INFERRING REGULATION PROGRAMS IN A TRANSCRIPTION REGULATORY MODULE NETWORK

JIANLONG QI

A THESIS
IN
THE DEPARTMENT
OF
COMPUTER SCIENCE AND SOFTWARE ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
CONCORDIA UNIVERSITY
MONTRÉAL, QUÉBEC, CANADA

SEPTEMBER 2011

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By:	Mr. Jianlong Qi
Entitled:	Inferring regulation programs in a transcription regulatory module network

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

______________________________ Chair
Dr. A. Hanna

______________________________ External Examiner
Dr. M. T. Hallett

______________________________ Examiner
Dr. V. Chvatal

______________________________ Examiner
Dr. N. Shiri

______________________________ Examiner
Dr. A. Tsang

______________________________ Supervisor
Dr. Gregory Butler

Approved ______________________________
Chair of Department or Graduate Program Director

__________ 20 ______ ______________________________
Dr. Robin A. L. Drew, Dean
Faculty of Engineering and Computer Science

Abstract

Inferring regulation programs in a transcription regulatory module network

Jianlong Qi, Ph.D.

Concordia University, 2011

Cells have a complex mechanism to control the expression of genes so that they are capable of adapting to environmental changes or genetic perturbations. A major part of the mechanism is fulfilled by transcription factors which can regulate the expression of other genes. Transcriptional regulatory relationships between genes and their transcription factors can be represented by a network, called a transcription regulatory network.

Many algorithms have been proposed to learn transcription regulatory networks from gene expression data. In particular, the module network method, a special type of Bayesian networks, has shown promising results. In a module network, a regulatory module is a set of genes that show similar expression profiles and are regulated by a shared set of transcription factors (i.e., the regulation program of the module). This method significantly decreases the number of parameters to be learned. Module network learning consists of two tasks: clustering genes into modules and inferring the regulation program for each module. This thesis concentrates on designing algorithms for the latter task.

First, we introduce a regression tree-based Gibbs sampling algorithm for learning regulation programs in module networks. The novelty of this method is that a set of tree operations is defined for generating new regression trees from a given tree. We show that the set of tree operations is sufficient to generate a well mixing Gibbs sampler even for large datasets.

Second, we apply linear models to infer regulation programs. Given a gene module, this method partitions all experimental conditions into two condition clusters, between which the module's genes are most differentially expressed. Consequently, the process of learning the regulation program for the module becomes one of identifying transcription factors that are also differentially expressed between these two condition clusters.

Third, we explore the possibility of integrating results from different algorithms. The integration methods we select are union, intersection, and weighted rank aggregation. The experiments in a yeast dataset show that the union and weighted rank aggregation methods produce more accurate predictions than those given by individual algorithms, whereas the intersection method does not yield any improvement in the accuracy of predictions. In addition, somewhat surprisingly, the union method, which has a much lower computational cost than rank aggregation, archives comparable results as given by rank aggregation.

Dedication

I would like to thank most of all my supervisor, Dr. Gregory Butler, for his continued academic and financial support without which this work would not have been completed. His enlightening discussions and suggestions helped greatly to improve the thesis quality. In addition, I would like to thank Dr. Tom Michoel from Freiburg Institute for Advanced Studies for his help on my research. I would also like to thank Dr. Adrian Tsang for his valuable suggestions on my research. I would also like to thank Dr. Yuanzhu Chen from Memorial University of Newfoundland for his help on the revision of this thesis and valuable suggestions on my defence.

Lots of thanks to all my lab mates: Christine Kehyayan, Hajar Sadr, Byakuya Otaku, Yue Wang, Stephen Barrett, and Jun Luo for enjoyable time in our lab. I would like to give many thanks to LeGym where I get refreshed after a day's work in the lab.

My sincere thanks go to my parents and my in-laws for their enthusiastic support, kindness and understanding during these busy years. During my PhD. studies, my daughter Claire was born. I would like to thank her for granting me great joy. Last, I would like to dedicate this thesis to my wife. Without her encourage and support, I would not have finished my studies.

