models [131-137].

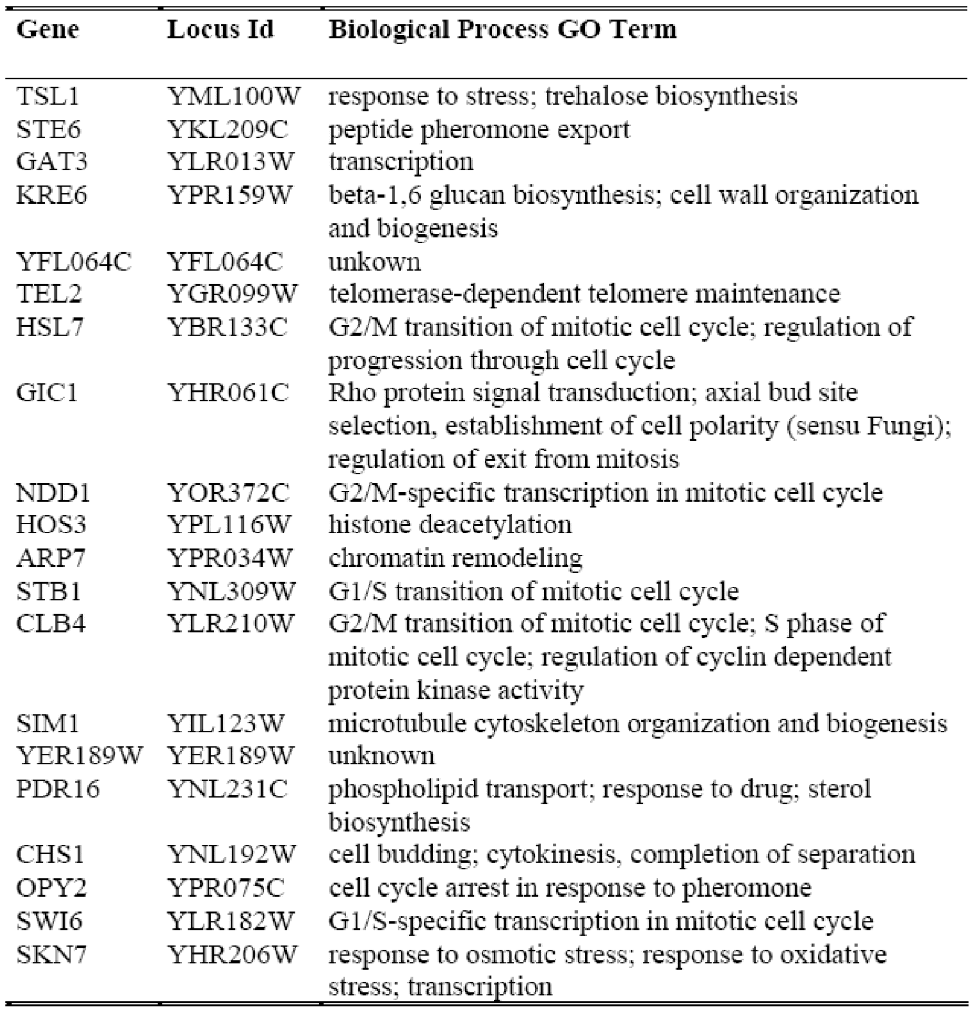
The frequency based methods are based on the frequency analysis of oligonucleotides in the upstream regions of co-regulated genes. The statistical significance of a site is calculated based on oligonucleotide frequency tables observed in all non-coding regions of the specific organism's genome. Usually, the length of oligonucleotide is varied from 4 to 9. Hexanucleotide (with oligonucleotide length equal to 6) analysis is most widely used. The identified significant oligonucleotides can be grouped as longer consensus motifs. The word counting based methods are not only simple and efficient but also rigorous (compared with heuristic methods) and exhaustive (all over-represented patterns of chosen length are detected). Price to pay is that it is limited to the detection of short and relatively conserved motifs and is not effective at identifying complex motif patterns.
For the probabilistic based methods, the motif is represented as a position probability matrix, Position Specific Scoring Matrix (PSSM), and the motifs are assumed to be hidden in the noisy background sequences. Maximum likelihood estimation is used to estimate model parameters. Heuristic methods, like Expectation Maximization (EM) [18, 138] and Gibbs sampling methods [127, 139, 140], are usually adopted to perform optimization. Actually Gibbs sampling is a stochastic equivalent of EM. One of the strengths of probabilistic based methods is the capability to identify motifs with complex patterns. Many potential motifs can also be identified, which actually is also a weakness, because it is difficult to distinguish the real one among them. Other limitations include: longer computational time, lack of unique solution due to the inherent randomness of the procedure, and the requirement of multiple runs. In this work, we will adopt the word frequency based method and use the online regulatory sequence analysis tool (BLAST). The analysis results show significant motifs tifs and consensus with a significant coefficient sig ≥ 0. sig $= 0$ means one expects one pattern to occur at random within each family. The increment of 1 for the significant coefficient sig represents a drop of 10 times in the occurrence

probability. A higher significant coefficient indicates a more significant motif. Next we look at the gene ontology data.

6.6.2 Gene Ontology

Gene Ontology (GO) [21, 26, 141] annotations are used for validation (and also describing their functions) of such genes with high scores (with 95% statistical significance) and belong-ing to any of the small networks. The GO comprises three orthogonal taxonomies corresponding to three domains: biological process (BP), molecular function (MF), and cellular component (CC). The work involves compiling annotations regarding the gene and its regulators in a tabular form. Below in Fig. 6.7 is a screen shot of a GO data.

Using this information, a search for keywords is carried out to find similarities between genes and their putative regulators predicted from the GRN as they may share common functionality. Approximately 80 percent of the regulatory relations were validated using the above method. Thus, it is shown that the proposed method has successfully inferred relations that are plausible and bio-logically significant. Next, we look at some of the well know yeast cell cycle networks available in the literature.



Gene	Locus Id	Biological Process GO Term
TSL1	YML100W	response to stress; trehalose biosynthesis
STE6	YKL209C	peptide pheromone export
GAT3	YLR013W	transcription
KRE6	YPR159W	beta-1,6 glucan biosynthesis; cell wall organization and biogenesis
YFL064C	YFL064C	unkown
TEL2	YGR099W	telomerase-dependent telomere maintenance
HSL7	YBR133C	G2/M transition of mitotic cell cycle; regulation of progression through cell cycle
GIC1	YHR061C	Rho protein signal transduction; axial bud site selection, establishment of cell polarity (sensu Fungi); regulation of exit from mitosis
NDD1	YOR372C	G2/M-specific transcription in mitotic cell cycle
HOS3	YPL116W	histone deacetylation
ARP7	YPR034W	chromatin remodeling
STB1	YNL309W	G1/S transition of mitotic cell cycle
CLB4	YLR210W	G2/M transition of mitotic cell cycle; S phase of mitotic cell cycle; regulation of cyclin dependent protein kinase activity
SIM1	YIL123W	microtubule cytoskeleton organization and biogenesis
YER189W	YER189W	unknown
PDR16	YNL231C	phospholipid transport; response to drug; sterol biosynthesis
CHS1	YNL192W	cell budding; cytokinesis, completion of separation
OPY2	YPR075C	cell cycle arrest in response to pheromone
SWI6	YLR182W	G1/S-specific transcription in mitotic cell cycle
SKN7	YHR206W	response to osmotic stress; response to oxidative stress; transcription

Fig. 6.7 Gene Ontology

6.6.3 Li *et al.* / Chen *et al.*'s yeast network

This section describes the Boolean model of the cell-cycle network in Saccharomyces cerevisiae developed by Li *et al.* [101]. Modifications to the GRN model carried out for its application for this thesis are discussed in the next section. There are 800 genes associated with the yeast cell cycle (Spellman *et al.*, [25]). It is, however, likely that the majority of these genes are controlled by a relatively small set of regulators [101]. The model of the yeast cell cycle under consideration involves a hand-constructed network of 11 key regulatory factors [101] (see Fig. 6.8 adapted from [101]). The constructed yeast cell cycle network consists of 12 nodes (the 11 regulatory factors identified and a 'cell size' signal node) and 30 links between network nodes. Links in the network have an associated weight of either +1 or −1, with a positive value indicating an excitatory interaction, and a negative value indicating an inhibitory interaction.

