A Computational Approach to Reconstructing Gene Regulatory Networks.

Xutao Deng

A Computational Approach to Reconstructing Gene Regulatory Networks

A Thesis

Presented to the

Department of Computer Science

and the

Faculty of the Graduate College

University of Nebraska

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

University of Nebraska at Omaha

by

Xutao Deng

August, 2003

THESIS ACCEPTANCE

Acceptance for the faculty of the Graduate College, University of Nebraska, in partial fulfillment of the requirements for the degree Master of Science, University of Nebraska at Omaha.

Committee

Hesham Ali

[illegible]

[illegible]

Chairperson [illegible]

Date 10/17/2003

A Computational Approach to Reconstructing Gene Regulatory Networks

Xutao Deng, MS

University of Nebraska, 2003

Advisor: Hesham Ali

ABSTRACT

Motivation: Many modeling frameworks have been applied to infer regulatory networks from gene expression data sets. Linear Additive Models (LAMs), as one large category of models, have been gaining more and more popularity. One problem associated with this kind of models is that the system is often under-determined because of excessive number of unknown parameters. In addition, the practical utility of these models has remained unclear.

Methods: Based on LAMs, we developed an improved method to infer gene regulatory networks from time-series gene expression data sets. The method includes an incremental connectivity model with indexed regulatory elements and a linear time complexity fitting algorithm embedded with genetic algorithm. Comparing to previous LAMs, where a fully connected model is used, the new technique reduces the number of parameters by *O(N),* therefore increasing the chance of recovering the underlying regulatory network. The fitting algorithm increment the connectivity during the fitting process until a satisfactory fit is obtained.

Results: We performed a systematic study to explore the data mining ability of LAMs. A guideline to use LAMs is provided: If the system is small (3-20 elements), more than 90% regulation pathways can be correctly determined. For large scale system, either a clustering is needed or it is necessary to integrate other information besides expression profile only. Coupled with clustering method, we applied our method to Rat Central Nervous System development (CNS) data with 112 genes. We were able to efficiently generate regulatory networks with statistically significant pathways which have been previously predicted.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor Prof. Hesham Ali for what he showed me as an advisor and as a person. There are millions of great things about him: kind, supportive, patient, good sense of humor, great taste of life, high academic standard, etc. What I have been learning from him is not just about research but about a decent and fruitful life. I was (am) privileged to be his student.

I also thank my committee members: Prof. Zhengxin Chen and Prof. Bruce Chase. They provided me many valuable comments and encouragements. This thesis will look different without their help.

Thanks to BIIG group members, Mark, Rong, Xiaolu, Alexander, Dan, Yifeng, and Matt, for being patient audience for a long time. Special thanks to Dr. Mark Pauley for his kind help in proofreading and revising.

Finally, I owe a lot to my family members: my wife Huimin for her constant love and support; my son Zachary, for refreshing my mind; and my parents for their unconditional help.

Table of Contents

Table of Figures

Chapter 1: Gene Expression

The difference between man and monkey is gene regulation.

-- *Leroy Hood*

This thesis is structured as the following: In chapter 1, I introduce some basic biological concepts about gene regulatory networks and techniques to profile gene expression. These concepts and data underlie the model of a real gene regulatory network system. In chapter 2, a thorough literature review and analysis of current modeling and simulation methods applied to the study of gene regulatory networks will be given. An introduction of clustering methods is also included in this chapter. The first two chapters serve a purpose of introduction. In chapter 3, a new modeling method are presented and explained. This method consists of two parts: an incremental model and a fitting algorithm. A theoretical analysis of this method is also provided. In chapter 4, the method is implemented and the performance is tested. Results from application to real expression data sets are also provided.

1.1 A Little Bit Biology

As the Human Genome Project is almost complete, biology is marching into post-genomics era. Research focus is also moving from biological sequence analysis to functional genomics analysis. Every gene in a genome plays a role like a node in a network. The functional study of genes and their relationships is indispensable to answer some basic questions such as development, evolution, diseases *etc.* However, these efforts are poorly made compared with sequence analysis. Some questions we would like to ask are:

- What are the functional roles of different genes?
- How are genes regulated?
- How do genes and their products interact?
- How does gene expression level differ in various cell types and diseases conditions?

This chapter introduces the basics of gene regulatory networks and experimental methods to measure and survey gene expression. Before we dive into gene expression data analysis and gene regulatory networks, I should give a brief review of biology background.

The central dogma (Figure 1) of molecular biology states that genetic information is stored in DNA, transcribed to messenger RNA (mRNA), and then translated into proteins. This picture is significantly augmented by the fact that certain proteins (transcription factors) can bind to DNA and regulate transcription. These mechanisms form the back bone of gene regulatory networks. Biology system is notoriously complex. In the human body, almost all cells have entire copy of genomic DNA. Yet this same genetic information yields a large number of different cell types. The fundamental difference between a neuron and a liver cell, for example, is which genes are expressed. Human genome contains about 30,000 genes. There are about 5,000 different proteins expressed in every cell, where there are more than 200 different cell types and more than 10^{14} cells in the entire body.

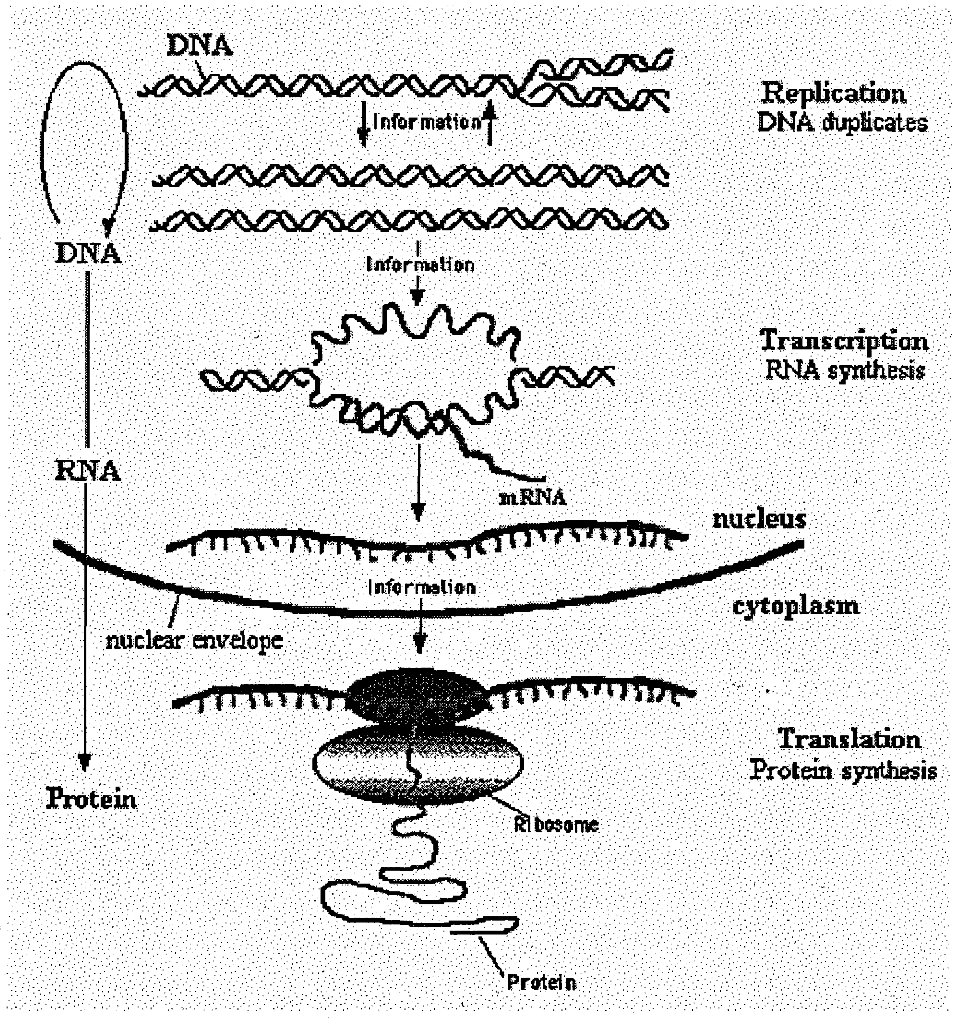


Figure 1 Central Dogma of Molecular Genetics

DNA, RNA and proteins have different functions. DNA is the molecular storehouse of genetic information. When cells divide, the DNA is replicated, so that each daughter cell maintains the same genetic information as the mother cell. RNA acts as an intermediate from DNA to proteins. Only a single copy of DNA is present in a cell, but multiple copies of the same piece of RNA may be present, allowing cells to make huge amount of protein. In eukaryotes, DNA is found in the nucleus only. RNA is copied in the nucleus and then moved out of nucleus, where it is transcribed into proteins. Proteins are specialized machines, each of which fulfills it own task, which may be transporting oxygen, catalyzing reactions, or responding to extra cellular signals etc. One of the most interesting functions a protein may have is binding directly or indirectly to DNA to perform transcriptional regulation, thus forming a closed feedback loop of gene regulation.

Transcription: Transcription is a complicated set of events that leads from DNA to messenger RNA. Usually a gene consists of a coding region and a regulatory region. The coding region is the part of gene that encodes a certain protein. The regulatory region is the part of the DNA that contributes to the control of the gene. In particular, it contains binding sites for transcription factors, which act by binding to the DNA (directly or with other transcription factors in a small complex) and affecting the initiation of transcription. The regulatory region varies in size from 10 to 100,000 bases and contains binding sites for single or multiple transcription factors. Transcription factors may act either positively or negatively; that is, an increase in the amount of transcription factor may lead to either more or less gene expression, respectively. Another input mechanism is phosphorylation or dephosphorylation of a bound transcription factor by other proteins.

Transcription factors are sometimes called *trans-regulatory elements*, and the DNA sites where transcription factors bind are called *cis-regulatory elements*. The collection of *cis*-regulatory elements upstream of the coding region can be called the *promoter*.

Splicing: In prokaryotes, the coding region is contiguous, but in eukaryotes the coding region is typically split up into several parts. Each of these coding parts is called an *exon*, and the parts in between the exons are called *introns*. In eukaryotes, transcription occurs in the nucleus, translation occurs in the cytosol. Between transcription and translation, the mRNA must be moved physically from inside the nucleus to outside. As part of this process, the introns are edited out, which is called *splicing*. In some cases, there are alternative splicings, that is, the same stretch of DNA can be edited in different ways to form different proteins. At the end of splicing process, or directly after the transcription process in prokaryotes, the mRNA is in the cytosol and ready to be translated.

Translation: In the cytosol, mRNA binds to ribosomes, complex macromolecules whose function is to create proteins. A ribosome moves along the mRNA three bases at a time and each three-base combination, or *codon*, is translated into one of the 20 amino acids.

Post-translational Modification: The function of the ribosome is to copy the one dimensional structure of mRNA into a one-dimensional sequence of amino acids. As the process goes on, the one-dimensional sequence of amino acids folds up into a final three-dimensional protein structure.

More Steps: Several more steps are possible in the flow of information from DNA to a protein. First, proteins may be modified after they are translated. Proteins may agglomerate. In particular, many transcription factors bind to DNA in a multimeric state. DNA is a stable molecule, but mRNA and proteins are constantly being degraded by cellular machinery and recycled. Specifically, mRNA is degraded by a ribonuclease (RNase), which competes with ribosomes to bind to mRNA. Proteins are degraded by cellular machinery, including proteasomes signaled by ubiquitin tagging. Protein degradation is regulated by a variety of more specific enzymes, which may differ from one protein target to another.

1.2 Gene Expression Measurement

As we have seen, gene expression is a complex process and is regulated in every step from DNA to its products. Knowing the gene transcript abundance in various tissues, developmental stages and under various conditions is important for understanding gene functions and relationships. Although mRNA is not the ultimate product of a gene, it makes sense to use mRNA level to monitor gene expression level because transcription is the first step in gene regulation. Moreover, the measurement of mRNA levels currently is considerably cheaper and can be done in a more high-throughput way then direct measurements of the protein levels. Many methods have been developed for detecting mRNA levels and identifying the differentially expressed gene. Other methods, such as mass spectrometric identification of gel-separated proteins, allow the state of a cell to be characterized on the proteomic level [Kahn 1995].

1.2.1 DD-PCR (Differential Display PCR)

Through the arbitrary amplification and comparison of different mRNA sources, DD-PCR allows identification of differentially expressed genes in various in vitro and in vivo systems. The key element of the method is to use a set of oligonucleotide primers to amplify messenger RNA 3' termini. One primer is anchored to polyadenylate tail of a subset of mRNAs and the other is a short primer with an arbitrary sequence so that it anneals at different position related to the first primer. The amplified cDNAs labeled with radioisotope are then distributed on a denaturing polyacrylamide gel and visualized by autoradiography. Side-by-side comparison of mRNA species from two or more related samples allows identification of both up- and downregulation genes of interest [Liang and Pardee].

1.2.2 DNA Microarray

Microarray is one of the latest breakthroughs in experimental molecular biology, which allow monitoring of gene expression for tens of thousands of genes in parallel. Terminologies that have been used in the literature to describe this technology include, but not limited to: biochip, DNA chip, DNA microarray, and gene array. In order to study gene expression in DNA microarray technology, mRNA is hybridized to a high-density array of immobilized target sequences, each corresponding to a specific gene. The mRNAs being sampled are labeled with fluorescent dyes and are then hybridized to the array where each mRNA will quantitatively hybridize to its complementary target sequence. The expression level of a particular gene can be visualized by observing the fluorescence at each spot of the array. If two differently labeled mRNAs are used then one can quantitatively compare gene expression level in both samples. There are two main variants of the DNA microarray technology.

1. cDNA array: probe cDNA (500~5,000 bases long) 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. This method is widely considered as developed at Stanford University [Schena *et al.*, 1995][Ekins and Chu 1999].

2. Oligonucleotide array: an array of oligonucleotide (20~80-mer oligos) or peptide nucleic acid (PNA) probes is synthesized either *in situ* (on-chip) or by conventional synthesis followed by on-chip immobilization. The array is exposed to labeled sample DNA, hybridized, and the identity/abundance of complementary sequences is determined. Due to the combinatorial nature of the process, very large numbers of mRNAs can be probed at the same time. This method, "historically" called DNA chips, was developed at Affymetrix, Inc. , which sells its photolithographically fabricated products under the

GeneChip® trademark. Many companies are manufacturing oligonucleotide based chips using alternative in-situ synthesis or depositioning technologies.

1.2.3 RT-PCR (Reverse Transcriptase PCR)

To measure gene expression using RT-PCR, the mRNA is first reverse-transcribed into cDNA, and the cDNA is then amplified to measurable levels using PCR. Using built-in calibration techniques, RT-PCR can achieve high accuracy coupled with an exceptional sensitivity. The method does require PCR primers for all the genes of interest, and is not inherently parallel like the previous methods, so automation is crucial to scale up.

1.2.4 SAGE (Serial Analysis of Gene Expression)

SAGE uses a very different technique for measuring mRNA levels. First, double stranded cDNA is created from the mRNA. A single 10 base pair (long enough to uniquely identify each gene) “sequence tag” is cut from specific location in each cDNA. Then the sequence tags are concatenated into a long double stranded DNA which can then be amplified and sequenced. This method has two advantages: the mRNA sequence does not need to be known a prior so that it can also detect previously unknown genes and it uses sequencing technology that many labs already have.

By measuring transcription levels of genes in an organism under various conditions, at different developmental stages and in different tissues, we can build up ‘gene expression profile’ which characterizes the dynamic behavior of each gene in the genome. The gene expression profile is often formatted in a matrix with rows representing genes, columns representing samples (e.g. various tissues, developmental stages, and treatments), and each cell containing a number characterizing the expression level of the particular gene in the particular sample. We will call such a table a *gene expression matrix.*

1.3 Functional Genomics

"Specifically, *functional genomics* refers to the development and application of global (genome-wide or system-wide) experimental approaches to assess gene function by making use of the information and reagents provided by structural genomics. It is characterized by high throughput or large scale experimental methodologies combined with statistical and computational analysis of the results." [Hieter and Boguski 1997]

Biology is rapidly evolving into a data-rich field, opening up the possibility of data-driven research, rather than hypothesis-driven research. Unfortunately, current computational techniques are too humble to handle the flood of large scale expression data. How to design reasonable computational technique to make the most use of biological data is a key issue in the research of functional genomics. And those answers can certainly lead our understandings of ourselves into a new stage. In this thesis, I will focus on the design of quantitative model for gene regulatory networks inference by using expression data (gene expression matrix). Hopefully, this thesis can help to answer some of the questions proposed in the beginning of this chapter.