Contents

List of Figures

List of Tables

Glossary

DNA	Deoxyribonucleic acid
mRNA	messenger ribonucleic acid
BAC	bacterial artificial chromosome
EMBL	European molecular biology laboratory
MF	Molecular function
BP	Biological process
CC	Cellular component
DAG	Directed acyclic graph
HMM	Hidden Markov model
cDNA	Complementary deoxyribonucleic acid
SAGE	Serial analysis of gene expression
DGE	Digital gene expression
RPKM	Reads per kilobase per million reads sequenced
FPKM	Fragments per kilobase of transcript per million fragments sequenced
SGD	*Saccharomyces* genome database
GSEA	Gene set enrichment analysis
CLR	Context likelihood of relatedness
MLE	Maximum likelihood estimation
MAP	Maximum a posteriori
EM	Expectation maximization
NCR	Nitrogen catabolite repression
EST	Expressed sequence tag
JGI	Joint genome institute
HWKP	Hardwood kraft pulp

SWMP	Softwood mechanical pulp
MIG1	Gene name in *Saccharomyces cerevisiae*
SNF1	Gene name in *Saccharomyces cerevisiae*
RGT1	Gene name in *Saccharomyces cerevisiae*
HXT3	Gene name in *Saccharomyces cerevisiae*
HXT1	Gene name in *Saccharomyces cerevisiae*
GAL4	Gene name in *Saccharomyces cerevisiae*
GAL7	Gene name in *Saccharomyces cerevisiae*
GAL3	Gene name in *Saccharomyces cerevisiae*
GAL10	Gene name in *Saccharomyces cerevisiae*
GAL80	Gene name in *Saccharomyces cerevisiae*
GCN4	Gene name in *Saccharomyces cerevisiae*
DAL80	Gene name in *Saccharomyces cerevisiae*
GLN3	Gene name in *Saccharomyces cerevisiae*
MET28	Gene name in *Saccharomyces cerevisiae*
DAL2	Gene name in *Saccharomyces cerevisiae*
DAL3	Gene name in *Saccharomyces cerevisiae*
GAT1	Gene name in *Saccharomyces cerevisiae*
PLP2	Gene name in *Saccharomyces cerevisiae*
MBP1_SWI6	Gene name in *Saccharomyces cerevisiae*
SWI4_SWI6	Gene name in *Saccharomyces cerevisiae*
SPT16	Gene name in *Saccharomyces cerevisiae*
ACE2	Gene name in *Saccharomyces cerevisiae*
TUP1	Gene name in *Saccharomyces cerevisiae*
ALPHA1	Gene name in *Saccharomyces cerevisiae*
REB1	Gene name in *Saccharomyces cerevisiae*
A1_ALPHA2	Gene name in *Saccharomyces cerevisiae*
GAL11	Gene name in *Saccharomyces cerevisiae*
SNF2_SWI1	Gene name in *Saccharomyces cerevisiae*
UGA3	Gene name in *Saccharomyces cerevisiae*
HAP4	Gene name in *Saccharomyces cerevisiae*
GSM1	Gene name in *Saccharomyces cerevisiae*

USV1	Gene name in *Saccharomyces cerevisiae*
STE5	Gene name in *Saccharomyces cerevisiae*
MET32	Gene name in *Saccharomyces cerevisiae*
GZF3	Gene name in *Saccharomyces cerevisiae*
URE2	Gene name in *Saccharomyces cerevisiae*
flhC	Gene name in *Escherichia Coli*
flhD	Gene name in *Escherichia Coli*
fliA	Gene name in *Escherichia Coli*
creA	Gene name in *Aspergillus niger*
xlnR	Gene name in *Aspergillus niger*
xyrA	Gene name in *Aspergillus niger*
xlnD	Gene name in *Aspergillus niger*
aguA	Gene name in *Aspergillus niger*

Chapter 1

Introduction

1.1 Transcription regulatory network

Living organisms are built by cells which are mainly made of proteins. Deoxyribonucleic acid (DNA) encodes the complete genetic information for protein synthesis that consists of two stages: transcription and translation. In the transcription stage, a gene, which is a strand of DNA molecule in the nucleus, is transcribed to a messenger ribonucleic acid (mRNA), and then this mRNA is translated to a protein in the translation stage. This entire process represents the *central dogma* of molecular biology.

Taking the information contained in genes and turning that information into proteins is called gene expression. Cells have a complex mechanism that controls the expression of genes so that they are able to express varied combinations of genes in response to environmental changes or genetic perturbations. Generally, this mechanism consists of two levels of controls: post-transcriptional regulation, and transcriptional regulation. The former controls protein synthesis after synthesis of RNA has begun, while the latter controls which genes are transcribed into mRNA.

A major part of transcriptional regulation is fulfilled by transcription factors, which are a specific type of proteins and are capable of either enhancing or inhibiting the expression of other genes by binding to binding sites (i.e., cis-regulatory DNA sequence elements) normally located in the upstream of these genes (Figure 1). Transcription factors often work together to regulate gene expression. In addition, transcription factors are themselves proteins, and are thus subject to transcriptional regulations accomplished by other transcription factors. This results in chains of transcriptional regulations.