The model of Li *et al.* used different node functions to that of Chen *et al.*'s model [51]. Li *et al.*'s model used summative activation instead of the inhibition overrides activation paradigm used by Chen *et al*'s model [51], and also Li *et al.* assumed that node activations were maintained rather than Chen *et al*'s model [51] assumption that activation automatically decayed (described in detail below). In addition, while Chen *et al*'s model [51] customized differential equations used for individual nodes, all nodes in Li *et al.*'s network share the same node function. The node function in Li *et al.*'s model uses threshold summative activation based on the activation of input nodes and the weight of input connections. A positive summed input to a node gives an activation of 1, a negative summed input gives an activation of 0, and a zero-sum input maintains the activation of the node at the previous time step.

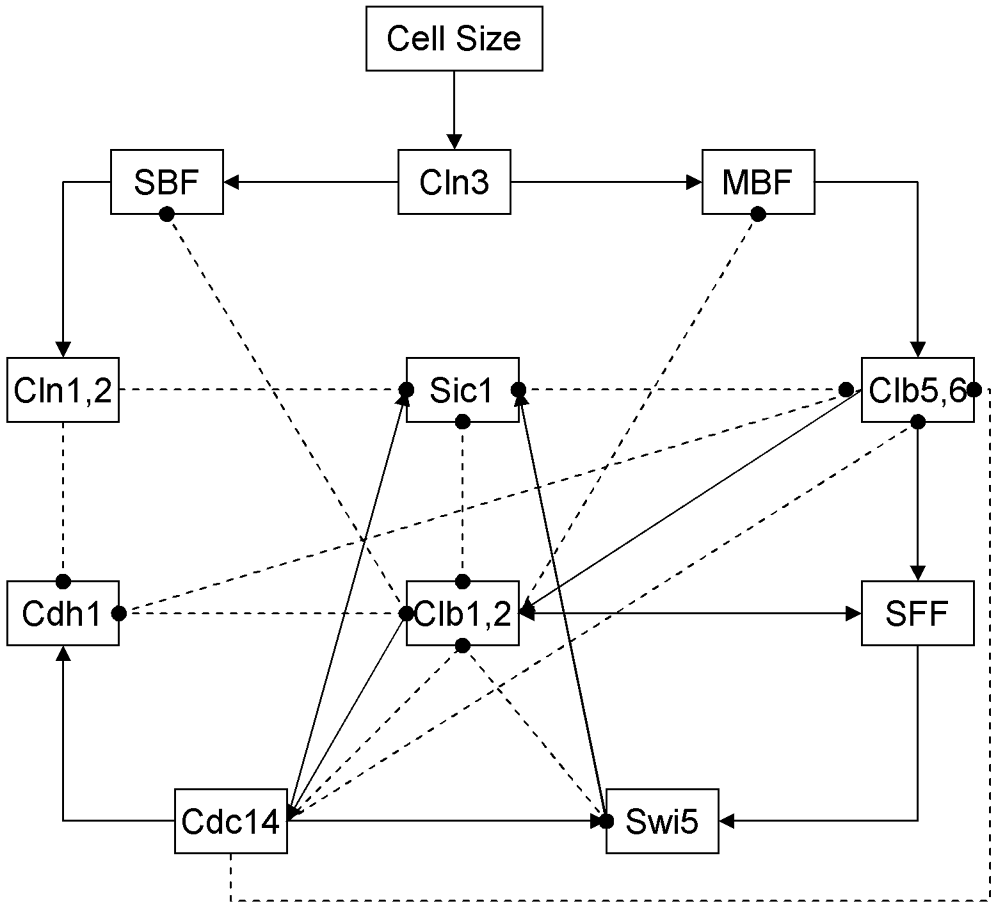


Fig. 6.8 Li *et al*'s yeast cell cycle network

6.6.4 Experiments and Results

We next show experiment results on integrating motif analysis and GO data to inferred budding yeast *S. Cerevisiae* cell cycle gene regulatory network shown in Fig. 6.9 below. While integration, we further observe four different types of regulations to be present in the inferred network:

- *Forward Activation (FA):* Gene X positively activates the subsequent gene Y (X +ve → Y)
- *Forward Inhibition (FI):* Gene X inhibits the subsequent gene Y (X -ve → Y)
- *Backward activation (BA):* Latter gene Y activates a former gene X (X ← +ve Y)

- *Backward inhibition (BI):* Latter gene Y inhibits the former gene X such that the former gene does not repeat (X 🡐 -ve Y).

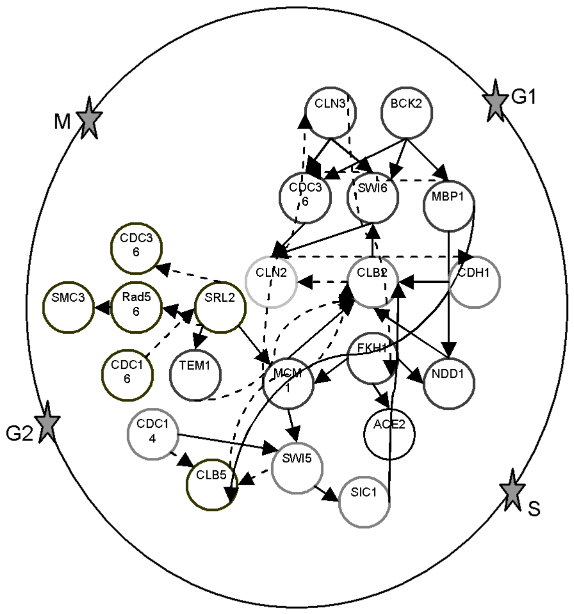


Fig. 6.9 The yeast Gene Network. Acronyms CLN1, SWI4 etc. denote various genes.

Brazhnik *et al.* [1] have emphasized that the interactions in any GRN are complicated actions occurring through more complex regulatory pathways involving the proteome and metabolome. The interactions observed in the GRN may (1) not necessarily correspond to direct physical interactions between the genes (2) be unknown and unverified. Thus the task of establishing a biological relevance of the recovered interactions tends to be quite difficult. For validating our approach, we choose a set of known genes identified by Spellman *et al.* [49] and which were later refined further by Gardner *et al.* [32, 101, 142] to result eventually in a set of 19 important genes. For validation, we will separate the GRN into 4 sub-networks (SN) based on the four phases (G1, M, S and G2). The sub-network1 (SN1) is simplest to validate since G1 phase involves cyclin genes which have well known simple interactions while SN3 is complex as it involves complicated spindle formation during the S phase.

The validation of the four SNs is carried out using

(i) interactions reported by Chen *et al.* [51] in which they used a set of differential equations (DE) to define the topology of the GRN containing 56 interactions that will serve as our first step of validation

(ii) known yeast cell cycle pathway diagram interpretation [51]

(iii) Motif data analysis and Gene Ontology data [51]

(iv) Saccharomyces Genome Database (SGD11) using software tools TRANSFAC, Entrez Gene [107, 131].

A. Integrating known interaction

First, we consider SN1 (Genes near G1 in Fig. 6.9) which is simplest of all sub-networks to validate as it essentially involves cell cycle genes of the type CLNx. We consider MBs of the genes from the inferred network of the G1 cyclin genes (CLN1, CLN2, CLN3, SWI4) which forms the basis for our verifying the G1 phase interactions. For example, from the inferred network, we find that SBF (Swi4, Swi6) and MBF (Swi6 and MBP1) are the G1-specific transcription factors (TF) that activate, directly and indirectly, genes in the G1 phase including the genes NDD1 and CLN2. This interaction is consistent with the observations of Chen *et al.* [51]. Further, in our inferred GRN, we observe that CLN3 and BCK2 act as parents of SBF and MBF TF genes, which in turn is indirectly causing CLN2 and CLB5 to forward activate (FA) or increase its expression. This interaction is also found to be in agreement with the pathway diagram given in [49]. Other interactions were also similarly found to be consistent with the previously reported results.

Further validation of G1 phase interactions

Further investigations of the validated interactions presented above reveals the following which are also in confirmation with the experimental observations of G1 phase.