Chapter 2: Related Works

Biologists can be divided into two classes: experimentalists who observe things that can not be explained, and theoreticians who explain things that cannot be observed.

-- Aharon Katzir-Katchalsky

2.1 Introduction

In this section, I will introduce the basics of modeling, reverse-engineering and its application in genetic regulatory networks. I will also introduce the concept of data mining and its relationship with modeling and. In the rest of this chapter, several current models will be introduced and discussed in detail.

2.1.1 Modeling and Simulation

"*The phrase 'modeling and simulation' designates the complex of activities associated with constructing models of real world systems and simulating them on a computer*" [Ziegler, 1976]

From this definition, it is obvious we are mostly concerned with three elements: Real system, Model and Computer; two relationships: Modeling and Simulation. Their relationships can be clearly illustrated in Figure 2.

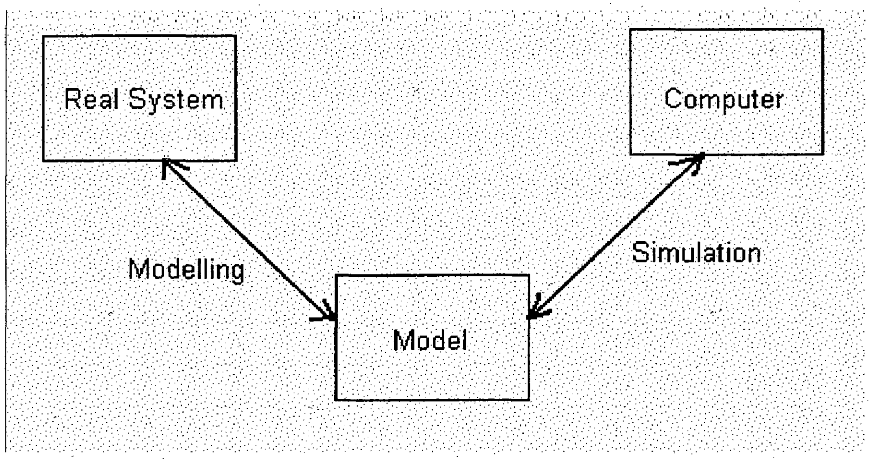


Figure 2 Relationships of Modeling and Simulation

By a *real system*, we mean some part of the real world, which is of interest. As a general rule, we can say that a real system is a *source of behavior data.* Any variable in the system can be plotted against time. This set of plot can be used to describe the behavior of the system. A *model* is a simplified representation of a real system intended to enhance our ability to understand, predict, and possibly control the behavior of the system. Models are often expressed in certain common ways such as differential equations or graphs. All these forms of model description provide instructions to generating data. Given a model, *computer* is often used to carry out the model instructions suitably encoded as a program.

Having looked at the three elements, let us consider the two relationships. The *modeling* relation concerns the *validity* of the model, that is, how well the model represents the real system. There are degrees of strengths for this validity. The model is *replicatively valid* if it matches data already acquired from the real system. Stronger than this is the condition in which the model is *predictively valid*, that is, when it can match data *before* data are acquired from the real system. A third, stronger level of validity concerns the relation between the structure of the model and the internal workings of the real system. A model is *structurally valid* if it not only reproduces the observed real system behavior, but truly reflects the way in which the real system operates to produce this behavior. The *simulation* relation concerns the faithfulness with which the computer carries out the instructions intended by the model, that is, number of bugs. The faithfulness with which a program realizes a model is often referred to as the correctness of the program. Simulation is often used to suggest approximate model, run the model and determine the validity of the model.

The reasons of modeling and simulation are:

- *Tractability* — it is too time-consuming or expensive to play with the real system to answer "what if" questions. Examples are car crash model and cosmos evolution model.
- *Training purposes*—such as military project or business game for managers.

- *Black Box problems* – when trying to figure out what is going on inside a poorly understood system. Examples are gene regulatory networks. This kind of model is like a discovering tool and is the focus of this thesis.

2.1.2 Quantitative models

Model is an abstraction or a prototype of a thing, a system, or a relationship. Quantitative models generally include one or more parameters. For example, if we plot people's age against height, we may find the dots roughly form a curve. If we model age and height as a linear relationship (such as using least square method), the intercept and slope are parameters need to be found. If the model fit data at an accepted level, the model may be used to describe reality or make predictions about further as yet unobserved data. If the model cannot be accepted, we may dump the model, or make a more complex model to fit the data (such as quadratic model instead of linear), or add application constraints to the model (such as limiting age from 0 to 18). Model building, fitting, and testing are three non-divisible processes in model construction. Once we formulated the model, we will use some techniques to optimize its parameters and we will see how the model behave compared with real data. Several model fitting methods are commonly used: analytical optimization (closed form mathematical methods, such as Linear Algebra, Least Square Methods, and Ordinary Differential Equations), iterative Hill-climbing techniques (try each parameter at a time iteratively) and meta-heuristic, such as artificial neural network and genetic algorithms. I will give more detailed discussion on the three methods in next chapter.

2.1.3 Modeling Gene Regulatory Networks

As Katzir-Katchalsky indicated, there is an obvious gap between experimental biologist and theoretical biologist. The apparent complexity of molecular and cellular interactions currently being studied will increasingly require modeling tools that can be used to properly design and interpret biological experiments. To bridge the gap between experiments and theories, formal methods for modeling and simulation of gene regulation processes are indispensable. As most genetic regulatory systems of interest involve many genes connected through interlocking positive and negative feedback loops, an intuitive understanding of their dynamics is hard to obtain. Using formal methods, the structure regulatory systems can be described unambiguously, while predictions of their behavior can be made in a systematic way. Especially when supported by user-friendly computer tools, modeling and simulation methods permit large and complex genetic regulatory systems to be analyzed.

Traditionally, biologists use "block and arrow" diagram to demonstrate a particular model of molecular mechanisms. In this type of modeling, the functional idea exists before the model itself is constructed. It needs to be noted that the term "modeling" in this thesis is beyond its traditional meaning. Not only being a means to demonstrate a particular preconceived functional idea, modeling in this thesis should be also seen as a way to organize and formalize existing data on experimentally derived relationships. Used in this way, models become a means of ***mining*** functionally relevant relationships rather than devices to prove the plausibility of a particular preexisting idea. In other words, modeling I am concerned in this thesis is a discovering tool using experimental data. This approach is sometimes called *inverse modeling,* or *reverse engineering* problem, which is different with the term forward modeling. Perform a forward modeling need extensive knowledge of the system of interest, while this is not the case of gene regulatory networks.

2.1.4 Reverse Engineering

Traditionally, reverse engineering refers to taking apart an object to see how it works in order to duplicate or enhance the object. In automobile industry, for example, a manufacturer may purpose a competitor's vehicle, disassemble it and figure out the components for the purpose of improving its own product. In software industry, an example is to reverse machine code to source code for the purpose of debugging, hacking or improving. In the domain of gene regulatory network study, reverse engineering is defined as follows:

Reverse engineering problem (network inference problem):

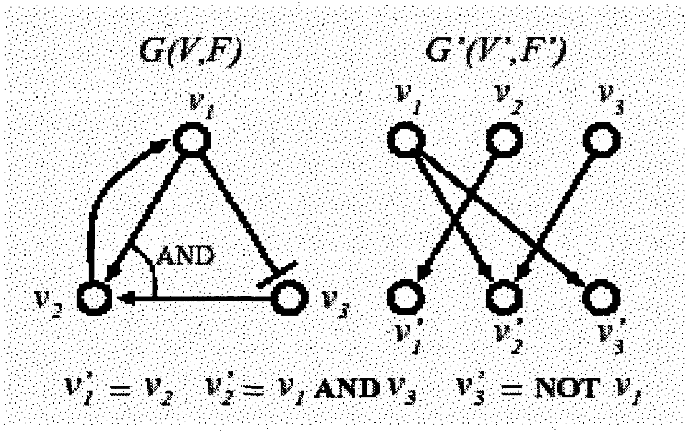
Given an amount of expression data, what can we deduce about the unknown underlying regulatory networks? [D'Haeseleer *et al.*, 2000]

Reverse engineering typically requires the use of a quantitative model, the parameters of which are then fit of the real-world data. If the connection structure of the regulatory network is unknown, the parametric model will necessarily have to be very general and simplistic. The results of this sort of model only relate to the overall network structure.

From this problem definition, one may wonder whether existing data mining tools can be applied to solve this problem. However, it is hard to apply data mining tools in their pure form. Data mining is often to extract hidden *predictive* information from large data sets. Current data mining tool include decision trees, clustering algorithm, rule induction, and artificial neural networks, etc. They are mostly used in classification and forecasting business. We can of course do a classification or clustering to our gene expression matrix. However, the inter-relationship between all genes remains unanswered. Data mining tool in its complex form (i.e. neural networks) is notorious for its "black-box" structure which is not good for clarifying structure of gene networks. Some literature has made attempts using neural networks in its modified form to solve reverse engineering problem [D'Haeseleer 2000]. The results are not quite satisfying. Thus, it is better to view this modeling technique as an advanced complement of current data mining tools. Specific framework of models to solve reverse engineering problem will be discussed in the rest of this chapter.

2.2 Boolean Network Model

As a first approximation, the state of a gene can be described by a Boolean variable expressing that it is active (on, 1) or inactive (off, 0) and hence its products are present or absent. Interconnections between elements can be represented by Boolean functions which calculate next state of a gene from current states of other genes. The result is a *Boolean Network* [Kauffman 1993].



INPUT			OUTPUT		
v_1	v_2	v_3	v'_1	v'_2	v'_3
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	1	1	0

Figure 3 A Boolean Network Example

A Boolean network (Figure 3) $G(V,F)$ consists of a set $V=\{v_1,\ldots,v_n\}$ of nodes representing genes and a list $F=\{f_1,\ldots,f_n\}$ of Boolean functions, where a Boolean function $f_i(v_{i1},\ldots,v_{ik})$ with inputs from specified nodes $v_{i1},\ldots,v_{ik}$ is assigned to each node v_i. Each node v_i has a *state* $S(v_i)=$ 0 or 1 which indicate "on" or "off" state of gene v_i. If it does not cause confusion, we can omit S. For example, we write v_i=1 for denoting $S(v_i)$=1. Hence, from definition we have the following definition for a Boolean network:

$$v_i\,|_{t+1} = f_i(v_{i1}\,|_t,\ldots,v_{ik}\,|_t) \qquad 1 \le i \le n \qquad (1)$$

We can rewrite $v_i|_{t+1}$ as v_i', $v_{i1},\ldots,v_{ik}$ as a vector $\vec{v}$, this way we can define the network is a nicer form:

$$v_i' = f_i(\vec{v}) \qquad 1 \le i \le n \qquad (2)$$

where $f_i = \{0,1\}^n \rightarrow \{0,1\}$ is a Boolean function mapping $\vec{v}$ to v_i'.

Wiring diagram [Liang *et al.*, 1998] *G'* and state transition table (complete) is also shown in Figure 3. In wiring diagram, the upper row lists the state at *t* and lower row the state at *t*+1. The transition from one state to the next state is usually determined in a parallel fashion, applying the Boolean function of each element to its inputs. Hence, transitions between states in a network are *deterministic*, with a single output state for a given input, and *synchronous*, in the sense that the outputs of the elements are updated simultaneously.

A sequence of states connected by transitions forms a *trajectory* of the system. Because the number of states in the state space is finite, the number of states in trajectory will be finite as well. More specifically, all initial states of a trajectory will eventually reach a steady state or a state cycle, also referred to as *point attractor* or *dynamic attractor*, respectively. All the state transitions constitute *the basin of attraction*. The dynamics of Boolean network has been summarized in Kauffman's book [Kauffman 1993]. $\vec{v}$ and $\vec{v}'$ can be measured as a time-series of expression which are normalized to $\{0,1\}$.

In Boolean network context, the reverse engineering problem becomes:

Definition: *Given a set of $\vec{v}$ and $\vec{v}'$ for a series of time points, find f_i for all i.*

An equivalent definition is:

Definition: *Given a set of time series gene expression data (which can be transformed in to a set of state transitional tables), find the complete state transitional table (which can be transformed into Boolean functions) with 2^n entries.*

Liang *et al.* developed an algorithm called REVEAL aimed to solve this problem [Liang *et al.*, 1998]. In brief, this algorithm uses information theory to establish how given elements are connected in the network and then determines the functions that specify the logic of the interactions from the data. Akutsu *et al.* designed a much simpler algorithm [Akutsu *et al.* 1999]. The idea is simply search all possible Boolean functions which fit expression pattern time series.

Boolean network represent the first step modeling for gene regulatory networks. It is simple to analyze and has efficient algorithms but this convenience comes with big price. The model makes huge assumption of the real system.

1. Discrete time model: synchronous updating which is obviously not true in real gene regulatory networks. All transcription factor works as a reactant in the system and they have different kinetics constant.

2. "On/Off" state: discrete gene states are too simple to characterize gene expression state. We knew that real gene expression patterns are not rectangular waves.
3. It is a deterministic model: a deterministic network is a rigid system, where the current system state unambiguously determines next system state. In a stochastic system, on the other hand, current state can have more than one next state. In reality, gene networks are stochastic. [Szallasi 1999]
4. Boolean functions: questionable. Transcription factor often work as a group but whether their relationship are Boolean or additive or something else is still an open problem. See Figure 4.

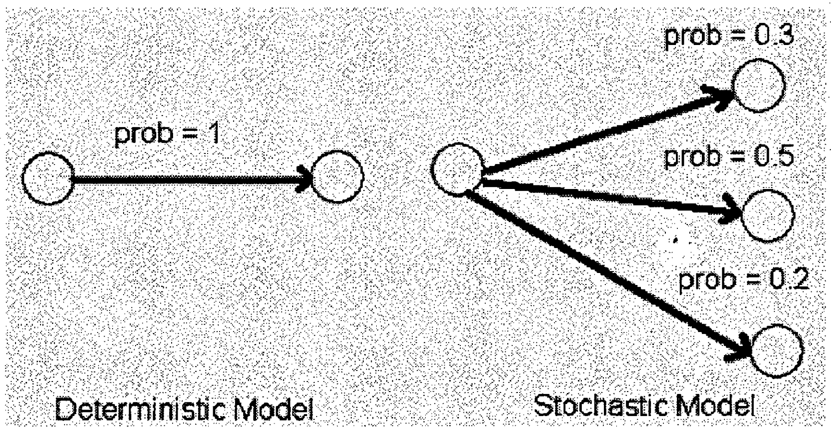


Figure 4 Deterministic or Stochastic State Transitions

These criticisms prohibit the use of Boolean network in fine-grained analysis. However, we can make use the concepts here and develop more sophisticate models.

2.3 Linear Additive Model

From chemical kinetics, we knew that one simple chemical reaction can be described by the following equations.

$$\begin{aligned} &A+B \xrightarrow{k} C \\ &C \xrightarrow{k'} A+B \end{aligned} \qquad (3)$$

A, *B*, and *C* are three chemical species. According to the equation, *C* is produced from *A* and *B* at rate constant *k*. *C* is also degenerated into *A* and *B* at rate constant *k'*. The actual rate *C* is generated in the system can be described by using the following equation:

$$\frac{d[C]}{dt} = k[A][B] - k'[C] \qquad (4)$$

The rate of *C* is generated is determined by the concentration of other species (may include *C* itself) and all the rate constant *k*. This is an Ordinary Differential Equation (ODE) model for chemical specie *C*. ODE is a widely used mathematical modeling method which has many applications in engineering. Inspired by this idea, we can use ODE to model gene regulatory networks since every gene can also be considered a chemical specie and its interaction can be considered chemical reaction. We can generalize the ODE model as the following equation:

$$\frac{d[v_i]}{dt} = f_i(\vec{v}) \qquad 1 \le i \le n \qquad (5)$$

v_i is the *i*th gene in the system and $\vec{v}$ is the vector denote all genes in the system. We can see this equation is very similar to equation (2) which represent Boolean network except that this is a time-continuous and value-continuous model and $f_i : R^n \to R$ is a general (possibly Boolean) function. In the linear model, reverse engineering problem becomes finding all *f_i's* that fit equation (5). Since *f_i's* are general functions, retrieve them is a hard job. We can make further assumptions that *f_i's* are linear functions. Several variants of such models have been proposed, with each group a different name: connectionist model [Mjolsness *et al.*, 1991], linear model [D'haeseleer *et al.*, 2000], linear transcription model [Chen *et al.*, 1999], and weight matrix model [Weaver *et al.*, 1999].