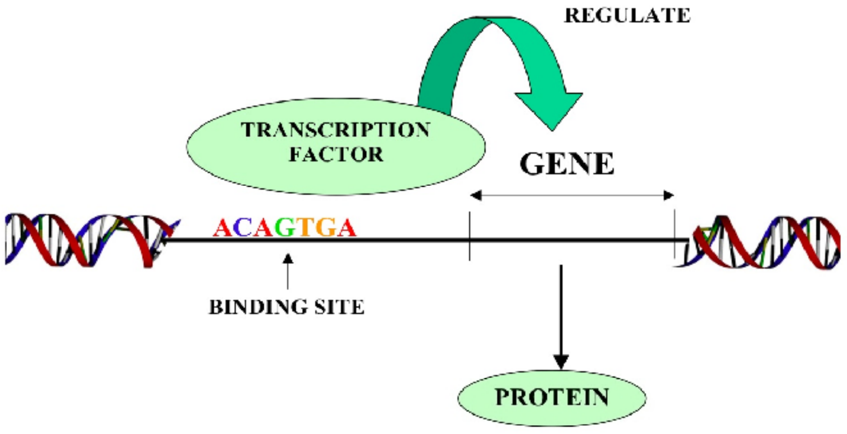

Figure 1: Regulatory relationship between a transcription factor and a gene [111].

The collection of transcriptional regulatory relationships in a cell can be represented by a network, where vertices and edges denote genes and transcriptional regulations, respectively. Such a network is referred to as a ***transcription regulatory network***. Figure 2 shows the regulatory network of 106 transcription factors and their targets in the budding yeast *Saccharomyces cerevisiae (S. cerevisiae)* [69].

Identifying transcription regulatory networks is critical, because it facilitates understanding biological processes in cells. For example, the preferred carbon source for the budding yeast is glucose, but yeast is capable of utilizing many other carbon sources. Experimental results have shown that the genes expressed by yeast are different under various levels of glucose [17]. Some genes (e.g., transporters of glucose) are induced by glucose, but another set of genes, such as genes involved in the utilization of alternate carbon sources, are repressed by glucose. The regulation of gene expression in response to varying carbon sources in the environment is primarily controlled by the transcription regulatory network in the yeast. In this thesis, we concentrate on designing methods for inferring transcriptional regulatory relationships between genes.

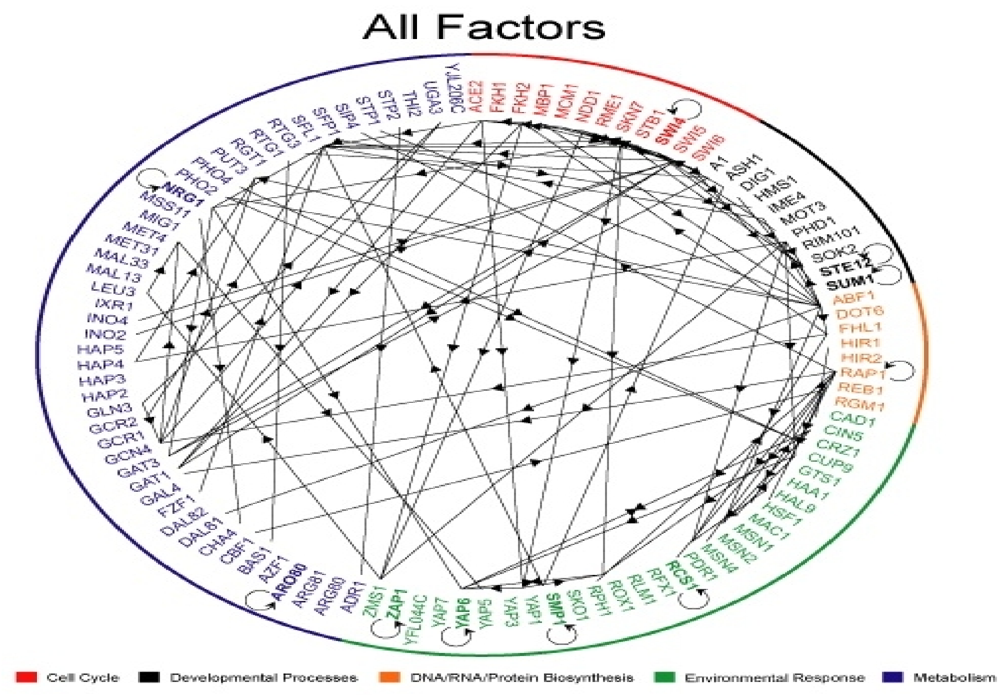

Figure 2: Transcription Regulatory Network in the budding yeast [69].