CLN2 activates CLB2. CLB2 inactivates SBF and MBF which shuts off the G1/S events and goes to G2 phase.

CLb2 activates MCM1, FKH1 and NDD1. These three genes form a TF which activates SWI5 and SEC2.

SEC2 activate SIC1.

SIC1 inhibits CLB2. Hence, NDD1, MCM1 and FKH1 form a TF complex which regulates SWI5. FKH1 has similar expression as that of SWi5 while NDD1 and MCM1 do not correlate and thus are weak parents of SWI5. NDD1 and FKH1 have no regulatory relationship and are regulated by SWi5.

CLN2 interactions: An activated CLN2 gene initiates bud formation as stated in SGD. The CLN2 has a negative relationship (FI) with CDH1, where CDH1 has a FI relationship with CLB2.

Cell cycle interactions: The interactions CLN3– SIC1, CLN2 – SIC1, CLN2 and SWi5 – ACE2 and SBF/MBF and SFF find support in the work for regulatory relations reported by Forster *et al.* [143].

B. Delays

Inferring delays correctly leads to a very significant understanding of the GRN. Since one cell cycle takes about 120 minutes, we use 4 steps for time delays for a total of 30 minutes and number these delays as DL-1, DL-2, DL-3 and DL-4 with DL-0 indicating no delay. Some of the significant time delayed relationships that were inferred are as follows:

(i) No delay DL-0: MCM1 -> Swi5 -> SIC1

(ii) Delay DL-1: MCM1 -> CLB2

(iii) Delay DL-2: SWI5 -> SIC1

(iv) Delay DL4: SIC -> CLN3.

Further, we know that the genes involved in the initiation of S phase (i.e. SN3) are Sic1 and Clb5. We find a FA between Sic1 and Clb5 which also indicates the start of S phase. When Sic1 and Cdh1 act as parents (when they both are down regulated) of CLB2, we see that CLB2 begins to activate (FA) transcription factors Mcm1 and SFF. CLB2 has activation (FI) on SBF and simultaneously, MBF is also inactivated with the Clb5 expression level beginning to fall. Rise in CLB2 starts mitosis M phase. Further investigation of M phase is provided in the next section.

C. Investigating unknown interaction patterns

Let us now investigate in detail the Spindle formation (i.e. SN2) which is a complex system with many of its interactions unknown. We will show that our proposed technique is able to correctly recover many of the important features of the spindle formation. Some of these interactions reported here are hitherto unknown and hence the capability of the proposed approach to make these predictions may find significant applications in future GRN research.

<u>Known interactions</u>

The genes involved in this spindle formation are: MAD2, SRL 1, 2, 3, TEM1, MCM1, CDH1, BUB2, LTE1, CDC15, RAD53. These genes have been selected from the yeast dataset [49] and are those genes having spindle as keyword in their description. In brief, the interactions during the spindle formation assembly are as follows. As CLB2 is an important gene in switching off the S phase as mentioned in [51], we begin from this gene in exploring the spindle formation. When CLB2 activates Lte2, Lte2 in turn activates TEM1. When TEM1 is suppressed by Bub2 later, the spindle checkpoint is exited. However, when CLB2 activates CDC20, MAL2 suppresses CLB2, which indicates the spindle checkpoint is active, as otherwise the CLB2 will be activated and exit. When the spindle is active, gene SRL2 controls the activities of the regulatory proteins (TEM1 and MCM1). Hence SRL2 is the switch of the DNA checkpoint system, controlling the spindle formation and spindle checkpoint processes. Now, the interactions from our inferred GRN are Srl2 -> Tem1, Srl2 -> Mcm1, Srl2->cdc36, Cdc16->srl2, indirect interaction with Smc3 through Rad53, indirect interaction with CLB2 through MCM1 and TEM1 (see Fig.6.9).

Unknown interactions

Next, let us consider the unknown interactions. Due to the organization and interaction of the genes in the spindle formation assembly being very complex, existing reconstructions methods have been unable to capture and report these interactions. For our investigations, we look into two uncharacterized ORFs: YPR097C and YCL167A. Both these ORFs are classified as dubious in the Saccharomyces Genome Database [131] since their functions are unknown. However, we observe that both these genes were found to be directly connected to TEM1 and thus may also have been active during the formation process and the controlling of TEM1. Again, consider another interesting regulatory relationship between genes Cln3 with the two genes Cln2/1, Swi4 and also the interaction of Clb5/6 with the two other genes Tem1 and Cln1/2 predicted from the network. These unknown interactions, although not cited anywhere in the literature, indicate that gene TEM1 is active between spindle formation phase and the cell cycle phase.

Apart from this, there are some interactions which are not found in literature and hence have remained unexplained so far. However, from our inferred GRN, we observe these other additional interactions are statistically highly significant and biologically plausible (see the motif data analysis in Sec. 3.2.3 below). Since all our results are statistically highly significant and over 90% of network interactions recovered using the proposed method are accurately inferred, we can consider that these remaining inferences could also be biologically plausible. For example,

i) Genes Smc1, Smc3 acting as parents with FI (forward inhibition) for CLB2

ii) Genes Mcm2, Mcm6 of the DNA replication which is forward inhibiting the budding genes (Cln1, Cln2)

iii) Genes Pol2, Dbf4 having a FA relationship with Cln1/Cln2.

Further, a difference in the inferred direction of regulation involving gene CLB2 (shown in Fig. 6.9) compared with the direction given in the pathway diagram [51] may indicate the existence of some unproven phenomena happening in the network (e.g. a reverse cell cycle).

D. Motif data analysis

Motif data analysis involves finding conserved residues from DNA sequences commonly related to TF protein functions. Some of the methods used for motif data analysis are finding consensus sequences, alignment, position specific weight matrix and hidden Markov models (HMM). As next stage of investigation, we use the inferred GRN for motif data analysis. The motif data used in this study is obtained from a comparative genome analysis between distinct yeast species (phylogenetic shadowing) performed by Kellis *et al.* [131]. These motifs, available online as regular expressions, are transformed into their corresponding weight matrices by selecting, for each motif, the 20 Saccharomyces cerevisiae genes in which the motif was most reliably detected [131]. In our investigations, we validate the presence of common regulator motif only from the Markov blanket of four major genes Cln3, FKH2, SWi5 and CLB2 that includes well known transcription factors. The results are displayed in Table 6.1 in which the motif sequences of the genes are specified and also the putative regulators in which the motifs were found. For example, we can see that for the Cln3 gene, the regulator motif is ACCAGC and the target genes are SW15, CLN2, SWE1, CLB6, CDC46. Hence, the predicted unknown interactions involving genes Cln3 and CLB2 are validated on the basis of motif pattern analysis.

Table 6.1 Motif Data Analysis

Gene	Regulator motif	Target Genes
Cln3	ACCAGC	SWI5 CLN2, SWE1, CLB6, CDC 46
Fkh2	GTAAACA	CLB2,SWI5,CDC2, SPO1
SWI5	ACCAGC	SIC1, CLN3
CLB2	TTACCNAATTNGGTAA; GTMAACAA	MCM1,SWI5, CDC46, (-CLN2), SFF

It is also significant to investigate the binding site which is a region on DNA to which specific TF proteins form a chemical bond to initiate gene regulation. To extract the binding sites, we restrict the search upto 500 bp upstream sequence. The 500 bp promoter sequences of all genes are downloaded from SGD (Saccharomyces Genome Database). For each of these sequences, we search for transcription factor (TF) binding sites using the PATCH software, a part of the TRANSFAC [131] having a library of known TF binding sites. Initially, all 532 binding site motif patterns available in TRANSFAC were used for motif finding. After removing redundant and rare binding sites, the total binding sites is reduced 354. Using PATCH, counts of motif occurrences in the promoter regions of the 19 select genes were obtained based on which we get 10 motifs that have high frequency of occurrence. Using this set of 10 motifs, our motif analysis is carried out next.