In D'haeseleer's dissertation [D'haeseleer's 2000], he used the following equation to describe the linear additive model:

$$\frac{dv_i}{dt} = \sum_j w_{ji} v_j + b_i \qquad 1 \le i \le n \qquad (6)$$

w_{ji} is the constant indicate the influence of gene *j* on the regulation of gene *i*. b_i is the external influence constant on gene *i*. He also transforms this equation into the following equation by multiply *dt* and add *v(t)* to both side of equation (6):

$$v_i(t) \approx v_i(t+dt) = \sum_j w'_{ji}\, v_j(t) + b'_i \qquad 1 \le i \le n \qquad (7)$$

Or,

$$v_i = \sum_j w'_{ji}\, v_j + b'_i \qquad 1 \le i \le n \qquad (8)$$

In a compact form, equation (8) can be written as:

$$\vec{v} = W\vec{v} + \vec{b} \qquad (9)$$

This is a linear equation system. Given enough example of v_i (expression data), we can use linear algebra or multiple regression equivalently to solve W and $\vec{b}$ so that f_i's are retrieved.

Chen *et al.* develop a more elaborate model for the gene regulatory system [Chen *et al.*, 1999].

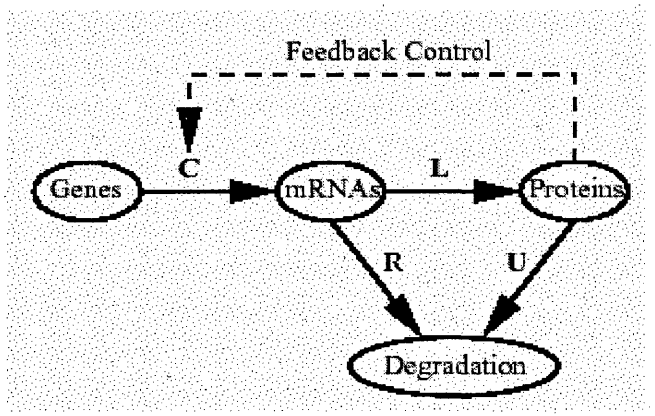


Figure 5 Dynamic Modeling of Gene Regulatory Network [Chen *et al.*, 1999]

$$\frac{d\vec{v}}{dt} = C\vec{p} - R\vec{v} \qquad \frac{d\vec{p}}{dt} = L\vec{v} - U\vec{p} \qquad (10)$$

where the variables and functions are defined as follows:

$\vec{v}$:	mRNA concentration, n dimensional vector
$\vec{p}$:	protein concentration, n dimensional vector
C :	Transcription function; $n \times n$ non-degenerate matrix
R :	Degeneration rates of mRNAs; $n \times n$ non-degenerate diagonal matrix
L :	Translational constants; $n \times n$ non-degenerate diagonal matrix
U:	Degeneration rates of proteins; $n \times n$ non-degenerate diagonal matrix

From Figure 5, each gene is regulated by multiple other gene products, so that C is not a diagonal matrix. Other functions *R, L, U* are controlled by a single source, so that they are diagonal matrices. This model is closer to real system but it needs protein level measurements. Also this model clearly has more parameters and reverse engineering this model is more difficult. This phenomenon is common in modeling community. No pain, no gain.

Note that the variables in equation (6) can theoretically become negative, or unboundedly large. Since these variables typically correspond to concentration levels, we may want to impose realistic upper and lower bounds. Most genes exhibit a sigmoid (S shape) curve. Weaver *et al.* [Weaver *et al.*, 1999] designed a variant linear discrete-time model, in which the linear additive model is further transformed using a sigmoid function to make it more biological realistic.

$$v_i(t+1) = S(\sum_j w_{ji} v_j(t) + b_i) \qquad 1 \le i \le n \qquad (11)$$

Or in a continuous time form proposed by Reinitz *et al.*:

$$\frac{dv_i}{dt} = S(\sum_j w_{ji} v_j + b_i) - r_i v_i \qquad 1 \le i \le n \qquad (12)$$

where r_i is the decay constant for gene *i*; sigmoid function *S(.)* can be $S(x) = (1+e^{-x})^{-1}$, $S(x) = \tanh(x)$, or a more biologically justified dose-response curve (see Figure 6). Clearly this model is inspired by the concept of Artificial Neural Network computing. Genes works as neurons in a recurrent neural network. Weighted input sum are viewed as regulating power for a gene and this sum is used to determine whether the gene will "fire" or not. Because ANN has many well developed theories and concepts, people in ANN field are more likely use it in their research.

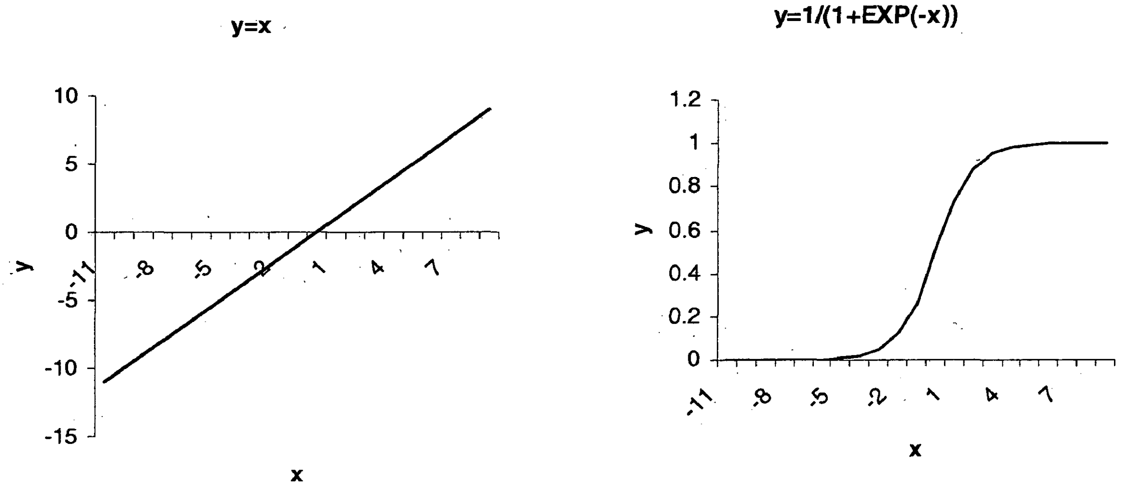


Figure 6 Sigmoid Transformation

In summary, linear additive model is a very useful and simple tool for modeling dynamic gene regulation system. Many powerful mathematical tools, such as

Ordinary Differential Equation, Linear Algebra, and Statistics, support the reverse engineering of this model. Its continuous-value functions are also more realistic than Boolean network. However, its name indicates its weakness: it is linear and additive, which imply that regulations are modeled as independent events. Actual biological system is far more complicated than this. For example, it is known that transcriptional regulators have different activities depending on their protein partners [Garrell *et al.*, 1991]. Additive functions are also a huge assumption and thus need to be verified or confirmed.

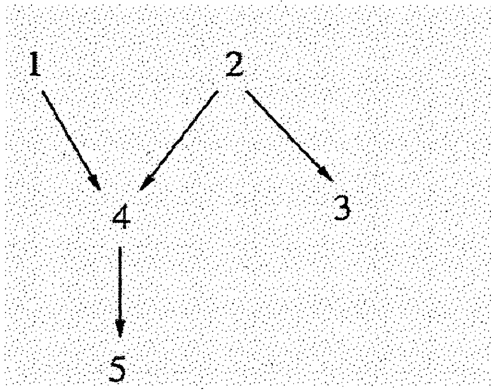
2.4 Bayesian Network Model

In the formalism of *Bayesian networks* [Friedman *et al.*, 2000], the structure of a genetic regulatory system is modeled by a directed acyclic graph *G(V,E)*. The vertices $i \in V$, $1 \le i \le n$, represent genes and correspond to random variables X_i.

If *i* is a gene, then X_i will describe the expression level of *i*. For each X_i, a *conditional distribution* $p(X_i \mid parent(X_i))$ is defined, where $parent(X_i)$ denotes the variables corresponding the direct regulators of *i* in *G*. The graph *G* and conditional distribution $p(X_i \mid parent(X_i))$, together defining the Bayesian network, uniquely specify a *joint probability distribution p(X)*.

A *conditional independency* $i(X_i; Y|Z)$ express the fact that X_i is independent of $\boldsymbol{Y}$ given $\boldsymbol{Z}$, where $\boldsymbol{Y}$ and $\boldsymbol{Z}$ denote sets of variables. The graph encodes the *Markov assumptions,* stating that for every gene i in G, $i(X_i;\ nondescendants(X_i)|parents(X_i))$. By means of the Markov assumption, the joint probability distribution can be decomposed into

$$p(X)=\prod_{i=1}^{n} p(X_i \mid parents(X_i)) \tag{12}$$



$p(X_1),\ p(X_2),\ p(X_4 \mid X_1, X_2)$
$p(X_5 \mid X_4),\ p(X_3 \mid X_2)$

$p(X) = p(X_5 \mid X_4)\, p(X_4 \mid X_1, X_2)\, p(X_3 \mid X_2)\, p(X_2)\, p(X_1)$

$i(X_1; X_2, X_3),\ i(X_2; X_1),\ i(X_4; X_3 \mid X_1, X_2)$
$i(X_3; X_1, X_4, X_5 \mid X_2),\ i(X_5; X_1, X_2, X_3 \mid X_4)$

Figure 7 A Bayesian Network Example [Friedman *et al.*, 2000]

Given a set of independent values for X transformed by a set of expression data, learning techniques for Bayesian networks allow one to induce the network. Basically, the techniques rely on a matching score to evaluate the networks with respect to the data and search for the network with optimal score.

A Bayesian network approach towards modeling regulatory networks is attractive because of its solid basis in statistics, which enables it to deal with the stochastic aspects of gene expression and noisy measurements in a natural way. Moreover, Bayesian networks can be used when only incomplete knowledge about the system is available.

2.5 Clustering

A key step in the analysis of gene expression data is the identification of groups of genes that manifest similar expression patterns. This translates to the algorithmic problem of clustering gene expression data. A clustering problem consists of elements and a feature vector for each element. A measure of similarity is defined between pairs of such vectors. (In gene expression, elements are usually genes, the vector of each gene contains its expression levels under each of the monitored conditions, and similarity can be measured, by the correlation coefficient between vectors.) The goal is to partition the elements into subsets, which are called *clusters*, so that two criteria are satisfied: *homogeneity* –elements in the same cluster are highly similar to each other; and *separation* – elements from different clusters have low similarity to each other. For elements with unknown or largely unknown mechanisms such as a gene regulatory network, clustering often represent the first step data analysis. In artificial intelligence terminology, clustering is an un-supervising learning method because clusters are totally derived from data itself (no pre-knowledge about the data). The purpose of clustering gene expression data is two folded: organize expression data to facilitate visual analysis; reduce the number of entries to enable advanced data mining and analysis. Several algorithmic techniques were previously used in clustering gene expression data, including hierarchical clustering, self organizing maps, K-means, and graph theoretic approaches. I will use a combination of clustering methods in the last chapter.

2.6 Other Issues

In the beginning of this chapter, I have introduced the concept of *validity* of model. Our goal is to create a structural valid model of the real system. However, all current models are far from this goal because the internal detailed mechanisms of the gene regulatory networks are largely unknown. Every model makes some assumptions of the real system, which is fine for coarse-grained modeling. For fine-grained modeling are needed in the future to reveal detailed gene networks.

Computing complexity is another concern in the modeling work. One genome often contains thousands of genes. One organism may have billions of cells. Fitting the parameters of the model require heavy computation. Deriving parameter for the model is often NP-hard, such as the case of Bayesian networks. Designing efficient algorithms is one major topic in the modeling community. In the next two chapters, I will apply genetic algorithms for efficient modeling of gene networks.

Measuring more variables (so that parameters) allows for a more exact modeling, but makes the correct model exponentially harder to find. It is well known that the more variables one models, the harder the modeling task becomes, because the space of models to be searched increases exponentially with the numbers of parameters of model, and therefore with the number of variables. This is often referred to as the *Curse of Dimensionality*. To get around the curse, I will perform a clustering procedure to reduce the number of variables and parameters.

Data requirement is another key concern when we perform a modeling. Gene expression data are often awkward in dimensionality (this dimensionality is not the same as Curse of Dimensionality which means search space). We are often confronted with a table of data with thousands of genes and only dozens of time point measurements. An analogy to this is trying to solve a linear equation system with thousands of unknown variables and only dozens of equations. However, this situation will be improved with the advance of biological technology. Again, clustering may also help to avoid this problem. Anyway, determining how many data points are needed to perform a successful modeling is a prerequisite work.

In summary, several modeling frameworks are introduced in this chapter. These examples are by no means exhaustive. Actually, papers in this area are coming at an exponential rate and this trend has no signs to stop in the near future. The works in this chapter arouse my interest and set up a background for this thesis. In next chapter, a new incremental model and its reverse-engineering algorithm will be introduced. I will also give a formal analysis of its data requirement and algorithm complexity. Then results and conclusions will be presented in chapter 4.

Chapter 3: Incremental Modeling and Fitting Algorithm

I may be wrong and you may be right, and by an effort, we may get nearer to the truth.

--Karl R. Popper

Recent technology advancement has made large-scale gene expression surveys a reality. Along with genome sequence data, massive gene expression data sets have made biology a data-rich subject. These data sets provide an opportunity to directly view the activity of hundreds of genes in parallel. However, manual analysis of these huge data sets is often not practical. Development of computational methods and data mining tools for knowledge inference from gene expression data base is the only way to face this challenge. Current widely used methods to facilitate analysis of gene expression data sets are clustering, classification, and visualization tools. These methods are used to group genes based on the similarity of expression patterns. If two genes are clustered together in this way, then they may share a common functional role. But if they have distinct expression patterns, how they are related? It is obvious that a simple clustering analysis cannot answer this question.

In order to answer this kind of query, we need to construct a gene regulatory network. The knowledge of gene regulatory network and its interactions will further the understanding of important biological processes such as disease, cell cycle, and development. Drawing regulatory network information from time series expression data sets is a reverse engineering problem. A common approach to solving this problem has its basis in mathematical modeling; we adopt this approach here. We first construct a mathematical model which simulates the real gene regulatory system with some simplification. Then we apply fitting algorithms to search for the best model parameters that will let the model behave closest to the data. The result parameters are then used to construct the regulatory network.

3.1 Introduction

In this chapter, I will present a new incremental model formally and give a fitting algorithm. A formal complexity analysis and data requirement analysis will also be prepared. How to build, test and fit the model will be discussed in detail. The implementation and testing of the fitting algorithm and model is left in the next chapter.

As I have introduced in chapter 2, modeling a real system is a complex process. Several important issues have to be considered:

- What is the formalism of the model?

- How well the model structurally represents the real system?
- How to reverse-engineer (fit, train) the model?
- How many data we need to fit the model?
- How long it takes to fit the model?
- How well the model fit the data?
- What information can we get from the model?

In this chapter, I will give a detailed answer for the first five problems. The last two problems will be discussed in the next chapter.

3.2 Genetic Algorithms

In this section, a very brief description of Genetic Algorithms (GAs) will be given. For a more complete description, see *e.g.*[Holland 1975], [Whitley 1994].