1.2 The research objective and contributions

Gene expression data, which measure mRNA levels of genes, are widely used for inferring transcription regulatory networks. Expression data can be collected under static or time series experiments. In a static experiment, expression data are measured under a particular condition. In contrast, time series expression date are collected in predefined time points after changes in the environment. In this thesis, we focus on designing methods to infer regulatory networks from expression data collected under static experiments. However, we should notice that time series data are also valuable for the reconstruction of regulatory networks. For example, the binding activity of transcription factors to their targets can be represented by a temporal process.

The reconstruction of regulatory networks from gene expression data is based on the following idea: molecular interactions (i.e., regulatory relationships) between transcription factors and their targets might lead to corresponding correlations between their expression values [92].

Many algorithms have been proposed to learn transcription regulatory networks from gene expression data, including information-theoretic approaches [35], linear regression [11] and clustering algorithms [33]. Given an expression dataset, these algorithms first generate an ordered list of potential regulator-gene interactions according to their significance. Then, only interactions with significance more than a given threshold are considered as true regulatory relationships. In this thesis, we only focus on the first task. The selection of a threshold will be left for future work.

In particular, Bayesian networks [39] have shown promising results. The major advantage of the Bayesian network method is that it studies the joint probability distribution over the expression values of a set of genes, instead of evaluating pairwise correlations. In addition, the method detects the dependence structure between genes which is similar to the structure of regulatory networks. One limitation of the method is that regulatory networks consist of feedloops and self-regulations which can not be represented by directed acyclic graphs. In addition, the models inferred by standard Bayesian networks often overfit data, because the number of parameters to be learned is enormous compared to the number of samples (experimental conditions) in a typical gene expression dataset. To overcome this issue, the module network method [105], a special type of Bayesian networks, has been proposed. The module network method, groups genes with similar expression profiles into regulatory modules, and consequently reduces the number of parameters to be learned.

Module network learning consists of two tasks: clustering genes into modules, and inferring a regulation program for each module. The regulation program of a gene module includes one or several transcription factors, which regulate the transcription of genes in this module. Segal $et\ al.$ [106] applied the expectation maximization algorithm [28] to alternate between these two tasks. Furthermore, Michoel $et\ al.$ [62] enhanced the learning procedure by introducing a two-step method, which separates clustering genes into modules and learning the regulation program for each module. That is, they grouped genes into modules *before* learning the regulation program of each module. Experimental results showed that the separation improves the performance of the module network method in inferring regulatory networks.

This research focuses on the second step in module network learning, i.e., inferring the regulation program (transcription factors) for a given gene module. This thesis presents three methods for this task.

The first method is a regression tree-based Gibbs sampling algorithm [99]. The regulation program of a module is normally learned by a deterministic search [106], and the major shortcoming of this search algorithm is that its result may only represent one of several possible regulation programs. In order to account for the model uncertainty, we propose a regression tree-based Gibbs sampling algorithm. With the Gibbs sampler, we build a Markov chain whose stationary distribution is the posterior probability of regression trees given the data. To build the chain, we define the neighborhood of a given regression tree using a set of tree operations and transition probabilities for sampling a new regression tree from the neighbourhood. Moreover, we show that the set of tree operations is sufficient to generate a Gibbs sampler with well mixing rate even for large datasets.

The second method applies linear models to inferring regulation programs [100]. Given a gene module, this method groups all experimental conditions into two condition clusters, between which the module's genes are most significantly differentially expressed. Consequently, the process of learning the regulation program for the module becomes one of identifying transcription factors that are also differentially expressed between these two condition clusters. The contribution of this method is that it does not rely on regression trees, which have been widely used to infer regulation programs in module networks [106, 62, 99]. Our experimental results in two real biological datasets indicate that the tree structure has a limitation that may affect the performance of regression tree-based algorithms, but the proposed method can overcome the limitation. In addition, the linear model is capable of detecting under which conditions the transcription factors of a module regulate the genes in the module.

The third method explores the possibility of integrating results from complementary regulation program learning algorithms [101]. To the best of our knowledge, this is the first such an attempt. The integration methods we select are union, intersection, and weighted rank aggregation. They are applied to combine ordered lists of regulator-module interactions in a yeast benchmark dataset from three algorithms: LeMoNe [62], the LIMMA-based method [100], and Inferelator [11]. These three algorithms rely on distinct techniques, and consequently show different biases in detecting regulatory relationships. The experiments show that integrating their results by the union or the weighted rank aggregation produces promising results.