Many of the small sub-networks (other than the 4 SNs mentioned earlier) within the reconstructed GRN also contain genes that are uncharacterized or characterized as dubious. Our studies indicate that these networks can either interact with each other or remain autonomous. Gene Ontology (GO) annotations are used for validation (and also describing their functions) of such genes with high scores (with 95% statistical significance) and belonging to any of the small networks. The GO comprises three orthogonal taxonomies corresponding to three domains: biological process (BP), molecular function (MF), and cellular component (CC). The work involves compiling annotations regarding the gene and its regulators in a tabular form. Using this information, a search for keywords is carried out to find similarities between genes and their putative regulators predicted from the GRN as they may share common functionality. Approximately 80 percent of the regulatory relations were validated using the above method. Thus, it is shown that the proposed method has successfully inferred relations that are plausible and biologically significant.

6.7 Summary

Estimation of parameters, which involves estimation of conditional probability distributions (CPD), is an important aspect of GRN modeling. In this chapter, a new Markov chain Monte Carlo approach using Gibbs sampling is presented for estimating the conditional probability distribution underlying gene regulatory network structures. The approach is novel as it performs the rank ordering of genes based on the Markov Blanket scoring metric, applies parallel Markov chains using different high scoring starting network structures and clamps genes which are higher in the order for faster and efficient convergence. Rather than initializing the Markov chains with randomly chosen networks, our GGA proposed in previous chapter is used to generate the high scoring initial networks and the probability distribution. Both synthetic and real world yeast cell data sets have been applied in the investigations. The experiment on synthetic data set shows that the proposed technique performs significantly better than the standard MCMC algorithm for estimating probability distributions of the genes in the network. From the yeast cell cycle data experiment, we observe that the minimal network derived using real life yeast dataset has more accurate reconstruction of regulatory interactions. However, due to the nature of the microarray data set, the resulting minimal GRN is not unique.

However, the integration of other related data such as regulatory motif sequence analysis and GO data in the form of prior probability allowed us to recover a unique minimal network which represents the underlying structure of gene expression data.

Gene Ontology annotation results show very significant GO Biological Processes in these networks reconstructed. Detailed interactions and highly connected genes can be identified as a future work. A similar situation occurs for the integration of regulatory sequence analysis which helped in significant regulatory sequence motifs to be identified. By combining motif information with the genetic network, we were able to obtain biological explanations which are in agreement with the available literature. In the next chapter, we summarize the work reported in the thesis and also provide future direction of research.

Chapter 7

7 Conclusion

7.1 Conclusion

Gene regulatory network is the interconnection between the genes and its transcription factor genes. Since the high-throughput data acquisition technology for gene expression measurement known as the biological microarray technology emerged in the late 1990s, applications of computational intelligence techniques to microarray data analysis have drawn attention of the bioinformatics community. Microarrays allow the monitoring of expression levels of thousands of genes simultaneously and the data provide the basis to discover gene regulation networks, life evolution, and other important bio-problems. Modeling of gene regulatory networks is a challenging task because gene expression microarray data is characterized as massive, heterogeneous (high dimension), NP-hard, stochastic, and has irregular sampling rate and also has measurement errors (leading to noisy data). Due to the nature of data, its analysis is apparently beyond the ability of traditional analysis methods as they are not able to capture many of the system intricacies, e.g. the time-varying dependencies between the different genes in the gene regulatory network. Further, researchers are faced with enormous quantity of data in which lies important hidden information including, e.g. the transcription factor activity profiles.

Many methods presented for inference of gene networks have focused on statistical methods, such as, Bayesian networks [17], dynamic Bayesian networks [42], relevance networks[30] and graphical models [73, 83, 92, 144-146]. Graphical models have emerged as powerful tools for learning, description and manipulation of conditional independencies among the genes. However, these approaches overlooked many important issues including, e.g. time delays, direction and sign of regulation, interactions amongst parents, children and spouse genes in a biological regulation system. Further, other important issues for accurate GRN modeling, e.g. suitable search techniques to explore the astronomically large space of networks, parameter learning of GRNs was not investigated in greater detail..

This thesis has made efforts to address these problems as follows:

- *Causal model learning framework based on the Bayesian network:* The proposed method considers various GRN modeling aspects, namely network structure, direction of regulation, time delay and sign/orientation of regulation (i.e. up/down regulation). The method decomposes the entire network into sub-models based on Markov Blanket of each gene. We observe that, in terms of accuracy, robustness, noise and scalability, the proposed method is superior to the widely used simple Bayesian network model and the Graphical Gaussian model. The computational increase due to the scalability is dealt effectively by the new constraint minimization technique. Further, unlike the traditional Bayesian network, the proposed framework uses continuous rather than discrete data values. A set of experiments using the synthetic datasets were carried out by varying topology, interaction types, noise levels, time delay of the interactions and so on to see the effect on accuracy of reconstruction. From the results, it is evident that modeling technique has very low spurious relationships in the reconstructed network even in the presence of noise.

- *Path analysis post processing method:* Due to stochastic nature and high dimensionality of the microarray data, a large number of regulation relationships (from 60% to 80%) can be found to be false positive or spurious. As a part of the overall causal model design of GRN, the proposed post processing step, incorporating d-separation, alternative causal hypothesis and time delay as its tools, greatly improves the accuracy of the inferred network by pruning the network of the false positives. Experiments with the synthetic and real life yeast networks have shown overall accuracy of the inferable network structure improved by up to 40%. The performance of method is maintained even with network parametric variations, namely, network topologies, structure, delays and noise.
- *Synthetic Data generation:* The network generator system presented in this thesis generates synthetic datasets based on causal modeling approach for GRN reconstruction. The proposed system can generate four different network topologies, namely scale free, small world, random and hierarchical. Further, the generated synthetic network is made realistic by incorporating complex network characteristics such as transmission delays, biological and experimental noise. The datasets permit large scale experimental investigations. These datasets are generated for rigorous evaluation of methodologies presented in this thesis. The system also allowed comparisons between different methods, namely, GGM and conventional BN techniques.
- *Guided Genetic Algorithm technique (GGA):*. The GGA approach employs a diversity switching to adaptively alternate between the ordinary operators and the guided operators. An important characteristic of GGA approach is that it combines a ranking schema; multiple path constraint and standard Gaussian function to perform high level heuristic operations. Experiments using artificial data and real yeast data demonstrate the superior performance of

GGA approach compared with the commonly used simple GA approach. Based on the tested data sets, the experiments revealed the folowing interesting features:

  - GGA approach generates superior quality of GRN
  - Fitness evaluations reduced by approximately 25% compared with the simple GA resulting in reduced computational time
  - Diversity measure allows the search to escape local optima.

- *Frequently occurring Markov Blanket Genetic Algorithm (FOMBGA):* The proposed FOMBGA technique avoids the necessity of specifying set of standard GA parameters (crossover, mutation and selection) by replacing these operations with a process of probability estimation and sampling. The entire FOMBGA search technique is successfully evaluated by the synthetic dataset and its superiority established over both the SGA and GGA. As an application of the technique, we have also investigated the well understood yeast cell cycle data set and examined various known and unknown gene interactions that were found to be in agreement with the information from well known datasets.
- *Parameter estimation using MCMC:* The new Markov chain Monte Carlo approach using Gibbs sampling proposed for estimating the conditional probability distribution underlying gene regulatory network structures performs the rank ordering of genes based on the Markov Blanket scoring metric, applies parallel Markov chains using different high scoring starting network structures and clamps genes which are higher in the order for faster and efficient convergence. Rather than initializing the Markov chains with randomly chosen networks, our GGA is used to generate the high scoring initial networks and the probability distribution. Both synthetic and real world yeast cell data sets have been applied in the investigations. Synthetic data set experiments show that proposed MCMC technique performs significantly better than the standard MCMC algorithm. From the yeast cell cycle data experiment, we observe that the minimal network derived using

real life yeast dataset has more accurate reconstruction of regulatory interactions compared to previously reconstructed using FOMBGA method.

- *Biological data Integration:* Due to the nature of the microarray data set, the resulting minimal GRN is not unique. To overcome this, we integrate gene regulatory sequence information with genetic network inference to obtain a unique network. Based on the results for parameter estimation using Markov chain Monte Carlo results , all major cell cycle related motifs were identified. By combining the genetic networks with the promoter information corresponding to the motifs, we obtained a reasonable biologically plausible gene regulatory network.