In a GA, the unknown parameters of the problem are encoded in strings of digits referred to as *chromosomes*. Initially, a *population* of *individuals*, each associated with one such string, is generated by assigning random values to all the locations (*genes*) along the strings. Then, for each individual, the variables are read off from the chromosome, the relevant computation is carried out, and the *fitness* of the individual is evaluated. The assignment of fitness values should be such that individuals close to reaching the goal set by the user obtain higher fitnesses than those who are far from the goal. When all the individuals in the first *generation* have been evaluated, the second generation is formed by first selecting *parent strings* in such a way that individuals with high fitness have a greater probability of being selected than those with low fitness, and then combining the genetic material contained in their chromosomes to form new strings, and, finally, allowing a small degree of *recombination* (i.e. cross-over) and *mutation* (i.e. random variation) of the newly formed chromosomes. The chromosomes thus formed constitute the second generation, which is evaluated by repeating the procedure used for the evaluation of the first generation. This iteration continues until a satisfactory solution has been found. A pseudo-code of canonical GA is shown below:

```
while(1)
{
      Compute and save the fitness u(m) for each
      individual m in current population M(t);

      Define selection probabilities p(m) for each
      individual m in M(t) so that p(m) is
      proportional to u(m);

      Generate M(t+1) by probabilistically
      selecting individuals from M(t) to produce
      offspring via genetic operators;
}
```

Figure 8 A Canonical Genetic Algorithm

3.3 Incremental Model

The model is inspired by previous works as I introduced in chapter 2, especially by the linear additive models and neural network models. First let us revisit Reinitz's neural network model [Reinitz *et al.*, 1995]:

$$\frac{dv_i}{dt} = S(\sum_j w_{ji} v_j + b_i) - r_i v_i \qquad 1 \le i \le n \qquad (1)$$

The growth rate of the expression level for a gene is controlled by the transformed weighted sum of all other genes plus a bias term and a decay rate. v_i is the expression level of the *ith* gene; the left-hand side is the growth rate of v_i; w_{ji} is the $n \times n$ weight matrix element represent regulation power of the *jth* gene to the *ith* gene; b_i is the constant bias term for the *ith* gene; $S(\cdot)$ is a sigmoid function which can be $S(x) = (1 + e^{-x})^{-1}$. This is a continuous-time and continuous-value model which reflects more reality than a simple Boolean discrete model. The sigmoid transformation constrains the expression growth rate in a reasonable range. This is the advantage over a simple linear additive model in which the growth rate can be extremely high and low. The downside of this model is also significant: biological processes are simplified by the weighted matrix; gene regulations are modeled as independent events without considering combined effects; the huge sparse matrix contains too many parameters and thus is computationally hard to solve.

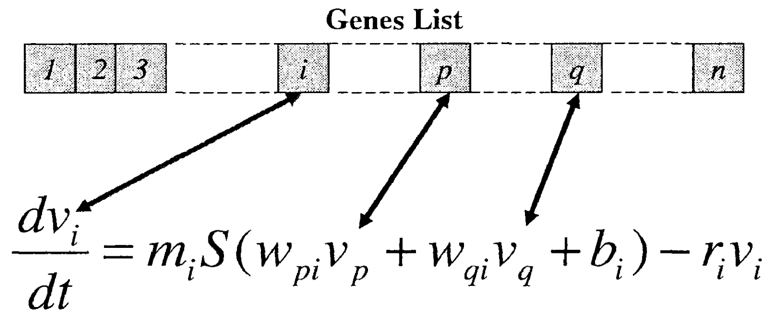
GAs have been widely applied in many areas including fitting genetic regulatory network models [Ando *et al.*, 2000][Wahde *et al.*, 1999]. In their papers, they describe the application of GA to existing models such as Weaver *et al.* [Weaver *et al.*, 1999]. They claim the results are plausible and make biological sense. However, since Weaver's model consist of too many parameters ($n \times n$, n is the number of genes), we need at least n time-series data points to effectively fitting the model, which is not possible for genome scale modeling under current technology. In other words, this kind of models is usually under-determined. Another problem with this kind of models is the parameter matrix is very sparse. Most of them are zeros. Thus most of the computing power is wasted in determining this sparse matrix. Because GAs are usually regarded as slow, it is not possible to do a large-scale analysis. In fact, both methods from Ando *et al.* and Wahde *et al.* can only be used to deal with very small networks such as 4 nodes. A third problem is encoding a large number of parameters is not feasible. For example: if we are studying a 1,000 genes system (which is a relatively small genome), we have at least 1,000,000 parameters to fit. If we are using genetic algorithm, each parameter is a substring the chromosome. If each parameter is encoded as 3 bits, the chromosome will be more than 3,000,000 bits long. Handling such a huge chromosome in a genetic algorithm is almost impossible. We have to design a new encoding scheme or a new model for such system. A fourth problem is that this kind of models only considers linear effect of gene regulation. In other words, genes are independently regulating target gene. This is a huge assumption of the gene regulation pattern and clearly not true from experimental results. In my model, interaction effects of gene regulation are considered as a non-linear term, e.g. $v_a \times v_b$.

3.3.1 Basic Model

Following the above analysis, the models were built:

$$\frac{dv_i}{dt} = m_i S(w_{pi}v_p - w_{qi}v_q + b_i) - r_i v_i \tag{2}$$

In order to improve computability of (1), Basic Model (2) is introduced. Instead of including whole weight matrix as parameters, I search for the one that is responsible for regulating other genes activity. By doing this I add two new parameters *p* and *q* in (2). See Figure 9 for an illustration. *p* is a gene encode an enhancer for gene *i*. *q* is a gene encode an repressor for gene *i*. Traditionally all parameters we have seen in this thesis are on coefficient position. But now *p* and *q* are on subscript position. This trick allows us to save significant time and space to reverse engineering the model without losing important information. This is like encoding computer memory address using address buses. In (1), we have more than $n \times n$ parameters. In (2), we have only 6*n* parameters. Model (2) is not easy to fit by mathematical method since it doesn't have a canonical form. However, if we fit the model using GA, the chromosome is relatively short due to lower number of parameters. Evaluation the fitness is also much easier since this model has a lighter computation load than (1).



$$\frac{dv_i}{dt} = m_i S(w_{pi} v_p + w_{qi} v_q + b_i) - r_i v_i$$

Figure 9 Basic Model

m_i is the maximum expression growth rate allowed for gene *i*. Weaver *et al.* [Weaver *et al.*, 1999] introduced this parameter in their model. This parameter is used to scale the growth rate to a reasonable level because Sigmoid function *S(·)* output a value from 0 to 1. Note that all parameters in the model are non-negative (*p*, *q* are integers).

There is a natural constraint that every gene in the system has at least one enhancer and one repressor, so that the expression is controlled in a reasonable level. I thus conveniently use gene p and gene q to model this fact. An obvious problem with this model is the connectivity of each gene is limited to 2 with one enhancer and one repressor. However, the fact is that every node often has low connectivity (usually less than 10) so that this is a reasonable approximation. Anyway, we can also extend this model to include more connections easily if we wish to model in greater detail:

$$\frac{dv_i}{dt} = m_i S(w_{pi}v_p - w_{qi}v_q + w_{si}v_s + b_i) - r_i v_i \tag{3}$$

In (3), I added another promoter s for i to increase the connectivity to 3. In fact, we could include as many ingredients in the model without adding much complexity.

For better resemblance to a real biological system, the model is designed to be value-continuous, time-continuous, and value-constrained by a sigmoid function. Because gene regulatory networks are very sparse networks with most connections being zero, it is not necessary to design a fully connected model. By indexing regulating genes instead of including all genes in the model, we avoid a fully connected model which is the case in a recurrent neural network. The advantage of doing this is two-folded: computation load is significantly reduced and, more importantly, the number of parameters is reduced by

$O(n)$(see next section for detail). As we know, search space grows exponentially with number of parameters. Reducing the number of parameters improves the chances of pinning down the right regulatory network.

3.3.2 Nonlinear Model

In order to attack the linear assumption of (1), I included a nonlinear term in this nonlinear model:

$$\frac{dv_i}{dt} = m_i S(w_{pi} v_p + w_{qi} v_q + w_{sti} v_s v_t + b_i) - r_i v_i \tag{4}$$

The intention of introducing non-linear term $w_{sti}v_sv_t$ is to honor the fact some gene regulations are dependent to each other. This term means that the regulation to gene i from gene s and t are dependent to each other and the co-regulating effect is scaled by parameter w_{sti}. The bold arc in Figure 10 demonstrates this dependent interaction. We can use this term to model this kind of interaction conveniently because the value of the term depends on the expression level of both gene s and gene t. The genes regulations are no longer independent events. One can affect others. Previous work fears to put this nonlinear term to the model because of the computability. But now, with our basic model frame work, this term can be calculated without too much pain. See basic model for the reason.

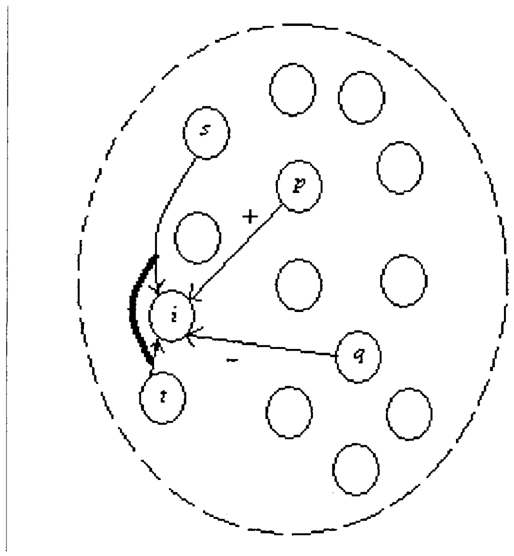


Figure 10 Nonlinear Model

Similar to basic model, the nonlinear model may become complex if we allow connectivity more than 2. For example, $w_{abci}v_av_bv_c$ could mean gene a, b and c are co-regulating gene *i* and they depend on each other. It should be noted that parameters *s* and *t* may be same as *p* and *q* in (4). Also note that w_{sti} can be negative to represent the negative co-regulation effect.

The nonlinear model can also be extended to model more subtle relationship. For example, it is known that transcriptional regulators have different activities depending on their protein partners [Garrell *et al.,* 1991]. To model such complex relationship, we could add this term to the model:

$$(w_{abi}v_av_b + w_{aci}v_av_c) \tag{5}$$

This term models the fact that gene a has different regulating effects to gene i depending on gene b and gene c. It is obvious that the value of this term depends on the value of v_b and v_c. Because there are two weight parameters in the term, the discriminating effects can be represented. We can see that the nonlinear model framework is flexible, which is required to model complex effects of biological mechanisms.

A common modeling guideline is to increase model complexity until reality (data) is satisfied. From equation (2) to (5), we are following this guideline. One trick we can play here is we can automate this process to make an incremental modeling. This will be explained in next section.

3.4 Model Analysis

For a black-box system, modeling is in the middle of science and art. We build the model based on limited knowledge of the black box, then add flavor to the model according to personal understanding. We then collect data from experiments to train the model. Parameters of the model are known when the training is done. Those parameters can be used to test the validity of the model and also reveal the internal interaction of the real system. Recall that if the predictive data generated from the model is close to the real data in an acceptable level, the model is called replicatively valid or predictively valid. However, our goal is to generate a structurally valid model.

Only when the interaction revealed from the model is observed by experiment, the model can be called structurally valid. Otherwise, this model can be wrong or incomplete. All previous work in this field is still in the predictively valid model level and this fact is somewhat overlooked by the researchers. Generating a structurally model needs greater knowledge about the system we are studying and also a greater mathematical insight. Even if a structurally valid model for one biological system can be derived, don't forget the fact that diversity is a major biological feature. So in this page, I'd rather say that this field is only a beginning, not an end. The situation in this field is just like some proponents [Bittner *et al.,* 1999] said: "We can stay in our room, maximizing our feeling of safety, but minimizing our vision. Or we can venture forth – albeit with more "baby steps" – to confront our greatest fears: non-linear or multi-input relationships." Although gene regulatory networks seems too complex to model, computation is our only way out.

In my model, I added nonlinear modeling and subscript parameter to overcome the deficiencies of linear additive modeling. Nonlinear terms help to reveal dependent regulating interactions. Introducing subscript parameters significantly reduce the number of parameters. In a fully connected network model, the number of parameters is $O(n^2)$, where n is number of genes. In this proposed incremental connectivity model, the number of parameters is $O(nk)$, where k is network connectivity. For large scale system, k is far less than n and can be seen as a constant, so the number of parameter is actually $O(n)$. So we see the number of parameters is reduced by $O(n)$ in this model. Recall the *curse of dimension*: search space exponentially grow with the number of parameters.

The search space is then significantly reduced. This analysis suggest that fitting the incremental connectivity model by some kind of search algorithms, such as a GA, could more likely yield better results than fitting a fully connected model. As far as I knew, those improvements are first introduced to this field. However, the inference from the model fitting needs to be confirmed by experiment results to prove my model is structurally valid. A structurally valid model is still a long-term goal at this point.

Another major advantage of this modeling framework I presented is its adaptive capability. When used with GA, we can add any ingredients to the model to incorporate greater detailed modeling as I have shown above. This process can be automated by designing a fitting algorithm embedded with GAs. The advantage doing this is we avoid a fully connected model without losing modeling performance.

It should be kept in mind that the models I introduced in this chapter are coarse-grained models. The data and understanding of the system limit our ability to model it at in-depth biochemical level. We cannot realistically hope to characterize all the relevant molecular interactions one-by-one—at least not in the near future. Many aspects of the system are ignored temporarily.

The model presented here is a non-spatial model. Spatiality can play an important role, both at the level of intercellular interactions, and at the level of cell compartments (e.g. nucleus vs. cytoplasm vs. membrane). Most processes in multicellular organisms, especially during development, involve interactions between different cells types, or even between cells of the same type. A spatial model is needed to accommodate such modeling requirements.

The deterministic nature of my model determines this is a coarse-grained one. In order to make a fine-grained model, stochastic character has to be added. However, due to its complexity, stochastic model is limited for small system. For large system such as a genome, we can only focus on deterministic model.

3.5 Data Requirement

From linear algebra or multiple regression, we know that we need at least n+1 data points to train a fully connected network model. However, my model is a continuous model with a limited connectivity k. Up till now, it is still an open problem that the data required for continuous modeling with limited connectivity. Some researcher suggest that the data requirement be $k\log(n/k)$ [D'haeseleer *et al.*, 2000]. However, this is still a conjecture. In later chapter, I will try to use experimental methods to test this hypothesis.

3.6 Model Fitting Algorithms

As I have mentioned in chapter 2, many methods exist for model fitting. If the search space is not big, simple exhaustive can be used to guarantee a global optimal found. If the problem can be solved mathematically, an analytical optimization should be enough and fast. If the search space is huge but has single smooth global optimal, a hill climbing technique (gradient based) could be used. If all these "good" conditions fail, a GA or other meta-heuristic can be used to fit the model. The topic of this thesis is not on GA itself but modeling. Instead, I will only focus its application as a heuristic function optimizer. For this purpose, GA can be useful for two largely distinct purposes [Everett 1996]. One purpose is the selection of parameters to optimize the performance of a system. Such systems typically depend upon decision parameters, chosen by the system designer. Appropriate or inappropriate choice of decision parameters will cause the system to perform better or worse, as measured by some relevant objective or fitness function. By encoding the set of parameters as a bit string and evaluate there performance by fitness function in a GA, one can usually find a good (not necessarily best) set of parameters to optimize the system performance. The second potential use for GAs has been less discussed, but lies in the field of testing and fitting quantitative models. This is the focus of this thesis. We know that fitting a quantitative model is the work to find a set of parameters which make the model behaves like real system. In contrast to the situation where we were trying to maximize the performance of a system, we are now trying to find parameters that minimize the difference between the mode and the data. Fitness function in this sense is expressed as the difference between the model and the measured data. We know this model may contain thousands of genes and thus thousands of parameters. Recall dimension curse: search space exponentially grows with the number of parameters. Thus it is not possible to explore the whole search space to find global optimal. Another by-product from dimension curse is that large scale expression data often does not have enough data points to train the model so that the system is under-determined. This fact infers that we may find many locations in the search space that have very good fitness. This kind of model is hard to train by analytical method such as linear algebra. GAs are especially useful to do this kind of "dirty" work. We don't need to know how to solve the parameters, a fitness function is enough. This flexibility offered from GA is particularly useful in training non-regular form of models. We know the gene regulatory network is a complex system. The mechanisms are not well understood. The fitness function is noisy since gene expression measurement is noisy. The shape of the fitness function is very likely to be non-unimodal (no single hill) and non-smooth. All these characteristics of the problem inspire us to use GAs in the model fitting process. The problem of GA is that it is a heuristic and thus no global optimal is guaranteed and it is a weak method, which means it is a general method and should be inferior to a domain specific method. However, there are no domain specific methods available for this problem and we have to accept GA as a desirable method for fitting this model for now.