1.3 Road map

This thesis is organized as follows. Chapter 2 and Chapter 3 introduce the backgound of this thesis. Chapter 2 gives brief descriptions of the major tasks in the pipeline of annotating the genome of a species, such as genome sequencing, gene prediction, and identifying regulatory binding sites. Chapter 3 focus on reviewing the work for applying Bayesian networks and module networks to infer transcription regulatory networks from gene expression data.

Our work is presented in Chapters 4-7. Chapter 4 introduces a regression tree-based Gibbs sampling algorithm for learning regulation programs in module networks. Chapter 5 applies linear models to infer regulation programs. Chapter 6 describes an integrative approach to inferring regulation programs. In addition, Chapter 7 applies module networks to infer regulatory relationships in three fungal species. Last, Chapter 8 concludes the thesis by summarizing the main results and point to future work.

Chapter 2

Bioinformatics Background

There are two stages for analyzing a selected species: genome sequencing and genome annotation. In the former stage, the DNA composition of the genome of the species is identified, while in the latter stage the completed sequence is annotated. In this chapter, we will briefly review the tasks involved in these steps. Section 2.1 describes genome sequencing and assembly. Section 2.2 reviews several major tasks in genome annotation. Since this thesis is about using gene expression data to infer transcription regulatory networks, we describe technologies for collecting gene expression data and techniques for analyzing expression data in Sections 2.3-2.6.

2.1 Genome sequencing and assembly

There are two main sequencing genome technologies: whole-genome shotgun sequencing and hierarchical shotgun sequencing [94]. The former is employed to sequence genomes that are smaller in size and contain less repetitive DNAs. To apply this method, genomic DNA is isolated from an organism and mechanically sheared into fragments that are subcloned into libraries. These libraries, normally consisting of 10 to 20 kb DNAs, are sequenced from both ends by operations that are able to generate short sequence data (e.g., about 500 to 800 bp). These short sequence data are connected to become contiguous segments (called contigs) that are assembled to obtain a map of the complete genome.

In contrast, hierarchical shotgun sequencing subclones digested genomic DNA into bacterial artificial chromosome (BAC) libraries, which contain longer DNA fragments (e.g., 100 to 500 kb) and mapped to known chromosomal locations. The advantage of hierarchical shotgun sequencing over whole-genome shotgun sequencing is that it is less likely to make mistakes in sequence assembly. However, hierarchical shotgun sequencing takes more time and cost than whole-genome shotgun sequencing. Hence, the size and complexity of the selected genome determines which sequencing method should be adopted.

2.2 Genome annotation

Genome annotation can be divided into two processes: structure annotation and functional annotation. Genome structure annotation focuses on detecting genomic elements. For example, one of the major tasks in structure annotation is to identify genes and predict their structures. In contrast, genome functional annotation focuses on attaching functions to these genomic elements. Basic tasks in functional annotation are to determine biological and biochemical functions of products of genes, such as associating genes with Gene Ontology annotations and assigning proteins to protein families. Typical examples of advanced tasks in functional annotation include the reconstruction of metabolic networks and transcription regulatory networks. In this section, we will give a brief description of these tasks.

2.2.1 Gene prediction

Gene prediction not only searches for locations of genes in a target genome, but also identifies the structures of these genes, including coding regions (known as exons) and intervening sequences (known as introns). There are two types of methods for gene prediction: extrinsic and intrinsic.

In extrinsic gene finding systems, genomic DNA is compared to sequences of known protein products or expressed sequence tags. Hence, these systems rely on extensive transcript and protein sequence databases. Typical examples of such databases are the Reference Sequence (RefSeq) [98] and Ensembl [37]. Many algorithms, such as BLAST [5] and sim4 [38], have been designed to search for the match between a known sequence and a region of the target genome, indicating that a protein-coding gene has been identified. The limitation of extrinsic gene finding systems is that they might produce poor performance in a genome for which few known protein or expressed sequence tags are available [12].

Intrinsic gene finding systems, also called *ab initio* approaches, are based on the observation that the base composition of gene-coding regions is normally very different from that of non-coding regions in a genome. Consequently, these algorithms search the whole genomic DNA sequences for long open reading frames. A well known example of this type gene finding method is GlimmerHMM [78], which is based on a generalized Hidden Markov model. The major advantage of this algorithm is that users are allowed to re-train it for a target genome by using complete coding sequences collected from the genome.