7.2 Future Work

The major contributions of this thesis, such as the Markov blanket based causal model, the GGA and FOMBGA search techniques and fixing parameters for MCMC based parameter learning, are investigated for the first time in the context of GRN reconstruction. Aside from the synthetic data sets, real yeast cell cycle data set has been investigated to establish the advantages of these concepts over existing techniques. The research presented in this thesis can be presented in several directions.

- *Increasing model complexity*: Although the proposed model is more sophisticated than other state of the art models, it is possible to enhance it further by additional information. A gene regulation and expression systems are complicated. For example, proteins or metabolites can be included as hidden variables to improve the model accuracy. Further, methods to include feedbacks can also be studied.
- *Improving FOMBGA*: As we have seen, the MB approach had a better performance than the single arc approach for constructing the probability vector. It will be worthwhile to investigate whether the Markov blankets can be further combined with one another resulting in small sub-networks which

can then be utilized for generating the probability vector. This may not only improve the accuracy but also the computational speed.

- *Improving computational speed by implementation on grid:* Like all bioinformatics computations, experiments for modeling GRN require excessively massive computational power, specially if the order of genes in a genome reach hundreds of thousands. Further, the procedure of inferencing parameters using MCMC is computationally expensive. Hence, a grid based computing algorithm can be investigated to further improve the speed of computations.
- *Integrating additional biological information:* Currently, data integration is one of the major tasks of system biology. Efforts to integrate genomics (genome sequence), transcriptomics (microarray), proteomics and metabolomics data, effect of non coding RNA on regulation and related prior knowledge can further enhance the accuracy of the reconstructed GRN.

References

[1] P. Brazhnik, A. de la Fuente, and P. Mendes. , “Gene networks: how to put the function in genomics,” *Trends Biotechnol,* vol. 20, pp. 467-472, 2002.

[2] P. Mendes, Sha, W., and Ye, K., “Artificial gene networks for objective comparison of analysis algorithms,” *Bioinformatics* vol. 19, pp. 122–129, 2003.

[3] H. Bolouri, and E. H. Davidson, “Modeling transcriptional regulatory networks,” *Bioessays,* vol. 24, no. 12, pp. 1118-29, 2002.

[4] D. P. S. Verma, “Temporal and Spatial Regulation of Plant Genes,” *Springer-Verlag. Vienna,* 1988.

[5] S. Bay, Chrisman,L., Pohorille,A., and Shrager,J., “Temporal aggregation bias and inference of causal regulatory networks,” *J Comp Biol,* vol. 11, pp. 971-85, 2004.

[6] T. Akutsu, S. Miyano, et al., “Algorithms for Inferring Qualitative Models of Biological Networks,” in Pacific Symposium on Biocomputing 5, Hawaii, 2000.

[7] E. Segal, Shapira,M., Regev,A., Pe’er,D., Botstein,D., Koller,D., and Friedman,N., “Module networks: identifying regulatory modules and their

condition-specific regulators from gene expression data," *Nat Genet.,* vol. 34, pp. 166–176, 2003.

[8] J. W. Stucki, "Stability analysis of biochemical systems. A practical guide.," *Progress Biophys. Mol. Biol.,* vol. 33, pp. 99–187, 1978.

[9] T. Chu, Glymour,C., Scheines,R., and Spirtes,P., "A statistical problem for inference to regulatory structure from associations of gene expression measurements with microarrays," *Bioinformatics,* vol. 19, pp. 1147–1152, 2003.

[10] H. de Jong, "Modeling and simulation of genetic regulatory systems: a literature review," *J Comput Biol* vol. 9, no. 1, pp. 67-103, 2002.

[11] R. Thomas, Thieffry H. et al., "Dynamical behaviour of biological regulatory networks--I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state," *Bull Math Biol,* vol. 57, no. 2, pp. 247-76, 1995.

[12] J. J. Tyson, and Othmer, H. G., *The Dynamics of Feedback Control Circuits in Biochemical Pathways.* , New York: Academic Press, 1978.

[13] A. P. a. M. B. E. Gasch, "Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. ," *Genome Biol,* vol. 3, no. 11, pp. 59, 2002.

[14] P. D'Haeseleer, S. Liang, et al. , "Gene expression analysis and modeling," in Pac Symp Biocomput(Tutorial), 1999.

[15] E. P. Van Someren, L. F. Wessels, et al., "Genetic network modeling.," *Pharmacogenomics* vol. 3, no. 3, pp. 507-25, 2002.

[16] M. a. J. H. Wahde, "Coarse-grained reverse engineering of genetic regulatory networks," *Biosystems,* vol. 55, no. 1-3, pp. 129-36, 2000.

[17] N. Friedman, Linial,M., Nachman,I., Pe'er,D., "Using Bayesian networks to analyze expression data," *J Comp Biol,* vol. 7, pp. 601-620, 2000.

[18] J. E. Bailey, "Lessons from metabolic engineering for functional genomics and drug discovery," *Nat Biotechnol.,* vol. 17, no. 7, pp. 616-8, 1999.

[19] R. Blanco, "Learning Bayesian Networks from Data with Factorisation and Classification Purpose, Applications in Biomedicine," Department of Computer Science and Artificial Intelligence University of the Basque Country, Spain, 2005.

[20] L. Hood, J. R. Heath, et al., "Systems biology and new technologies enable predictive and preventative medicine," *Science* vol. 306, no. 5696, pp. 640-3, 2004.

[21] F. Al-Shahrour, R. Diaz-Uriarte, et al., "FatiGO: a web tool for finding significant associations of Gene Ontology terms with groups of genes," *Bioinformatics,* vol. 20, no. 4, pp. 578-80, 2004.

[22] J. D. Hughes, P. W. Estep, et al., "Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae," *J Mol Biol* vol. 296, no. 5, pp. 1205-14, 2000.

[23] M. Kamvysselis, *Computational comparative genomics: genes, regulation, evolution,* vol. 100, Massachusetts Institute of Technology, Cambridge, MA, 2003.

[24] S. A. Kauffman, “Metabolic stability and epigenesis in randomly constructed genetic nets,” *J Theor Biol,* vol. 22, no. 3, pp. 437-67, 1969.

[25] A. Arkin, J. Ross, et al., “Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected Escherichia coli cells,” *Genetics,* vol. 149, no. 4, pp. 1633-48, 1998.

[26] H. M. Blake JA, *The Gene Ontology Project: Structured vocabularies for molecular biology and their application to genome and expression analysis,* New York, N.Y: Wiley & Sons, Inc., 2003.

[27] M. Tompa, N. Li, et al. , “Assessing computational tools for the discovery of transcription factor binding sites,” *Nat Biotechnol,* vol. 23, no. 1, pp. 137-44, 2005.

[28] I. Foster, *The Grid: Blueprint for a new computing infrastructure*: Morgan Kaufmann, 2004.

[29] H. Kitano, “Systems biology: a brief overview,” *Science,* vol. 295, no. 5560, pp. 1662-4, 2002.

[30] R. A. Nachman I, Friedman N, “Inferring quantitative models of regulatory networks from expression data,” *Bioinformatics,* vol. 20, pp. I248-I256, 2004.

[31] M. J. Beal, Falciani, F., Ghahramani, Z., Rangel, C. and Wild, D. L., “A Beyesian approach to reconstructing genetic regulatory networks with hidden factors,” *Bioinformatics,* vol. 21, no. 3, pp. 349-356, 2005.

[32] U. Güldener, Münsterkötter, M., Kastenmüller, G., et al., “CYGD: the Comprehensive Yeast Genome Database,” *Nucleic Acids Research,* vol. 33, pp. D364-D368, 2005.

[33] J. Schäfer, and Strimmer,K., “An empirical Bayes approach to inferring large-scale gene association networks,” *Bioinformatics,* vol. 21, pp. 754-764, 2005.

[34] M. Swain, Hunniford, T, Dubitzky, W, Mandel, J and Palfreyman, N, “Reverse-Engineering Gene-Regulatory Networks Using Evolutionary Algorithms And Grid Computing,” *Journal of Clinical Monitoring and Computing,* vol. 19, pp. 329-337, 2005.

[35] W. Luo, Hankenson, K.D., and Woolf, P.J., “Learning transcriptional regulatory networks from high throughput gene expression data using continuous three-way mutual information,” *BMC Bioinformatics,* vol. 2008, no. 9, pp. 467, 2008.