The model fitting algorithm is slightly different from a traditional GA. In order to adapt the connectivity of the model and automate the modeling process, I embed a GA inside a loop. If the model fitness is not sufficient, the model will update by increasing connectivity by one. A GA then fit the new model in hope of increasing the fitness to an acceptable threshold. This process will repeat until the model is acceptable. This algorithm is displayed as in Figure 11.

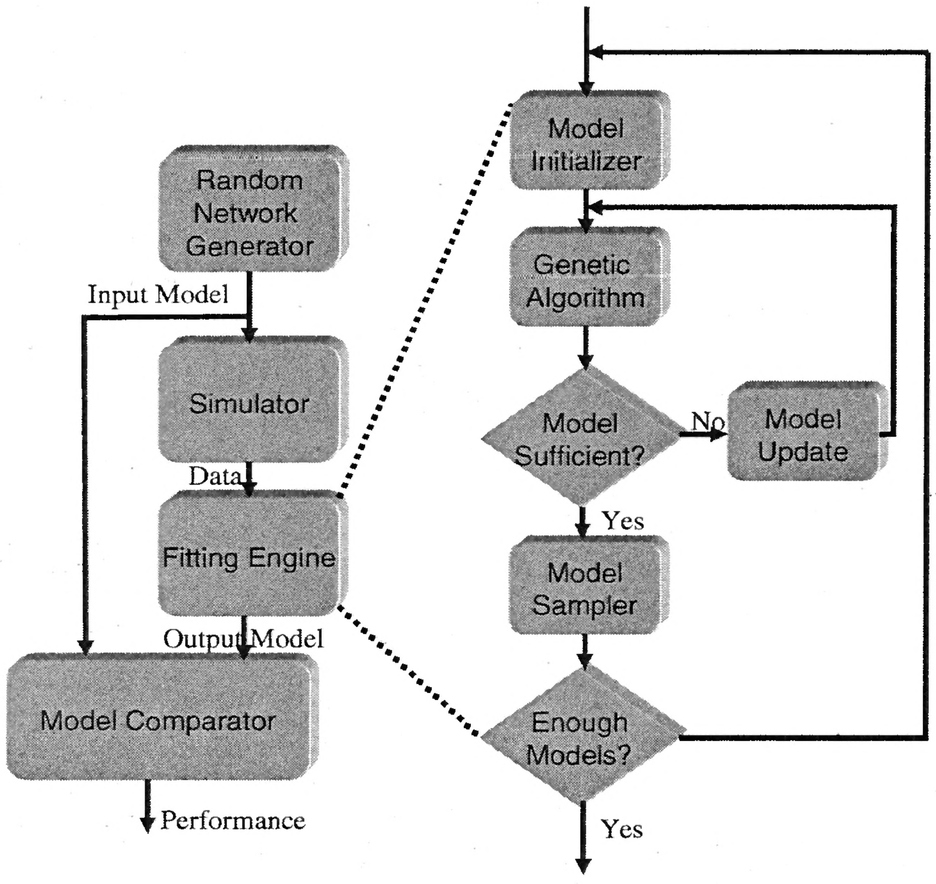


Figure 11 Model Fitting Algorithm

3.7 Algorithm Complexity

We have shown the number of parameters is significantly reduced by *O(n)*. Now we will prove that computation load in terms of time and memory space are also significantly reduced in the incremental connectivity model. The performance of a GA is determined by many factors, such as population size, number of generations, evaluation functions, *etc.* Evaluating a chromosome requires calculating numerical solution of a set of differential equations utilizing Runge-Kutta method. Thus, efficiency in evaluating the model is a major concern when model is designed. For fully connected model, at each time point, there are *n* differential equations to evaluate. For each equation, we have to compute the sum of *n* weighted regulations terms. Thus,

$$T(n) = g \times p \times t \times n \times n = O(n^2)$$

where *g: number of generations*

p: population size

t: length of time series

n: number of genes

When fitting the fully connected model utilizing a GA, time complexity is determined by number of generation, population size, length of time series, and number of genes. For large-scale system, number of genes can be treated as the only variable and the other are constants.

For the incremental connected model fitting algorithm, time complexity is different because connectivity is restricted by *k*. For each loop, a GA is performed and model connectivity increase one until fitness reaches a satisfied level. For large-scale system, connectivity can be treated as a constant because it is far less than n. So the algorithm has a time complexity:

$$T(n) = g \times p \times t \times n \times \sum_{i=1}^{k} i = g \times p \times t \times n \times O(k^2) = O(nk^2) = O(n)$$

Similar analysis can be applied to space complexity. In a GA, each chromosome is an array of parameters. In a fully connected model, every chromosome has a space complexity of *O(n)*, so the space complexity for the GA is:

$$S(n) = p \times n \times O(n) = O(n^2)$$

In the incremental connected model, space complexity for each chromosome is reduced to *O(k)*. So the total space complexity for the fitting algorithm is reduced to: $S(n) = p \times n \times O(k) = O(nk) = O(n)$

		Fully Connected Model	Incremental Connected Model
Time Complexity	Small System	$O(n^2)$	$O(nk^2)$
	Large System	$O(n^2)$	$O(n)$
Space Complexity	Small System	$O(n^2)$	$O(nk)$
	Large System	$O(n^2)$	$O(n)$

Table 1 Complexity Comparison of Fully Connected Model and Incremental Connected Model

It is important to note that the above analysis is based on constant number of generations. If the termination method is by convergence, things will be a lot complicated and that is beyond the scope of this thesis.

3.8 Summary

In this chapter, I present a continuous-time continuous-value neural network flavor gene regulatory network model with incremental connectivity. I adapt the connectivity of a recurrent neural network model by indexing regulatory elements and including nonlinear interaction terms. The new technique reduces the number of parameters by $O(n)$, therefore increasing the chance of recovering the underlying regulatory network. In order to fit the model from data, I have developed a genetic fitting algorithm with $O(n)$ time complexity and that adapts the connectivity during the fitting process until a satisfactory fit is obtained. The advantage and limitation is also discussed for this model.

Chapter 4: Implementation, Testing and Results

Biology easily has 500 years of exciting problems to work on.

-- Donald E. Knuth

The purpose of this thesis is two folded: develop a method and apply this method to deal with specific problems. Based on the method I presented in last chapter, the model fitting algorithm will be implemented in this chapter. Some experiments will be designed to test the performance of the algorithm. The model and algorithm will also be used to reveal real gene regulatory networks. This method along with clustering analysis is applied to two sets of data, rat central nervous system (CNS) and yeast whole genome. These data sets represent small-scale and large-scale gene expression measurements respectively. The results followed by a discussion were also given in the end.

4.1 Algorithm Implementation

The fitting algorithm is implemented in C++. The core of this fitting algorithm is a GA. In order to save time and ensure accuracy, I use a C++ GA library obtained from http://lancet.mit.edu/ga/ [Wall 2002] as part of the code. When implementing a GA, we usually have many schemes and parameters to choose. And the decision of choosing which scheme and parameter often has great impacts on the GA performance. In this case, I found that standard setting work pretty well in terms of fitness and efficiency. Below is a table of parameter settings for this algorithm.

Scheme/Parameter	**Type**
Replacement	Non-Overlapping
Scaling	Sigma-Truncation
Selection	Roulette Wheel
Chromosome-Type	Real Number
Mutation	Gaussian
Mutation Rate	0.01
Cross-Over	Uniform Cross-Over
Cross-Over Rate	0.7
Number of Generations	500
Population Size	30

Table 2 Algorithm Settings

I use sigma-truncation scaling in stead of linear scaling to avoid pre-convergence problem. Real number is convenient to encode model parameters such as weight and bias terms. It should be noted that GA is basically a heuristic. Thus we can not guarantee the model retrieved from GA is a global optimal one. The algorithm may be trapped in a local optimum with near optimal fitness. In order to overwhelm local optimal noise, I make multiple runs take the majority of each parameter.

4.2 Evaluation Function

The evaluation function demands a simulation to be implemented. For each setting or parameters, I use the model to compute a set of simulation time series expression data. This data is compared with real time series data and the difference is the misfit (reverse of fitness). In a formal definition:

$$fitness = \frac{1}{1+misfit} = \frac{1}{1+\sum_{i=1}^{N}\sum_{t=0}^{T}(u_i(t)-v_i(t))^2} \tag{6}$$

In this definition, fitness is defined in the region (0,1]; $u_i(t)$ is the computed expression data for gene i at time t under a set of parameters (which is coded as a chromosome in GA); $v_i(t)$ is the observed expression data for gene i at time t.

Instead of trying to get analytical solution for the ODE system, numerical method (fourth order Runge-Kutta method) is used here to calculate approximate solution.

$$\vec{V}_{i+1} = \vec{V}_i + \frac{h}{6}(\vec{K}_1 + 2\vec{K}_2 + 2\vec{K}_3 + \vec{K}_4) \tag{7}$$

$t_{i+1} = t_i + h, \qquad i = 0,1,\ldots,n-1,$

where

$$\begin{aligned}
\vec{K}_1 &= \vec{f}(t_i, \vec{V}_i), \\
\vec{K}_2 &= \vec{f}(t_i + \frac{h}{2}, \vec{V}_i + \frac{h}{2}\vec{K}_1), \\
\vec{K}_3 &= \vec{f}(t_i + \frac{h}{2}, \vec{V}_i + \frac{h}{2}\vec{K}_2), \\
\vec{K}_4 &= \vec{f}(t_i + h, \vec{V}_i + h\vec{K}_3).
\end{aligned}$$

4.3 Algorithm Testing

The task of testing and validating this method is two-folded: 1. testing whether the fitting algorithm is effective in reconstructing the correct gene regulatory network. 2. testing the whether the model can represent biological reality. In order to perform the first task, we need to generate some simulated networks from the model, and then run the algorithm to compare the input and output networks. If the first testing passed, we can then go ahead with the second task by applying the algorithm to real gene expression data sets and confirm the resulting network with known gene regulatory pathways. In this section, we perform the first task. The performance of the fitting algorithm can be measured by the difference between the input and output networks.

Definition: For two directed graphs with the same number of nodes, the ***graph similarity/distance*** is the number of matches/mismatches in the vertices matrix divided by the product of connectivity and the number of nodes.

We use Graph Similarity to measure the ***performance*** of the reverse modeling. The total number of edges in the graph can be assumed as *O(nk)*. We divide the number of mismatches by $n \times k$ so as to normalize the graph distance. Thus, the graph distance ***d*** and can be used as a universal measurement for all graphs.

We first generate networks with random connections. Based on these networks, we then generate simulated expression profile according to the model. We then apply the fitting algorithms to those expression data to reconstruct the networks. The results are shown in Figure 12.

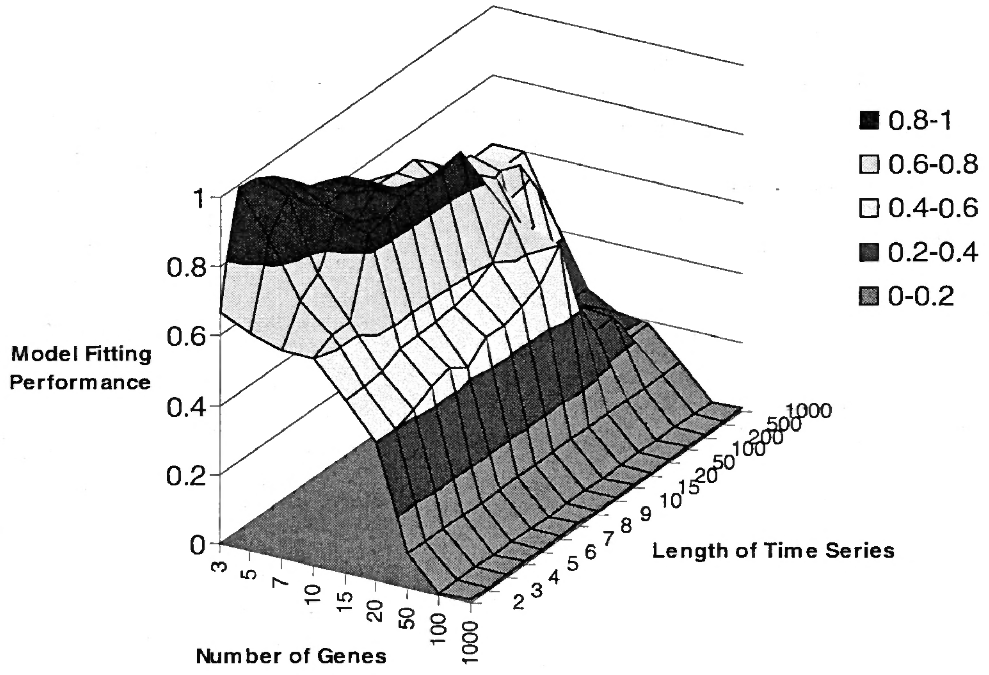


Figure 12 Performance Surface of Fitting Algorithm

For each random network, fitting algorithm runs 100 times and the value with majority votes was selected for each parameter. The output network was then constructed from model parameters. Modeling performance was then computed by comparing input and output networks. Each point in the performance surface is the average measurement of 100 random networks with corresponding input sizes.

From this figure, we can get the following important conclusions:

1. Because of the sheer complexity of modeling a biological system, practical input size for the fitting algorithm is 3-15 gene clusters. For large input size, clustering is necessary to reduce the input.
2. When input size is 3-15 genes, the algorithm can identify as many as 90% correct regulatory pathways.
3. Surprisingly, the length of time series is NOT the longer, the better. We found when the length of time series is close to the number of genes, this fitting algorithm generates optimal results.
4. Modeling performance is clearly better than [Wahde *et al.,* 2000] in which a fully connected model are used. Fully connected model required more data points to fit the model and the resulting network largely divergent [Wahde *et al.,* 2000].

Now that we have shown the fitting algorithm is able to reconstruct gene regulatory network under certain circumstances, I will apply this method to some real gene expression data sets to test whether the model can reflect biological reality.

4.4 Application to Rat CNS Data

The analysis in this section is based on the data obtained by Wen *et al.* [Wen *et al.*, 1998] The data set consists of measurements of gene expression levels for 112 genes during the development of central nervous system of rats, focusing on cervical spinal cord. Each gene was measured at nine different points in time using RT-PCR protocol. The first measurement was made 10 days before the birth, and the intervals between measurements were 2 or 3 days. The data provide a temporal gene expression pattern of spinal cord development based on major families of inter- and intracellular signaling genes. The 112 genes were clustered into 5 groups (waves) according to their Euclidean distances. Wen *et al.* found that genes belong to distinct functional classes and gene families clearly map to particular expression patterns. Specifically, wave 1 contains genes active during initial proliferation, wave 2 is associated with neurogenesis, wave 3 contains most genes for neurotransmitter signaling, wave 4 consists of genes active during the final maturation of the tissue, and wave 5 is made up of diverse genes.