2.2.2 Gene annotation using sequence similarity

There are several well-known sequence databases, such as the GenBank [8], European Molecular Biology Laboratory (EMBL), Nucleotide Sequence Database[22], and the DNA Databank of Japan [91]. They include characterized sequences from many species. One approach to identifying the function of a new gene is to search these databases for genes with similar sequences whose functions have been identified,

The comparison of sequences is performed by sequence alignment, which helps calculating the alignment score between two sequences, using similarity matrices in which higher scores are given to similar characters (e.g., nucleotides) and lower scores are given to dissimilar characters. Given a query sequence and a library of sequences, sequence alignment first computes the score between the query sequence and each sequence in the library. Then, the query sequence can be predicted to have the function of the sequence in the library that has the maximum alignment score to the query.

There are two types of sequence alignment algorithms: global alignment and local alignment. The former aligns a complete nucleotide or protein sequence to another sequence by spanning the entire length of both sequences. The Needleman-Wunsch algorithm [90] is often applied to perform global alignment. In contrast, a local alignment method aligns the most similar stretches within two sequences. In other words, a local alignment method is used to determine similar regions between sequences, and consequently it can start and end at arbitrary positions in aligned sequences. A well-known method for local sequence alignment is the Smith-Waterman algorithm [112].

2.2.3 Gene ontology annotation

The Gene Ontology (GO) is one of the most important ontologies within the bioinformatics community, and is being developed by the Gene Ontology Consortium [47]. The primary goal of GO is to define a shared, structured and controlled vocabulary to annotate molecular attributes across model organisms [46]. It represents a repository of computable biological knowledge and comprises three ontologies: molecular function (MF), biological process (BP), and cellular component (CC). MF represents information on the role played by a gene product. BP refers to a biological objective to which a gene product contributes. CC represents the cellular localization of a gene product, including cellular structures and complexes [132].

GO terms and their relationships are represented by Directed Acyclic Graphs (DAGs), where each node except for the root has one or more parent nodes and no cyclic relationships between terms are allowed. There are two kinds of relationships between children nodes and parent nodes: "is a" and "part of". The former is used when a child class is a subclass of a parent class, while the latter is used when a child is a component of a parent. Each gene product can be annotated with a set of GO terms.

The GO-based functional annotation consists of two steps. First, given a list of genes, their associated GO-terms are extracted from GO databases. Second, enriched terms in the GO annotation of these genes are identified. These terms may provide an insight into the biological functions that these gene are involved in. Many algorithms have been designed to identify enriched GO terms, such as BiNGO [76] and GlueGO [10].

2.2.4 Protein feature annotation

Proteins are normally associated with some features (e.g., protein families and domains), and identifying these features can provide insights into functions of proteins. The InterPro database [58] consists of a large collection of such features that are found in proteins with known functions. Consequently, they are often used to annotate functions of new protein sequences. InterPro consists of 11 member databases, which are based on different but complementary techniques. When different algorithms identify features that are located in the same region of same proteins, these features are merged into a single feature by a curator. This design ensures the consistency of data in InterPro.

For example, Pfam [36] is a member of the InterPro database, and focuses on conserved protein families. The core of Pfam is a set of seed alignments. Each of these seed alignments consists of a group of sequences that are relatively stable between releases of the database, and can be used to build profile hidden Markov models (HMMS). Given a profile HMM and a set of protein sequences, HMMER [31] can search homologous sequences of this profile HMM from this set of sequences. The identified sequences are predicted to be associated with the same functions as the proteins from which the profile HMM is built.

2.2.5 Reconstruction of metabolic networks

Metabolism is the set of chemical reactions that occur in a living organism. These chemical reactions are organized into two types of pathways: catabolism and anabolism. The former convert molecules, which serve as food, into energy, while the latter use energy to construct components of cells (e.g., proteins and nucleic acids). The set of all pathways in a cell is called the metabolic network. Enzymes, a special type of proteins, play an important role in metabolism, because they act as catalysts to allow chemical reactions in metabolism to proceed quickly and efficiently.

After identifying genes and their functions in an organism, one task is to reconstruct the metabolic network of this organism. The Pathway Tools [64] is widely used for this task. The reconstruction by the Pathway tools consists of two steps. In the first step, this tool predicts the metabolic-pathway complements of this organism from its genome by the comparison to MetaCyc [19], a metabolic pathway reference database. This database records more than 1,400 experimentally determined pathways from all domains of life. In the second step, the Pathway Tools calculates the evidence that each pathway recorded in MetaCyc occurs in this organism. The evidence of a pathway is based on how many enzymes in this pathway exist in this organism. Moreover, the presence of an enzyme is determined by checking whether a gene in this organism is associated with the enzyme. Hence, the quality of the reconstructed metabolic network strongly relies on the accuracy of the gene prediction in this organism.