[36] A. J. Butte, and Kohane,I.S., “Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements,” in Pac. Symp. Biocomput., 2000, pp. 418–429.

[37] P. D'haseleer, Liang,S., Somogoyi,R., “Genetic network inference: from co-expression clustering to reverse engineering,” *Bioinformatics,* vol. 16, pp. 707-726, 2000.

[38] M. B. Eisen, Spellman,P.T., Brown,P.O. and Botstein,D., “Cluster analysis and display of genome-wide expression patterns,” *Proc. Natl Acad. Sci. USA,* vol. 95, pp. 14863-14868, 1998.

[39] T. Akutsu, S. Miyano, et al., "Identification of Genetic Networks from a Small Number of Gene Expression Patterns Under the Boolean Network Model," in Pacific Symposium on Biocomputing 4, Hawaii, 1999.

[40] S. Liang, S. Fuhrman, et al., "REVEAL, A general reverse engineering algorithm for inference of genetic network architectures," in Pacific Symposium on Biocomputing 3, Hawaii, 1998.

[41] S. Kim, S. Imoto, et al., "Dynamic Bayesian network and nonparametric regression for nonlinear modeling of gene networks from time series gene expression data," *Biosystems,* vol. 75, no. (1-3), pp. 57-65, 2004.

[42] K. Murphy, and Mian,S., *Modelling gene expression data using dynamic Bayesian networks*, University of California, Berkeley., 1999.

[43] K. P. Murphy, "Dynamic Bayesian Networks: Representation, Inference and Learning," Computer Science Division, University of California, Berkeley, 2002.

[44] B. E. Perrin, L. Ralaivola, et al., "Gene networks inference using dynamic Bayesian networks," *Bioinformatics* vol. 19, no. Suppl 2, pp. 138-148, 2003.

[45] M. Zou, and Conzen, S. D., "A new dynamic Bayesian network (DBN) approach foridentifying gene regulatory networks from time course microarray data," *Bioinformatics,* vol. 21, no. 1, pp. 71-9, 2005.

[46] A. de la Fuente, Bing,N., Hoeschele,I., Mendes,P., "Discovery of meaningful associations in genomic data using partial correlation coefficients," *Bioinformatics,* vol. 20, pp. 3565-3574, 2004.

[47] P. D'Haeseleer, *Reconstructing Gene Networks from Large Scale Gene Expression Data,* vol. 207, The University of New Mexico, Albuquerque, NM,, 2000.

[48] R. D. Shachter, & Peot, M. A., "Simulation approaches to general probabilistic inference on belief networks," *Uncertainty in Artificial Intelligence,* vol. 5, pp. 221-231, 1989.

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

[50] A. Arkin, P. Shen, et al., "A test case of correlation metric construction of a reaction pathway from measurements," *Science,* vol. 277, no. 29, pp. 1275-1279, 1997.

[51] K. C. Chen, Calzone,L., Csikasz-Nagy,A., Cross,F.R., Novak,B., Tyson,J.J., "Integrative analysis of cell cycle control in budding yeast," *Mol Biol Cell,* vol. 15, pp. 3841-62, 2004.

[52] T. Chen, H. L. He, et al., "Modeling gene expression with differential equations," in Pac Symp Biocomput, 1999, pp. 29-40.

[53] Y. a. G. M. C. Cheng, "Biclustering of expression data," in Proc Int Conf Intell Syst Mol Biol 2002, pp. 93-103.

[54] H. Toh, and Horimoto,K., "Inference of a genetic network by a combined approach of cluster analysis and graphical Gaussian modeling," *Bioinformatics,* vol. 18, pp. 287–297, 2002.

[55] R. Ram, Chetty, M. and Dix,T.I., , "Fuzzy Model for Gene Regulatory Networks," in IEEE Congress on Evolutionary Computation, 2006, pp. 1450-1455.

[56] J. A. Dickerson, Z. Cox, et al., "Creating Metabolic and Regulatory Network Models using Fuzzy Cognitive Maps," in North American Fuzzy Information Processing Conference (NAFIPS), Vancouver, B.C., 2001.

[57] J. K. Dickerson, "Virtual Worlds as Fuzzy Cognitive Maps.," *Presence* vol. 3, no. (2, Spring), pp. 173-189, 1994.

[58] W. Woolf, "A fuzzy logic approach to analyzing gene expression data," *Physiol Genomics,* vol. 3, no. 1, pp. 9-15, 2000.

[59] N. Friedman, "Inferring cellular network using probabilistic graphical models," *Science,* vol. 33, pp. 799-805, 2004.

[60] S. I. de Hoon, K. Kobayashi, N. Ogasawara, and S. Miyano, "Inferring Gene Regulatory Networks From Time-Ordered Gene Expression Data Of Bacillus Subtilis Using Differential Equations," *Pacific symposium on computation biology,* vol. 8, pp. 17-28, 2003.

[61] Y. S.Watanabe, Y. Eguchi, D. Tominaga, and M. Okamoto, "Algorithms for Inference of Genetic Networks AIGNET," in Intl. Workshop on Genome Informatics, 1998, pp. 274-275.

[62] D. Goldberg, *Genetic algorithms*: Addison-Wesley, 1989.

[63] J. Cheng, Bell, D.A. and Liu, W., "An algorithm for Bayesian belief network construction from data." pp. 83-90.

[64] N. Friedman, and Koller, D., "Being Bayesian about network structure: A Bayesian approach to structure discovery in Bayesian networks," *Machine Learning,* vol. 50, pp. 95-126, 2003.

[65] C. S. Kim, "Bayesian Orthogonal Least Squares (BOLS) algorithm for reverse engineering of gene regulatory networks," *BMC Bioinformatics,* vol. 8, pp. 251, 2007.

[66] D. Heckerman, "A Tutorial on Learning with Bayesian Networks. In Learning in Graphical Models," M. Jordan, ed., Cambridge , MA: MIT Press, 1999.

[67] J. Pearl, *Causality: Models, Reasoning and Inference*: Cambridge University Press, Cambridge, UK, 2000.

[68] a. I. T. C.F. Aliferis, *Algorithms for Large-Scale Local Causal Discovery in the Presence of Small Sample or Large Causal Neighborhoods,* Vanderbilt University, 2002.

[69] I. T. C.F. Aliferis, A. Statnikov, *HITON, A Novel Markov Blanket Algorithm for Optimal Variable Selection,* Vanderbilt University 2003.

[70] A. de la Fuente, P. Brazhnik, et al., "Linking the genes: inferring quantitative gene networks from microarray data," *Trends Genet* vol. 18, no. 8, pp. 395-8, 2002.

[71] G. Bejerano, "Efficient exact p-value computation for small sample, sparse and surprising categorical data," *J. Computational Biology,* vol. 11, no. 5, pp. 867-886, 2004.

[72] P. Giudici, and Green, P., "Decomposable graphical Gaussian model determination," *Biometrika* vol. 86, no. 4, pp. 785-801, 1999.

[73] P. Giudici, P. Green, and C. Tarantola, *Efficient model determination for discrete graphical models*, Univ. Pavia, Italy, 2000.

[74] N. Guelzim, Bottani,S., Bourgine,P. and Kepes,F., "Topological and causal structure of the yeast transcriptional regulatory network," *Nat. Genet.*, vol. 31, pp. 60–63, 2002.

[75] P. M. Magwene, and Kim,J., "Estimating genomic coexpression networks using first-order conditional independence," *Genome Biol*, vol. 5, pp. R100, 2004.

[76] B. Shipley, *Cause and Correlation in Biology: A User's Guide to Path Analysis, Structural Equations and Causal Inference*: Cambridge University Press, Cambridge, UK., 2002.

[77] P. Sprites, Glymour,C. and Scheines,R., *Causation, Prediction, and Search: Adaptive Computation and Machine Learning*, 2 ed.: MIT Press. Cambridge, Massachusetts., 2001.

[78] J. Whittaker, *Graphical Models in Applied Multivariate Statistics*, Chichester, England: Wiley, 1990.

[79] J. Pearl. a. T. S. Verma., "A theory of inferred causation" *Principles of Knowledge Representation and Reasoning*, pp. 441–452,, 1991.