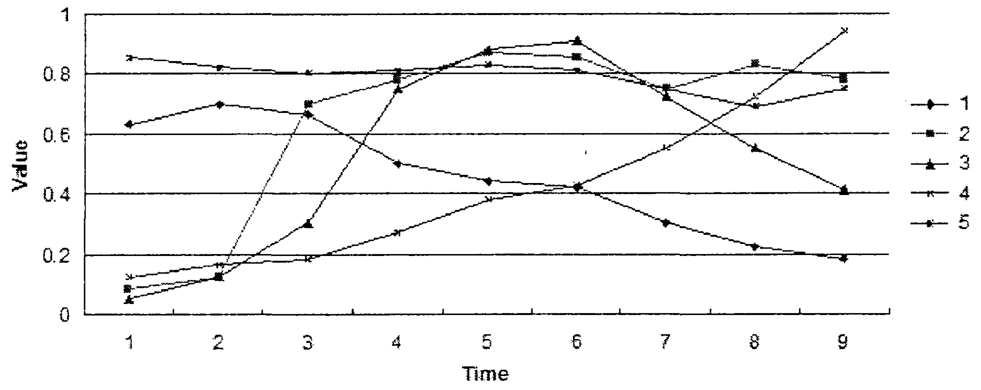
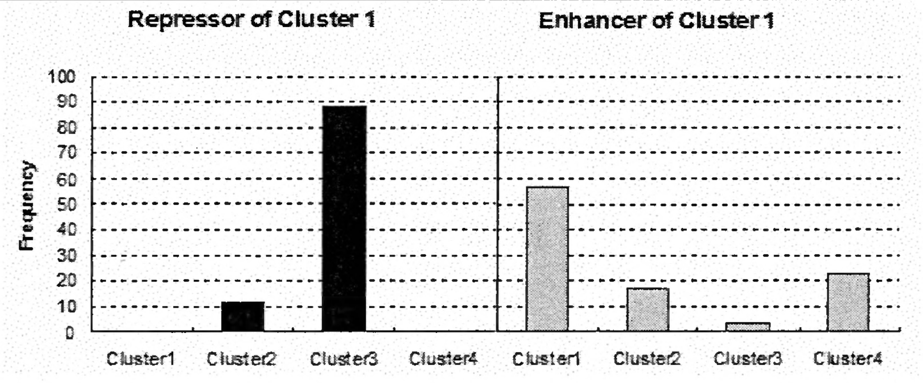
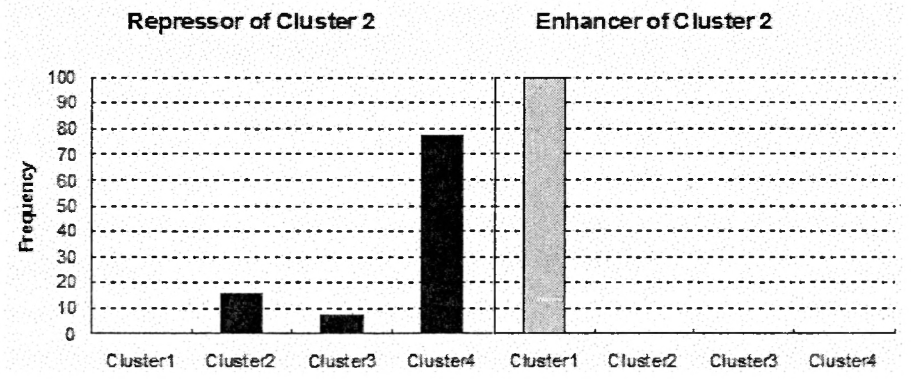


Figure 13 Five Gene Expression Waves [Wen *et al.*, 1998]

In order to avoid the local optimal noise, I run the fitting algorithm 100 times to the Rat CNS data set 1. For this data set, fitting algorithm terminated at in-connectivity 2 where average fitness reaches 0.95. Figure 14 shows the generated statistical distribution pattern of the values of parameters. Because the signal of the parameters clear stands out from noises, we can reconstruct unambiguous gene regulatory network for rat CNS data. Out of the five types of parameters: *indices, weights, biases, decay constants* and *scales*, we are more interested in indices and weights because they are the edges of the regulatory networks. For weights, the allowed parameter range is from -10.0 to 10.0. We can then compute average parameter values from multiple runs. For indices, we simply pick the one that gets majority votes.





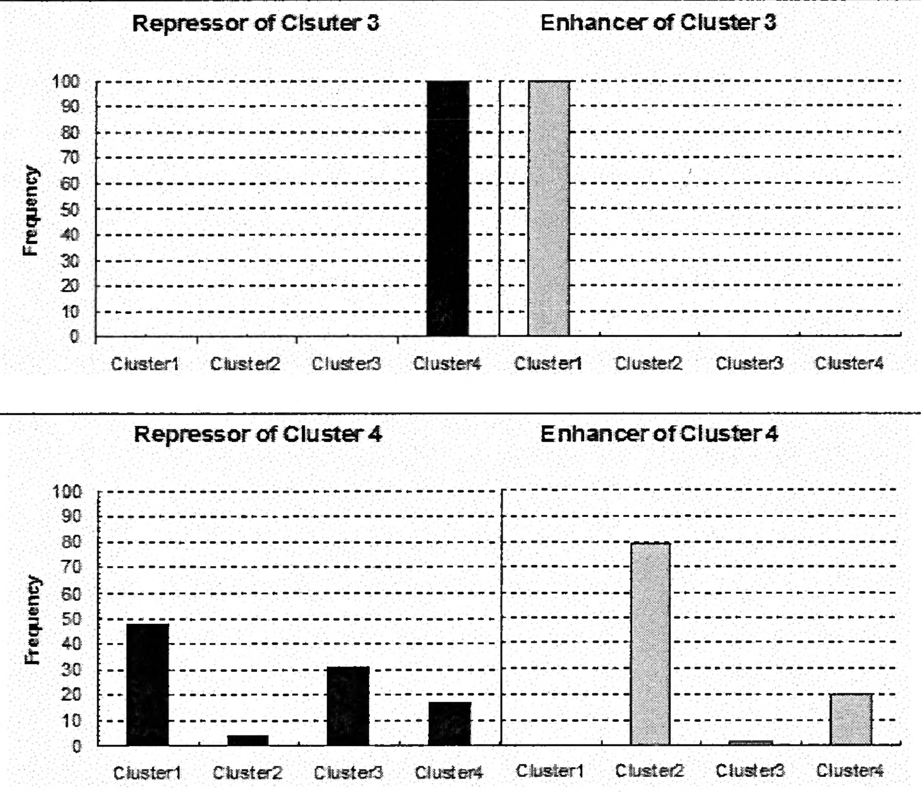


Figure 14 Distribution of *Indices* Parameters of Rat CNS Data

	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Enhancing Weight	3.1 ± 2.5	9.4 ± 0.5	8.9 ± 1.0	7.6 ± 1.7
Repressing Weight	-7.3 ± 1.9	-4.1 ± 1.1	-8.5 ± 1.4	-2.1 ± 2.0

Table 3 Distribution of *Weights* Parameters of Rat CNS Data

In previous chapter, we have shown the incremental modeling and fitting algorithm should have a better performance than a fully connected neural network model [Wahde *et al.,* 2001]. Now from Figure 14 and Table 3, we can see standard errors for the parameter sample are very small. This fact implies the results has higher signal-noise ratio and thus the fitting process pinned down model parameters much more effectively than the method of Wahde *et al.* In addition, our method does not require multiple time series data which the method of Wahde *et al.* does in order to pin down parameters. Thus, theoretically and practically, the method we are proposing is superior to the one presented by Wahde *et al.* Another interesting result is the distribution of *indices* is closely related to the distribution of *weights*. If the distribution of *indices* is crispy, that is, one value dominate the others, its regulation weight tends to be large with small standard error, such as Cluster 1 enhancing Cluster 2. On the other hand, if the distribution of *indices* is ambiguous, its regulation weight tends to be small with big standard error, such as Cluster 1 enhancing itself. In the former case, a regulation pathway can be identified and include in the final results. The latter case indicates there is no or weak regulation in this pathway. Following this analysis, I construct the underlying network which is shown in Figure 15.

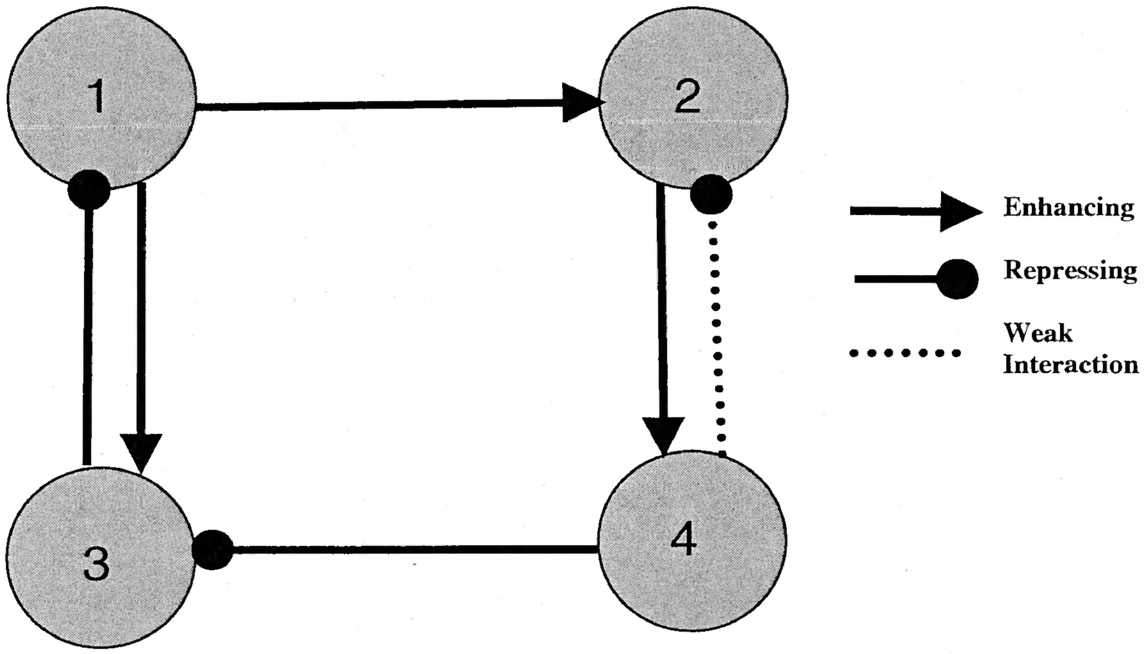


Figure 15 Reverse-Engineered Regulation Network from Rat CNS Data

Comparing my results in Figure 15 with the results of [Wahde *et al.*, 2001]:

1. Cluster 1 has strong up-regulation to Cluster 2, and 3. These regulations have been confirmed in by Wahde *et al.*.

2. Cluster 2 is accounted for up-regulation to Cluster 4 while there is a weak down-regulation from Cluster 4 to 2. Wahde *et al.*, reported the rise in Cluster 4 is driven by Cluster 1 and 3.

3. Cluster 4 has a down-regulation pathway to Cluster 1 through Cluster 3. Wahde *et al.*, also reported Cluster 4 is responsible for lowering expression of Cluster 1 and Cluster 3 but in a direct way.

Apparently, both results agree with some regulations. One major difference is what cluster is responsible for the rising expression level of Cluster 4? Wahde *et al.* suggest Cluster 1 and Cluster 3 are the cause for the rising expression in Cluster 4. However, from Figure 14 and Table 3, Cluster 2 got 79 votes out of 100 and the regulation weight is $7.6_{\pm 1.7}$ out of a possible 10.0. Clearly our results indicate Cluster 2 has a dominant and strong up-regulation effect to Cluster 4.

The generation of the regulatory network is based on the model and the data. It is important to note that the model is a simplification of real complex regulatory systems. We make some assumptions and may miss some important details in the model. The purpose of this method and result network is mainly to present biologists a more possible network rather than a final network. The results may also be biased by the selection of survey genes. A close and complete system is one assumption of the model. Although the 112 genes are important in the CNS development, there are still thousands of genes, which are not included in the study, may also play important roles in the regulatory system.

4.5 Application to Yeast Whole Genome Data

The fast advance of microarray technology allows us to measure expression thousands of genes in a single experiment. In this section, we have used one data set obtained from [Eisen *et al.,* 1998] for budding yeast *S. cerevisiae.* This data set includes all 2467 yeast genes which represent more than 35% of all yeast genes. All the genes have a functional annotation in the Saccharomyces Genome Database [Cherry *et al.*, 1997]. For this microarray, three time series experiments are performed of the mitotic cell division cycle [Spellman *et al.,* 1998], sporulation [Chu *et al.*, 1998], and the diauxic shift [DeRisi *et al.*, 1997]. Eisen *et al.* combined all the experimental data and performed a hierarchical clustering analysis on the data set. They found genes with unrelated sequence but similar functions reveals similar expression patterns. However, a simple clustering analysis cannot reveal the relationship among different clusters. Our goal here is two-folded: 1. uncover the relationships among the clusters for each experiment. 2. compare the networks constructed for each experiment. Hopefully, this may lead us to understand how experiment changes affect the gene regulations.

As we mentioned, clustering is usually the first step when organizing complex or unknown data such as gene expression data. Clustering is also a necessary step in order to perform a quantitative modeling process. Combining all experiments data, we carried out a clustering procedure using a software obtained from Eisen *et al.*. We first use hierarchical clustering to visualize the data set. The data set can be roughly divided into 10 distinct expression patterns. Because hierarchical clustering method can not explicitly separate the data set (the output of a hierarchical clustering is a tree), we then use *k*-mean clustering (set *k* to 10) to automatically divide the data set into 10 clusters. (see Figure 16). Based on clustering analysis, Eisen *et al.*, reported that genes with similar function cluster together. However, by closely examining the yeast genome wide gene expression pattern at http://genome-www.stanford.edu/clustering/Yeast.html, we can see genes with same function are distributed everywhere. On the other hand, one cluster can contain many genes with diverse functions. The correlation between functions and clusters is not significant and there is not a simply one to one mapping between clusters and functions. The sheer complexity of functional genomics cannot be explained by a simple clustering analysis.

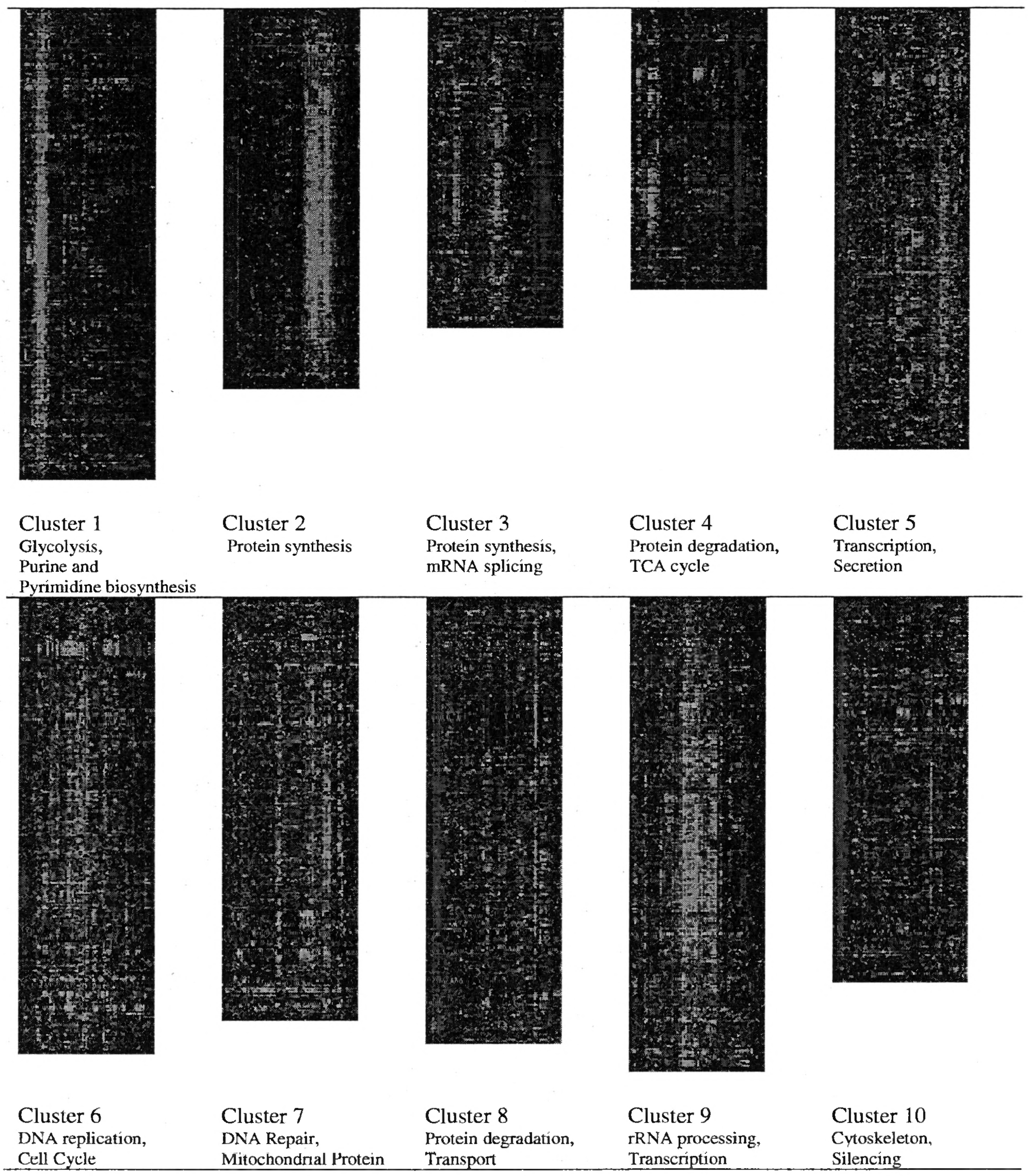


Figure 16 Clustered Display of Combined Yeast Data sets

The whole data sets were clustered into 10 clusters. As we can see in Figure 16, each cluster has a distinct expression pattern. Major functions associated with each cluster were listed.