2.2.6 Reconstruction of transcription regulatory networks

Transcription regulatory networks can be reconstructed by detecting the correlation of expressions between transcription factors and their targets. This thesis focuses on designing this type of methods, and Section 2.6 will review related work.

Binding specificity is an important property of transcription factors. That is, each transcription factor is capable of recognizing and binding to a particular pattern of binding sites, which is called a *motif*. Consequently, another direction to reconstruct transcription regulatory networks is to identify regulatory binding sites of transcription factors. This type of methods can broadly be categorized into two groups: gene expression-based approaches and phylogenetic approaches. The former are based on the observation that co-expressed genes are very likely to be regulated by the same transcription factors, and consequently they share common motifs. Gibbs sampling [125] has been extensively adopted to search for motifs in the upstream regions of co-expressed genes. In Section 2.3, we will describe how to collect gene expression data.

Phylogenetic approaches are based on the idea that functional elements in a genome, such as binding motifs in the upstream regions of genes, are more likely to be conserved than non-functional elements during evolution. Hence, motifs can be detected from the alignment of the upstream regions of orthologous genes from close species. The authors in [136] applied this strategy to find motifs in human genome.

Putative binding sites produced by the aforementioned computational methods can be experimentally validated by chromatin immunoprecipitation [13]. The other way to validate putative binding sites is to check whether they are recorded in databases of known binding sites. TRANSFAC [83] is the most comprehensive curated database for regulatory binding sites.

2.3 Collecting gene expression data

Gene expression data are often used in functional annotation. They are normally organized into a data matrix, where each row represents the expression levels of a particular gene across all experimental conditions, while each column represents the expression level of each gene in a particular condition. In this section, we describe several technologies for collecting expression data.

2.3.1 Microarray technology

Microarrays exploit the preferential binding of complementary single-stranded nucleic-acid sequences. The basic principle is that unknown samples are hybridized to an ordered array of immobilized DNA molecules whose sequences are known [48]. The idea of using a piece of DNA as a probe to determine the presence of the complementary DNA (cDNA) in a solution is evolved from Southern blotting technology. The most attractive advantage of microarray technology is that it is capable of measuring the expression levels of thousands of genes in parallel.

A microarray is a small chip (made of chemically coated glass, nylon membrane or silicon) onto which tens of thousands of DNA probes are attached in fixed grids. Generally, microarrays are categorized into two groups: cDNA microarrays and oligonucleotide arrays (abbreviated oligo chip). A cDNA microarray simultaneously analyzes two samples, a test sample and a reference sample. In contrast, in oligo chips, the test sample and the reference are separated, and they are analyzed on different chips. In other words, the two samples on a cDNA chip can be viewed as comparable to two samples on two oligo chips [122]. Despite differences in the details of their experimental protocols, both types of experiments consist of the following steps: target preparation, hybridization, scanning, and normalization [61].

2.3.2 Tag-based technology

Microarray technology relies on a collection of representative clones of all genes in a species, so this technology is only able to measure the expression levels of prior known genomic features. In contrast, tag-based technologies can provide absolute expression values without the need for any probe design [52].

Serial analysis of gene expression (SAGE) has pioneered the use of short sequence tags in expression profiling [131]. It consists of three steps. First, after mRNA has been isolated from samples, a small fragment of sequences (14-20 base pairs), called a tag, is extracted from a defined position of each mRNA molecule. Second, these fragments are linked together to form a long chain, which is then cloned into a vector. Third, this vector is sequenced, and consequently the tag frequency of each mRNA is counted. The laborious and costly cloning and sequencing steps limit the use of SAGE. Deep sequencing technology, also referred to as Digital Gene Expression tag profiling (DGE), has been widely used in gene expression analysis. Unlike classical SAGE, tags are not cloned in DGE, but sequenced immediately [117].

2.3.3 RNA-seq technology

RNA-seq [134] is a new technology to measure gene expression. Unlike SAGE and DGE that are based on expensive Sanger sequencing technology, RNA-seq relies on flow cell sequencing [56]. Given a population of RNA, RNA-seq converts them into a library of cDNA fragments. These fragments are then sequenced in a high-throughput sequencing manner to obtain reads. Typical examples of sequencing technologies used by RNA-seq include Illumina IG [3], Applied Biosystems SOLiD [2], and Roche 454 Life Science [1].