[80] C. Wallace, *Statistical and inductive inference by minimum message length*: Springer, 2005.

[81] D. M. Chickering, "Learning Equivalence Classes of Bayesian-Network Structures," *Journal of Machine Learning Research*, vol. 2, pp. 445- 498, 2002.

[82] J. Cheng, and W. Liu. , "An algorithm for Bayesian network construction from data," *Artificial Intelligence and Statistics*, 1997.

[83] D. Madigan, S. Andersson, M. Perlman, and C. Volinsky, "Bayesian model averaging and model selection for Markov equivalence classes of acyclic graphs," *Communications in Statistics: Theory and Methods*, vol. 25, pp. 2493-2519, 1996.

[84] C. Wallace, Korb, K. B. and dai, H, "Causal Discovery Via MML," in Proceedings Of The 13th International Conference On Machine Learning (ICML 96), 1996, pp. 516-524.

[85] E.Cinquemani, S.Summers, J.Lygeros, "Stochastic dynamics of genetic networks: modelling and parameter identification," *Bioinformatics*, vol. 24,, no. 23, pp. 2748-2754, 2008.

[86] H. H. McAdams, Arkin, A, "Stochastic mechanisms in gene expression," *Proc. Natl. Acad. Sci. USA*, vol. 94, pp. 814-819, 1997.

[87] A. Wagner, "Estimating coarse gene network structure from large-scale gene perturbation data," *Genome Res*, vol. 12, no. 2, pp. 309-15, 2002.

[88] M. Henrion, "Practical issues in constructing a Bayes belief network," *Int. J. Approx. Reasoning*, vol. 2, no. 3, pp. 337, 1988.

[89] A.J. Hartemink, T.S. Jaakkola, and R.A. Young, "Using Graphical Models and Genomic Expression Data to Statistically Validate Models of Genetic

Regulatory Networks," *Pacific Symposium on Biocomputing (PSB),* vol. 6, pp. 422-423, 2001.

[90] T. I. Lee, N. J. Rinaldi, et al., "Transcriptional regulatory networks in Saccharomyces cerevisiae," *Science* vol. 298, no. 5594, pp. 799-804, 2002.

[91] J. Yu, P. Wang, A.J. Haremink, and E.D. Jarvis, "Advances to Bayesian Network Inference for Generating Causal Networks from Observational Biological Data," *Bioinformatics,* vol. 20, no. 18, pp. 3594-3603, 2004.

[92] Y. Madigan, "Bayesian graphical models for discrete data," *International statistical Review,* vol. 63, pp. 215-232, 1995.

[93] A. Shinohara, Iida,K., Takeda,M., Maruyama,O., Miyano,S., and Kuhara,S., "Finding Sparse Gene Networks," *Genome Inf.,* vol. 11, pp. 249–250, 2000.

[94] Z. Bar-Joseph, Gerber, G.K., Lee, T.I., Rinaldi, N.J., Yoo, J.Y., Robert, F., Gordon, D.B., Fraenkel, E., Jaakkola, T.S., Young, R.A., Gifford, D.K., "Computational discovery of gene modules and regulatory networks," *Nat Biotechnol.,* vol. 21, no. 11, pp. 1295-7, 2003.

[95] Z. Bar-Joseph, "Analyzing time series gene expression data," *Bioinformatics,* vol. 20, no. 16, pp. 2493-503, 2004.

[96] S. Ando. and. H. Iba, "Inference of gene regulatory model by genetic algorithms," in Conference on Evolutionary Computation, 2001, pp. 712-719.

[97] G. M. Dasika, C.D. Maranas, and J.D. Varner, "A Mixed Integer Linear Programming (MILP) Framework for Inferring Time Delay in Gene Regulatory Networks," in Pacific Symposium on Biocomputing, 2004, pp. 474-485.

[98] R. Ram, Chetty,M., and Dix,T.I., "Causal Modeling of Gene Regulatory Network." pp. 1-8.

[99] R. Ram, and Chetty,M., "Learning Structure of Gene Regulatory Networks." pp. 525-531.

[100] M. M. Mano., *DIGITAL DESIGN,* 3 ed.: Prentice Hall, Inc., 2002.

[101] F. Li, Long, T., Lu, Y., Ouyang, Q., and Tang, C., "The Yeast Cell-cycle Network is Robustly Designed," *PNAS,* vol. 101, no. 14, pp. 4781-4786, 2004.

[102] P. a. R. e. Erd¨os, A., "On random graphs," *Publ Math. Debrecen,* vol. 6, pp. 290-297, 1959.

[103] A. L. a. A. Barabasi, R., "Emergence of scaling in random networks.," *Science,* vol. 286, pp. 509-512, 1999.

[104] K. L. Calvert, Doar,M.B. and Zegura,E.W., "Modeling Internet topology," *IEEE Communications Magazine,* vol. 35, pp. 160-163, 1997.

[105] J. Rahmel, "SplitNet: A Dynamic Hierarchical Network Model," *AAAI/IAAI,* vol. 2, pp. 1404, 1996.

[106] R. Ram, and Chetty M., , "Generating Synthetic Gene Regulatory Networks," *Lecture Notes in Bioinformatics,* vol. 5265, pp. 237-249, 2008.

[107] T. S. Gardner, et al., "Inferring genetic networks and indentifying compoud mode of action via expression profiling," *Science,* vol. 301, pp. 102–105, 2003.

[108] H. Kishino, and Waddell,P.J., “Correspondence analysis of genes and tissue types and finding genetic links from microarray data,” in Genome Inform Ser Workshop Genome Inform., 2000, pp. 83–95.
[109] D. Goldberg, *Design of Innovation*: Springer, 2002.
[110] P. Larranaga, and Lozano, J. A. , *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*: Kluwer Academic Publishers, 2002.
[111] T. Back, *Handbook on Evolutionary Computation* IOP Publishing Ltd and Oxford University Press, 1997.
[112] P. Larranaga, Y. Yurramendi, R.H. Murga, C.M.H Kuijpers, “Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters,” *IEEE Trans Pattern Anal Mach Intell,* vol. 18, no. 9, pp. 912-926, 1996.
[113] W. Lam, K.S Leung, P.S. Ngan, “Discovering probabilistic knowledge from databases using evolutionary computation and minimum description length principle,” in Genetic programming, 1998, pp. 786-794.
[114] D. Koller, “Toward optimal feature selection,” in International Conference on Machine Learning, Bari, Italy, 1996, pp. 284-292.
[115] J. Yu, P.P. Wang, A.J. Hartemink, and E.D. Jarvis, , “Advances to Bayesian network inference for generating causal networks from observational biological data,” *Bioinform.,* vol. 20, pp. 3594—3603, 2004.
[116] H. Shimodaira, “A Diversity Control Oriented Genetic Algorithm (DCGA),” in Genetic and Evolutionary Computation Conference (GECCO), 1999, pp. 603-611.
[117] M. Wineberg, “The Shifting Balance Genetic Algorithm: Improving the GA in a Dynamic Environment,” in Genetic and Evolutionary Computation (GECCO), 1999, pp. 504-510.
[118] A. Dobra, B. Jones, J.R. Nevins, and M. West., “Sparse graphical models for exploring gene expression data,” *J. Multiv. Analysis,* vol. 90, pp. 196-212, 2004.
[119] J. Hertz, “Statistical issues in reverse engineering of gene regulatory networks,” in Pacific Symposium on Biocomputing, 1998.
[120] V. R. I. a. P. O. B. J. L. DeRisi, “Exploring the metabolic and genetic control of gene expression on a genomic scale,” *Science,* vol. 278, pp. 680-686, 1997.
[121] "The complete dataset is available on the internet at http://cmgm.stanford.edu/pbrown/explore/array.txt."
[122] J. S. T. A. Chambers, C. Stanway, A. J. Kingsman and S. M. Kingsman, “Transcriptional control of the Saccharomyces cerevisiae PGKgene by RAP1,” *Journal on Molecular Cell Biology,* vol. 9, pp. 5516-5524, 1989.
[123] S. C. Wang, “Reconstructing Genetic Networks From Time Ordered Gene Expression Data Using Bayesian Method With Global Search Algorithm,” *Journal Of Bioinformatics And Computational Biology,* vol. 2, no. 3, pp. 441-458, 2004.