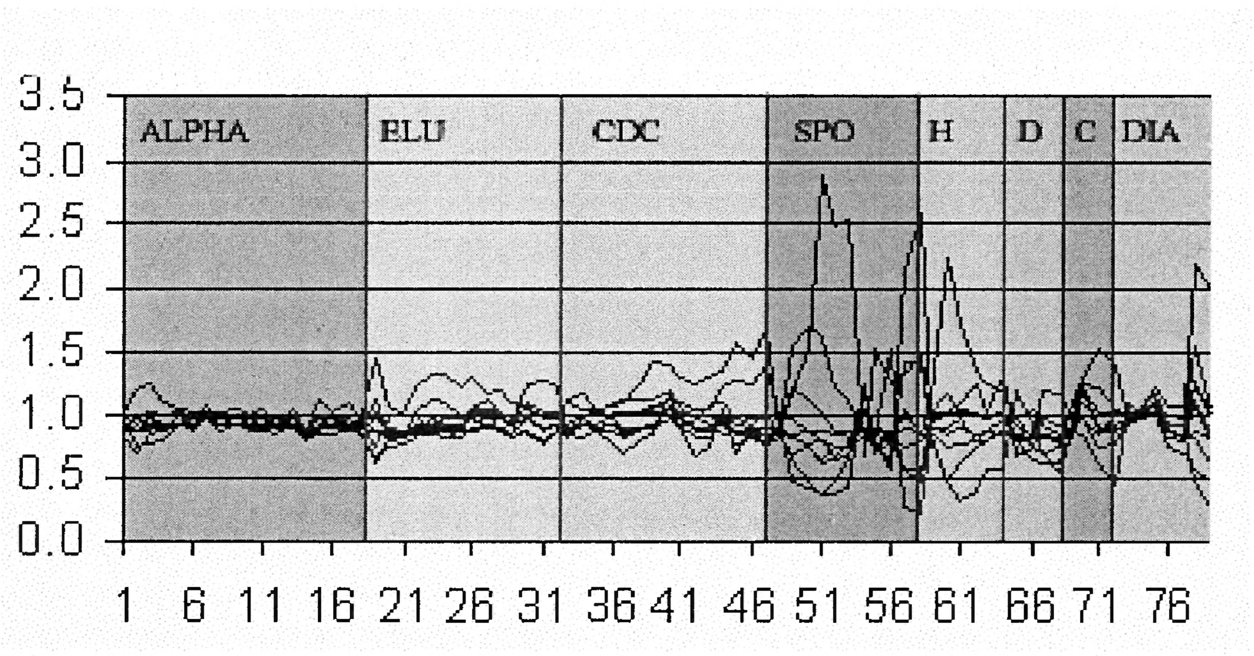


Figure 17 Expression Patterns of 10 Cluster Medians during Multiple Experiments

After performing the clustering procedure, we would like to see the expression patterns for each cluster. Because the original data are logged value, we transformed all the data to their base 2 exponentials. Then we extract medians of the 10 clusters respectively. Expression patters for the 10 cluster medians during multiple experiments are shown in Figure 17. We can see in the first three time series ALPHA, ELU, and CDC, the expression patterns for each of the 10 medians are very stable. These three time series are longest in all 8 data sets. However, unfortunately these nearly constant data do not contain much information which we may use to fit the model. In the other five time series: SPO, H, D, C, and DIA, expression patterns for each of the 10 medians are dynamic. However, H, D, and C contain 6, 4, and 4 data points respectively. Referring to our test of modeling performance, these time series are too short to fit a model with 10 equations. Only two choices left, SPO and DIA. Diauxic shift (DIA) [DeRisi *et al.*, 1997] data set contains a time series of length 7 (Figure 19). Sporulation (SPO) [Chu *et al.,* 1998] data set contains a time series of length 11. However, the last two data points cannot be included in our analysis because they are under different experiment settings with the rest data points. In addition, data points 7, 8, and 9 has different reference sample from the others, so that we transformed these data to the same reference sample. We then combine all the time points shown in Figure 18.

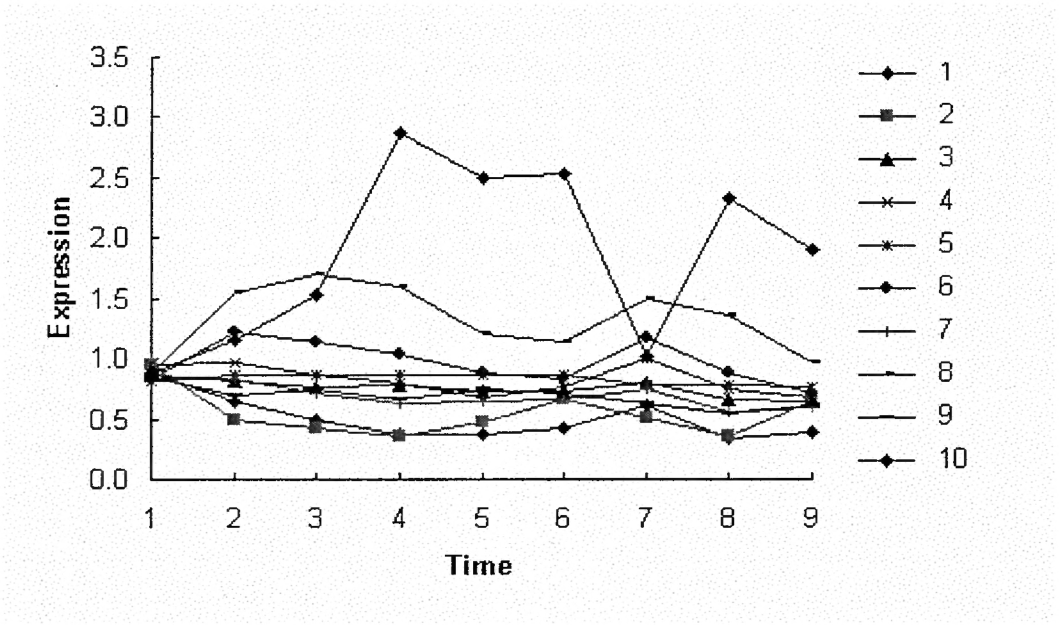


Figure 18 Expression Patterns of 10 Clusters during Sporulation

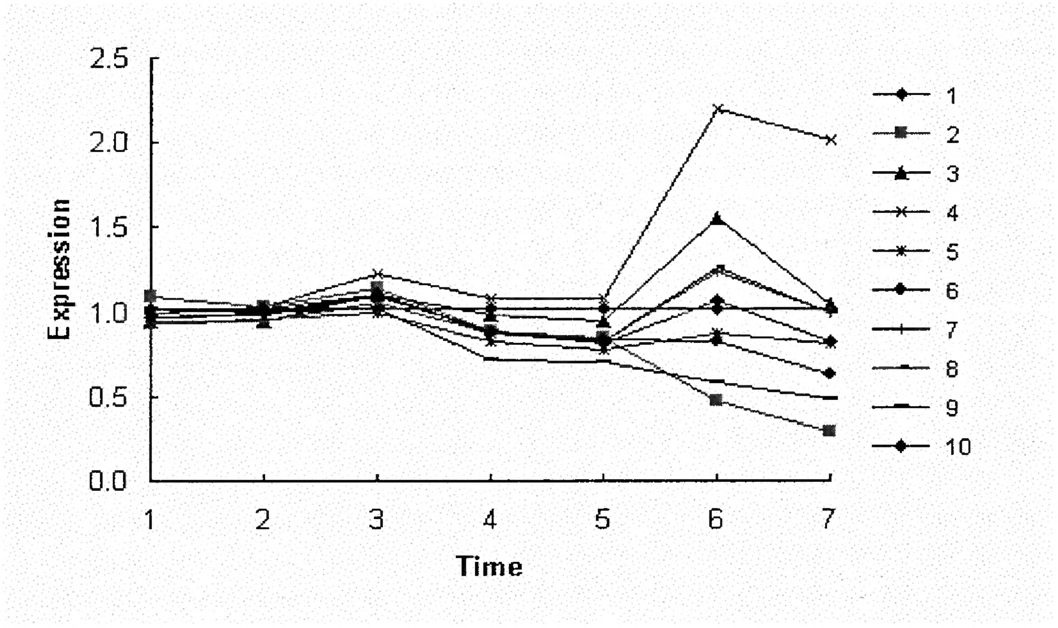


Figure 19 Expression Patterns of 10 Clusters during Diauxic Shift

The difference of expression patterns between DIA and SPO indicates different regulation mechanisms during these two processes. Comparing the two reconstructed networks may help us to understand the change of regulation caused by environment change. After preprocessing the data sets, we then apply our fitting algorithm to the two time series respectively. The procedure and settings are same as what we applied to the Rat CNS data set. The results are shown in Figure 20 and Figure 21.

The *Parameter* axis *r1, e1,..., r10, e10* include all *Indices* parameters. Each parameter has a name indicating its role. For example, *r1* is repressor of cluster 1, *e7* is enhancer of cluster 7, *etc.* Each parameter has a distribution over all 10 cluster indices. Each figure has two views to provide more clarity. If there is no standing out column in the distribution of a parameter, the parameter is poorly identified. This means this parameter is probably null. On the other hand, if there is some really high column dominating a distribution, the parameter is very likely identified with confidence. If the frequency of an index occurrence is larger than 40% from a distribution, then we classify it as an identified regulation. Then based on those identified parameters we can construct the underling regulatory networks (see Figure 22, 23).

From Figure 22 of DIA, we can summarize the regulation as the followings:

1. Clusters 2 and Cluster 5 play major roles in regulating the system during DIA. Cluster 9 and 10 has only one out-going edges. The other six clusters have no out-going regulating edges so that they are very silent in terms of regulating. We can also see this in ***b*** view of Figure 20.
2. Cluster 2 has 7 out-going edges and 5 of them are down-regulation. Cluster 5 has 6 out-going edges. 3 of them are up-regulation 3 of them are down-regulation.

From Figure 23 of SPO, we can summarize the regulations as the followings:

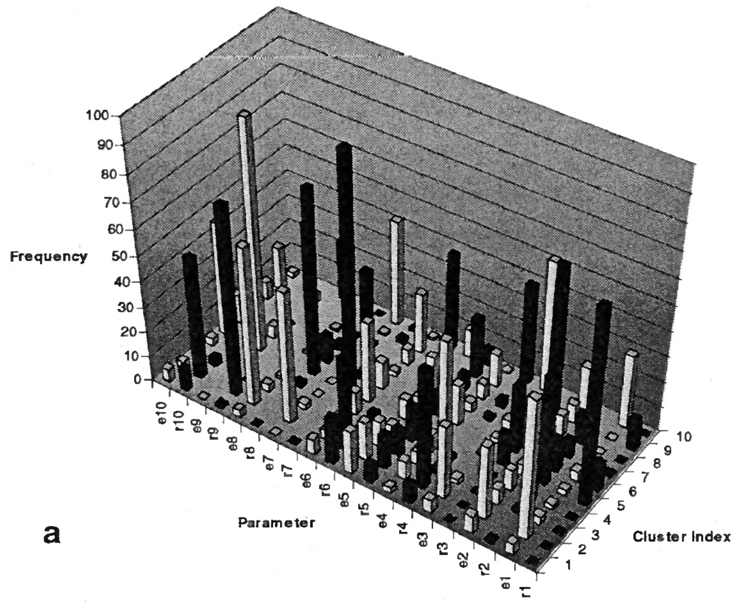
1. Cluster 1, 5, 8 and 10 plays important roles in the system during SPO. Other 6 clusters has no out-going edges. We can also see this result in ***b*** view of Figure 21.
2. Cluster 6 and 7 has exactly same regulation connections. Their expression patterns are also similar during SPO.

3. Cluster 1 has 5 out-going edges, 3 of them are down-regulations and 2 of them are up regulations. Cluster 5 has 5 out-going edges, 4 of them are up-regulations. Cluster 8 has 3 out-going edges with all of them are down-regulations.

Because the expression patterns are very different between DIA and SPO, the generated networks also have lots of difference. Comparing the two networks, we can get more interesting results.

1. From ***b*** view of DIA results, we can see cluster 2, 5, 9 and 10 are very actively regulating the network. From ***b*** view of SPO results, we can see cluster 1, 5, 8 and 10 are very actively regulating the network. Cluster 2 and 9 are very silent during the SPO process. On the other hand, Cluster 1 and 8 are also very silent during DIA process in terms of regulating.
2. Cluster 3, 4, 6, 7 work passively in both environments. They are simply receivers of the regulations. They do not have out-going edges at all. Interestingly, their expression level changes dynamically if we look at Figure 19. This phenomenon tells us one thing: the most dynamic one does not necessarily have to be the most active one. Expression level change is the result of the regulation, not the cause of the regulation.

In summary, the application of this method to rat CNS and yeast genome wide data sets does show its capability to mining knowledge from data. And the knowledge is not obviously displayed in the data.

a


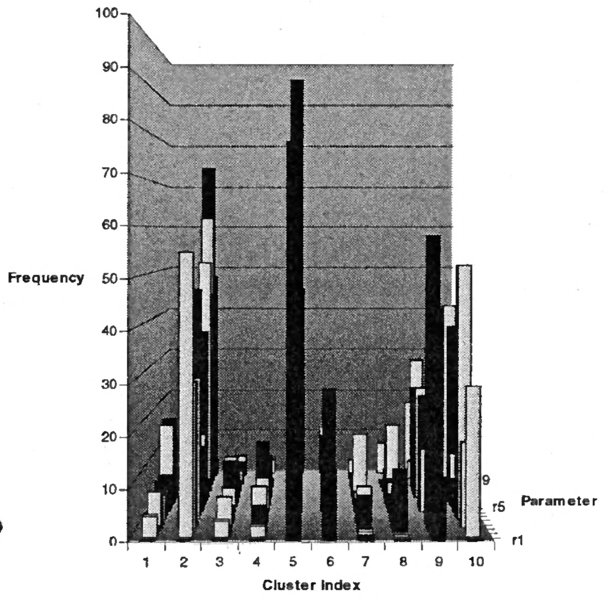
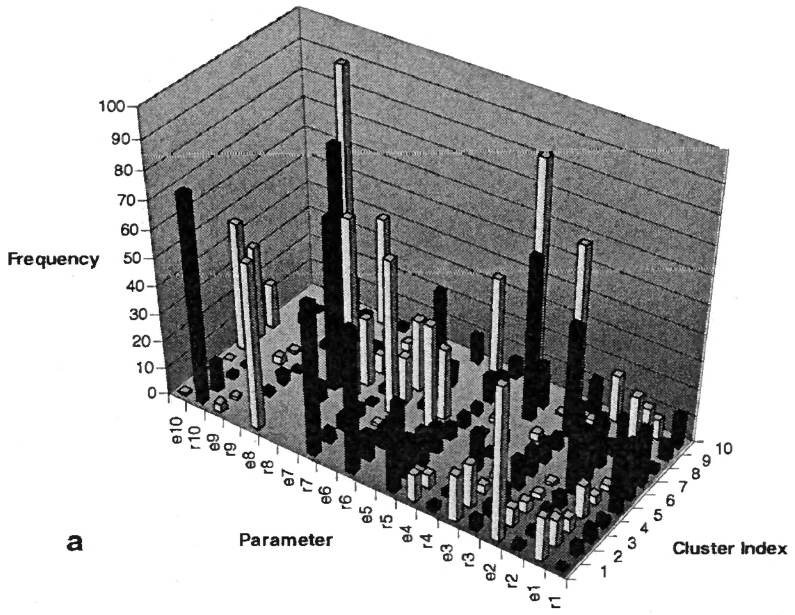
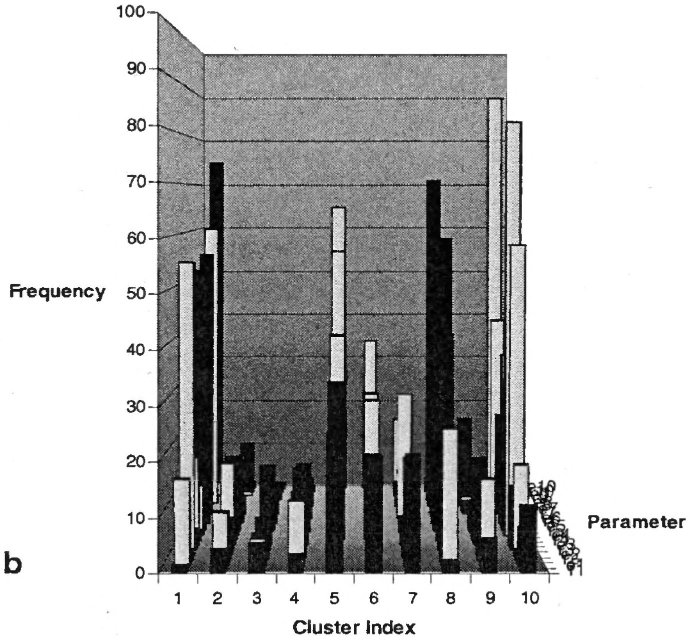
b


Figure 20 Distribution of *Indices* Parameter of Yeast DIA Data



a



b

Figure 21 Distribution of Indices Parameter of Yeast SPO Data

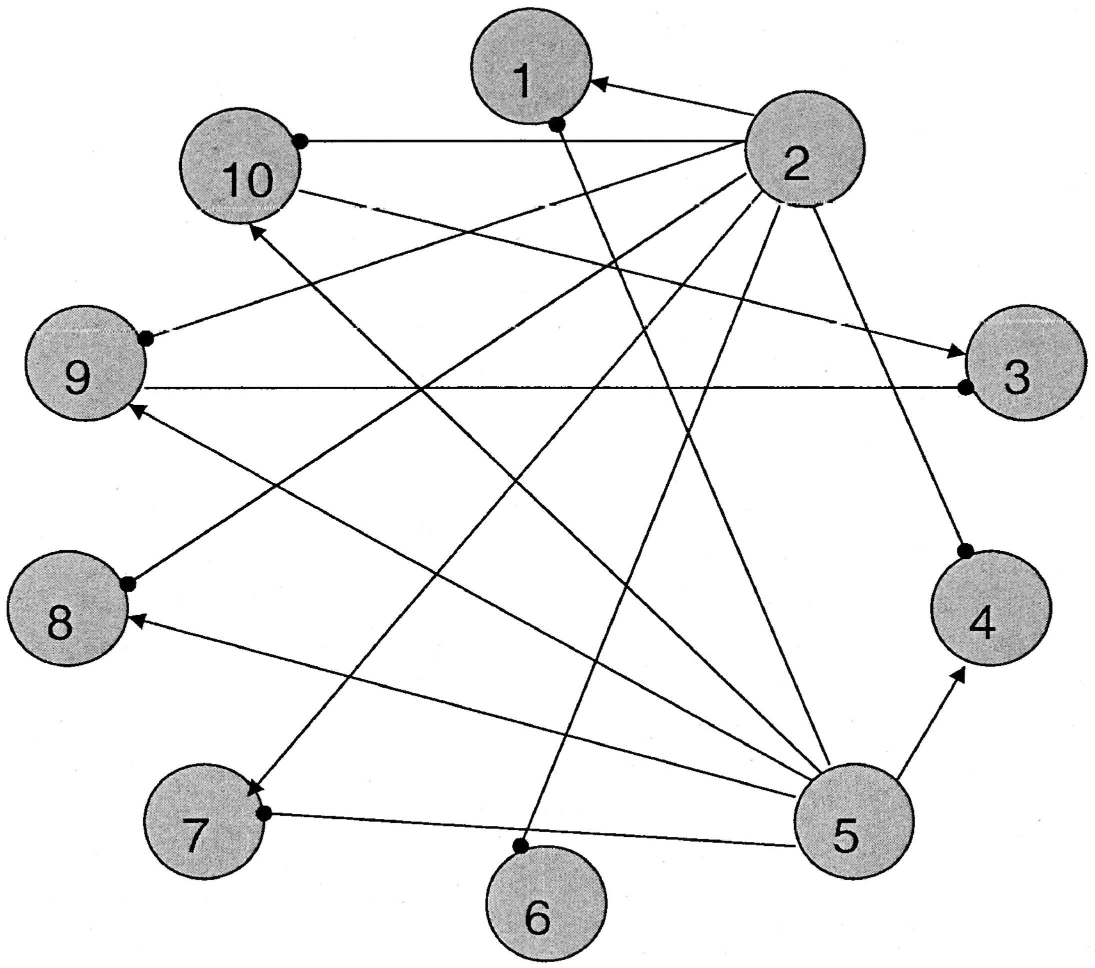


Figure 22 Reverse-Engineered Regulation Network from Yeast DIA Data

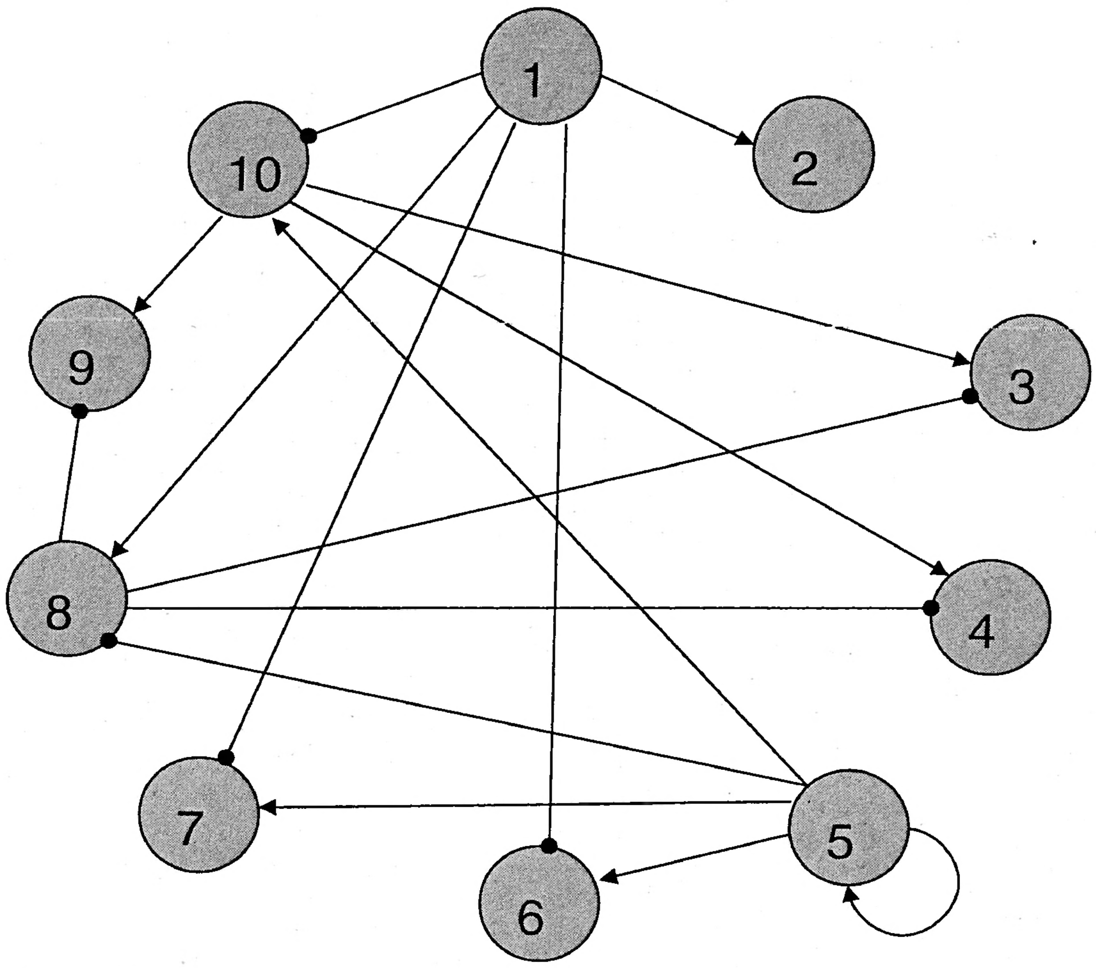


Figure 23 Reverse-Engineered Regulation Network from Yeast SPO Data

4.6 Conclusion

Linear Additive Models (LAMs) are one category of mathematic models which are applied to reconstruct gene regulatory networks from time series expression data sets. We pointed out a parameter problem associated with LAMs: all LAMs contain a fully connected network, which has too many parameters to be effectively pinned down. We proposed a new incremental connectivity model and a fitting algorithm to overcome the problem. By using this method, the number of parameters can be reduced by $O(N)$ and the minimal solution can be obtained. We provided a guideline for LAM users by systematically testing modeling performance of randomly generated networks. We also obtained satisfactory modeling performance by testing real expression data sets.

For known network, we define graph similarity to measure the modeling performance. For unknown network, we use certain statistic, such as p-value or variance to measure the modeling performance. These measurements can also be used to compare performance of different methods.

Because gene expression data sets, especially microarray data sets are highly noisy, overfitting the system would lead to fitting to noise; therefore, finding a near optimal solutions may be more appropriate than finding the optimal one. In such a scenario, a GA would be used to find near optimal solutions. When the number of elements increases, the number of near optimal solutions also increases. This may explain why we got very low modeling performance for large scale systems. Note that low modeling performance for large scale network does not mean no solutions found but mean too many near optimal solutions (fitness>0.95) found. Fitness threshold is set 0.95 to ensure high performance of fitting algorithm. If the threshold is too low, the solution tends to be divergent. On the other hand, if the threshold is set too high, such as 0.99, the algorithm may overfit the data. A reasonable threshold will tolerate noise and avoid overfitting problem.

We are facing a dilemmatic situation for large scale system: with large length of time series, the system may be determined but the noise in data will made the results unreliable. On the other hand, if we tolerate the noise to some degree, the large number of solutions will make the inference impossible. This is an intrinsic problem, therefore searching algorithms other than GA would not likely to improve the performance. Indeed, D'haeseleer (2000) reported very high condition number ($6.3 \cdot 10^4$) obtained from fitting a 65-node gene network using linear regression method. This fact implies a relatively large system can be largely underdetermined and thus the model fitting performance may be pretty low. The only way to improve the performance is to integrate other information with gene expression profile to constrain the system search space. One recent study by Gardner *et al.* (2003) reported integrating perturbations experiment design with linear additive models successfully predicted SOS pathways. However, there are no demonstrated utility of such methods to large scale network reconstruction. In addition, these methods needs specially designed data sets which are not the case of most current data bases.

As we knew, this is the first study to explore the utility of LAMs in data mining time series gene expression data sets. Because our method has reduced search space than other LAMs, it is not likely other LAMs will outperform this one. Therefore, our results represent not only an example of LAMs, but also a generalization of current LAM technology.

We found LAMs can only deal with small networks. One may wonder whether brute force method can be used to tackle the optimal solution. Because of the sheer complexity of biological system, the answer will be no. For example, for a small network with 10 elements, with each node having 6 parameters and each parameter having 10 possible values, the total possible values will be 10^{60} which is beyond the power current computers.

Because GA or other optimization tools are not a factor in determining modeling performance, the only constraint of the modeling performance is the size of search space. In the near future, we will investigate integrating promoter sequence data with expression data to constrain the system search space. Moreover, we will consider adding stochastic feature to the model and designing a domain specific fitting algorithm to make this method more accurate and powerful. At this step, we believe this method a useful tool for biologists who are interested in design and analysis of expression experiments.

Reference:

1) Ando, S., and Iba, H., Quantitative Modeling of Gene Regulatory Network – identifying the Network by Means of Genetic Algorithms, *Poster Session of Genome Informatics Workshop* 2000
2) Akutsu, T., Miyano, S., and Kuhara, S., Identification of genetic networks from a small number of gene expression patterns under the Boolean network model *PSB*'99
3) Bittner M., Meltzer P., and Trent J., Data analysis and integration: of steps and arrows. *Nat Genet.* 1999 Jul;22(3):213-5.
4) Chen, T., He, H.L., Church., and G. M., Modeling gene expression with differential equations. *PSB* '99, pages 29—40.
5) Cherry, J.M., Ball, C., Weng, S., Juvik, G., Schmidt, R., Adler, C., Dunn, B., Dwight, S., Riles, L., Mortimer, R. K., *et al*. Genetic and physical maps of Saccharomyces cerevisiae. (1997) *Nature (London)* 387, 67–73.
6) Chu, S., DeRisi, J., Eisen, M., Mulholland, J., Botstein, D., Brown, P. O. and Herskowitz, I., The transcriptional program of sporulation in budding yeast. *Science* 1998 282, 699–705.
7) De Jong K. A. (1993). *Genetic Algorithms Are NOT Function Optimizers*. In L. Darrell Whitley (ed.), Foundations of Genetic Algorithms 2, 5-17. CA: Morgan Kaufmann.
8) DeRisi, J. L., Iyer, V. R. & Brown, P. O., Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 1997, 278, 680–686.
9) D'haeseleer, P., *Reconstructing Gene Networks from Large Scale Gene Expression Data,* Ph.D. dissertation, University of New Mexico, 2000.
10) D'haeseleer, P., Liang, S., and Somogyi, R., Genetic Network Inference: From Co-Expression Clustering to Reverse Engineering. *Bioinformatics* 2000, 16(8):707-26,.
11) Eisen, M.B., Spellman, P.,T., Brown, P.O., and Botstein, D., Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci* 1998, Dec 8; 95(25): 14863-14868.
12) Ekins, R., and Chu, F.W., Microarrays: their origins and applications. *Trends in Biotechnology*, 1999, 17, 217-218.
13) Everett, J.E. Model building, model testing and model fitting. *The Practical Handbook Of Genetic Algorithms : Applications* 2001 CRC edited By Lance Chambers
14) Friedman, N., Linial, M., Nachman, I., and Pe'er, D. Using Bayesian networks to analyze expression data. *J. Comput. Biol.* 2000, 7, 601-620.
15) Garrell, J., Campuzano, S., The helix-loop-helix domain: a common motif for bristles, muscles and sex. *Bioessays.* 1991 Oct;13(10):493-8.
16) Hieter, P., and Boguski, M., Functional genomics: it's all how you read it, *Science* 1997, 278, 601-602.

17) Holland, J. H. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor. 1975

18) Kahn, P., From genome to proteome: Looking at cell's proteins. *Science* 1995 270, 369-370.

19) Kauffman, S.A., *The Origins of Order: Self-Organization and Selection in Evolution,* Oxford University Press, New York. 1993

20) Liang, S., Furman, S., and Somogyi, R., REVEAL, a general reverse engineering algorithm for inference of genetic network architectures, *PSB'98* 3-18

21) Liang, P., and Pardee, A.B., Differential display of eukaryotic messenger RNA by means of the polymerase chain reaction. *Science* 1992 257, 967-971.

22) Mjolsness, E., Sharp, D.H., and Reinitz, J. A connectionist model of development. *J. THeor. Biol.,* 1991, 152, 429-454.

23) Reinitz, J., and Sharp, D.H., Mechanism of *eve* stripe formation. *Mech Dev* 1995 49: 133-158.

24) Schena, M., Shalon, D. Davis, R.W., and Brown, P.O., Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science* 1995 270, 5325, 467-470.

25) Spellman, P. T., Sherlock, G., Iyer, V. R., Zhang, M., 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*, 1998 Vol. 9, 3273-3297.

26) Szallasi, Z., Genetic Network Analysis in Light of Massively Parallel Biological Data Acquisition *PSB'99* 4:5-16.

27) Wahde, M. and Hertz, J. Modeling Genetic Regulatory Dynamics in Neural Development, *Journal of Computational Biology,* 2001, 8, 429-442,

28) Wahde, M., and Hertz, J. *Coarse-grained Reverse Engineering of Genetic Regulatory Networks, BioSystems,* 2000,55, pp. 129-136,

29) Wall, M. Genetic Algorithms Library 2.4.5 http://lancet.mit.edu/ga/.

30) Weaver, D.C., Workman, C.T. and Stormo, G.D. Modeling regulatory networks with weight matrices. *PSB'99* 4:112-123.

31) Wen, X., Fuhrman, S., Michaels, G.S., Carr, D.B., Smith, S., Barker, J.L., and Somogyi, R., Large-scale temporal gene expression mapping of central nervous system development. *Proc Natl Acad Sci* U S A 1998 Jan 6;95(1):334-9

32) Whitley, D., A genetic algorithm tutorial. *Statistics and Computing* 1994 4: 65-85.

33) Zeigler, B.P., *Theory of Modeling and Simulation.* John Wiley, New York. 1976