[124] J.H. Tyson, et al, "The Swi5 transcription factor of Saccharomyces cerevisiae has a role in exit from mitosis through induction of the Cdk-inhibitor Sic1 in telophase," *Genetics*, vol. 145, pp. 85-96, 1997.

[125] R. J. Cho, et al. , "A genome-wide transcriptional analysis of the mitotic cell cycle," *Mol. Cell*, vol. 2, pp. 65-73, 1998.

[126] J. S. Liu, "Monte Carlo Strategies in Scientific Computing," *Springer-Verlag. Berlin, Heidelberg*, 2001.

[127] W. R. Gilks, and Wild, P., "Adaptive Rejective Sampling for Gibbs Sampling," *Applied Statistics*, vol. 41 pp. 337-348, 1992.

[128] R. Ram, and Chetty,M., "A Guided genetic algorithm for Gene Regulatory Network," in IEEE Congress on Evolutionary Computation, Singapore, 2007, pp. 3862-3869.

[129] R. Ram, and Chetty M.,, "Constraint Minimization for Efficient Modeling of Gene Regulatory Network," *Lecture Notes in Bioinformatics*, vol. 5265, pp. 201 - 213, 2008.

[130] R. Ram, and Chetty M., , "A Markov blanket based Probabilistic Genetic Algorithm for Causal Reconstruction of Gene Regulatory Networks," *BioSystems Special Issue on Evolving Gene Regulatory Networks, Elsevier (submitted)*, 2009.

[131] M. Kellis, et al., "Sequencing and comparison of yeast species to identify genes and regulatory elements," *Nature*, vol. 423, pp. 241–254, 2003.

[132] F. P. Roth, J. D. Hughes, et al. , "Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mRNA quantitation," *Nat Biotechnol.*, vol. 16, no. 10, pp. 939-45, 1998.

[133] J. Van Helden, B. Andre, et al., "Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies," *J Mol Biol*, vol. 281, no. 5, pp. 827-42, 1998.

[134] J. Van Helden, B. Andre, et al., "A web site for the computational analysis of yeast regulatory sequences," *Yeast*, vol. 16, no. 2, pp. 177-87, 2000.

[135] J. Van Helden, A. F. Rios, et al., "Discovering regulatory elements in non-coding sequences by analysis of spaced dyads," *Nucleic Acids Res*, vol. 28, no. 8, pp. 1808-18, 2000.

[136] J. Van Helden, "Regulatory sequence analysis tools," *Nucleic Acids Res* vol. 31, no. 13, pp. 3593-6, 2003.

[137] J. Van Helden, "Metrics for comparing regulatory sequences on the basis of pattern counts.," *Bioinformatics*, vol. 20, no. 3, pp. 399-406, 2004.

[138] T. L. a. C. E. Bailey, "Fitting a mixture model by expectation maximization to discover motifs in biopolymers," in Proc Int Conf Intell Syst Mol Biol, 1994, pp. 28-36.

[139] S. Geman, and Geman, D. , "Stcohastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721-741, 1984.

[140] G. Thijs, M. Lescot, et al., "A higher-order background model improves the detection of promoter regulatory elements by Gibbs sampling," *Bioinformatics* vol. 17, no. 12, pp. 1113-22, 2001.

[141] M. Ashburner, and Lewis, S.,, "On ontologies for biologists: the Gene Ontology - uncoupling the web," in In Silico Biology Novartis Found Symp, 2002, pp. 66-80; discussion 80-3, 84-90, 244-52.
[142] B. Futcher, "Transcriptional Regulatory Networks and the Yeast Cell Cycle," *Current Opinion in Cell Biology,* vol. 14, pp. 676-683, 2002.
[143] I. Famili, J. Forster, et al., "Saccharomyces cerevisiae phenotypes can be predicted by using constraint-based analysis of a genome-scale reconstructed metabolic network.," *Proc Natl Acad Sci U S A* vol. 100, no. 23, pp. 13134-9, 2003.
[144] D. Edwards, *Introduction to Graphical Modelling*: Springer, 2000.
[145] R. Edwards, H. T. Siegelmann, et al., "Symbolic dynamics and computation in model gene networks," *Chaos,* vol. 11, no. 1, pp. 160-169, 2001.
[146] S. L. Lauritzen, *Graphical models*, Oxford: Oxford University Press, 1996.

Appendix 1

A1 Yeast Cell Cycle Dataset

In this thesis we investigate the well known budding yeast S.Cerevisiae data set for the study of eukaryotic cell cycle in which majority of yeast cells divide every 120 minutes under suitable conditions.

The eukaryotic cell division cycle consists of four phases: G1, S, G2, and M as shown in Fig. A1.1. The two major steps in cell division are DNA replication (S phase) and mitosis (M phase). These two steps are separated by gap phases G1 and G2. G1 phase is a period during cell cycle when the major part of the cell grows and the bud emerges. The ending of G1 phase is indicated by the bud emergence. There is a point when the cell becomes irrevocably committed to entering the S (Synthesis) phase and traversing the rest of the cycle. The major event in the S phase is chromosome replication. At the end of S phase, each chromosome has two identical DNA double helix molecules. The S phase is followed by G2 phase when the bud gets larger as the cell continues to grow resulting in a part of nucleus gradually migrating into the daughter cell. This is then followed by M (mitosis) phase, where a series of events happen: spindle formation, chromosome segregation, nuclear division, and eventually cytokinesis which cause the daughter cell to finally separate from the mother cell. The daughter cell which is smaller than its mother must grow to a consider-able size before it can make any attempt to divide. Both mother and daughter cells remain in the G1 phase while growing, although it takes mother cells a shorter time to reach a size compatible with cell division.

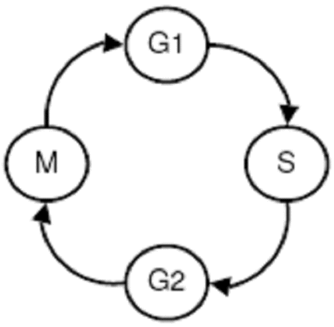


Fig. A1.1 Cell cyle

A gene whose expression level varies periodically with the cell cycle can be considered to be cell cycle regulated. However, not all such genes are functionally involved in mechanisms of the cell cycle, nor is it possible to say with certainty that all genes involved in the cell cycle necessarily display periodic behaviour [24]. The cell cycle has been studied extensively by molecular biologists and is relatively well understood. Spellman et al. [25] identify approximately 800 putative cell cycle regulated genes in Saccharomyces cerevisiae based on analysis of microarray time-series data. A scoring method is also devised to assess whether a gene is cell cycle regulated based on Fourier analysis and Pearson correlation coefficients. A score threshold was selected, whose value if exceeded by 91% of known cell cycle genes, were thus classified all those genes that scored above this threshold as cell cycle regulated. The resulting 800 genes (out of approximately 6200 genes) were clustered hierarchically by Spellman *et al*. From these 800 genes, Spellman et al. then identified and analyzed nine functionally related clusters of genes. The microarray hybridization data used consists mainly of four independent time-series datasets, which is referred to as alpha-factor, cdc15, cdc28, and elutriation. The cdc28 dataset, coming originally from Cho *et al.* [24], has each time-series measuring the mRNA transcript levels for the same set of genes, but synchronized using a different method. It must be noted that yeast cultures must be synchronized so that all cells are at the same point in the cell cycle before transcript levels are measured. Significant synchrony was achieved for one to three cell cycles, depending on the method. The number of time points in each time series varied from 14 to 24. In addition to the four time courses, Spellman et al. examined the response of genes to the cyclins Cln3p and Clb2p, two known cell cycle regulators. More than half of the 800 putative cell cycle regulated genes responded to at least one of these cyclins. The genes were clustered using the hierarchical clustering method described by Eisen *et al.* [26], and nine clusters were identified empirically. The genes in each cluster were functionally related and substantially co-regulated based on analysis of the promoter regions. For each cluster, Spellman et al. identified known and hypothesized binding sites for possible regulators, and described any significant regulatory effects of Cln3p and Clb2p. In addition, they discussed which genes take part in major functions of the cell cycle, such as DNA replication, budding, glycosylation, and nuclear division.