After cDNA fragments are sequenced, the next task in the pipeline of analyzing RNA-seq data is to align reads to a reference genome. This task is challenging, because these reads are generally short. Many algorithms have been designed to perform this task, such as QPALMA [25] and TopHat [127]. The major difficulty here is how to align reads that span exon boundaries (i.e., splice junctions). Given a set of known splice junctions from the reference genome, QPALMA trains a support vector machine-like algorithm, and then the algorithm is applied to predict the alignments of junction reads. Unlike QPALMA, TopHat does not rely on known splice sites. TopHat first identifies reads that can not be directly mapped to the reference genome. TopHat then aligns these unmapped reads to splice junctions using a seed-and-extend strategy.

After reads are mapped to the reference genome, the next task is to count the numbers of reads in particular genomic regions, because these numbers can be used as a measure of the prevalence of transcripts from known and previous unknown genes. This represents one of RNA-seq's advantages. That is, it is capable of providing digital gene expression levels, in contrast to analog-style signals from microarray technology [134].

There is a positive association between the length of a gene's transcript and the number of aligned reads in its corresponding genomic region. Hence, raw counts are required to be normalized before they can be used to select highly expressed genes [14]. ERANGE [89] proposes to apply the following formula to normalize reads :

$$R = \frac{10^9 C}{NL},$$

where C denotes the number of reads that are aligned to the exons of a gene, N denotes the total number of reads aligned to the reference genome, and L denotes the total length of the gene's exons in base pairs. R is referred to as reads per kilobase per million reads sequenced (RPKM). Another state-of-the-art algorithm for measuring transcript abundances is Cufflinks [128]. This algorithm uses expected fragments per kilobase of transcript per million fragments sequenced, abbreviated as FPKM, to measure transcript abundances. The abundance of a transcript t in FPKM units is equal to :

$$\frac{10^6 \times 10^3 \times \alpha_t}{\sum_{i=1}^{l(t)} F(i)(l(t) - i + 1)},$$

where α_t denotes the probability that a fragment selected at random comes from t; $l(t)$ is the length of t, F represents a normal distribution, and $F(i)$ denotes the probability that a fragment has length i.

2.4 Known transcription factors in *Saccharomyces cerevisiae*

The budding yeast *Saccharomyces cerevisiae* is one of the most extensively studied eukaryotic model organisms. The YEASTRACT database [123, 88] records information on documented regulatory relationships in this species. In addition, it contains binding sites of transcription factors. In release Dec 13, 2010 [4], there are more than 48,200 regulatory relationships between 183 transcription factors and 6,403 genes based on nearly 1,200 research papers. The transcription factors included in this database cover most transcription factors recorded in the Saccharomyces Genome Database [21], which is the most comprehensive repository of the molecular biology and genetics in the yeast. Due to the comprehensiveness of the YEASTRACT database, it is widely used as a reference database to validate results of regulation network learning algorithms in yeast datasets [133, 137]. In this thesis, we also rely on records in YEASTRACT to evaluate the performance of our methods in a yeast benchmark dataset.

Experimental confidence for regulatory relationships in YEASTRACT can be categorized into two types: direct evidence and indirect evidence. The former is assigned to associations, where bindings of transcription factors to the promoter regions of their target genes have been validated by experiments, such as chromatin immunoprecipitation assays. In contrast, if a regulatory relationship between a transcription factor and a gene is supported by an indirect evidence, this indicates that the expression level of the target gene is changed owing to the mutation (or deletion) of the gene encoding this transcription factor. There are 29,051 regulatory associations based on direct evidence and 19,182 on indirect evidence in YEASTRACT (released on Dec 13, 2010).

The regulation of galactose utilization in the budding yeast is one of the best characterized eukaryotic systems of transcriptional regulation, and is well documented in YEASTRACT. The yeast is capable of converting galactose into glucose-6-phosphate for energy. However, the expression of genes for galactose utilization is repressed when glucose is added into the environment, because glucose is the preferred carbon source for the budding yeast [9]. This is referred to as glucose repression. The transcription factor, MIG1, plays a major role in glucose repression [104]. In the presence of glucose, cytoplasmically located MIG1 is translocated to the nucleus to inhibit the expression of glucose-repressed genes, such as GAL4 which is the activator of genes involved in galactose utilization (Figure 3).

The utilization of galactose consists of two steps: i) galactose is transported across the membrane into the cell by GAL2; ii) glycolysis enzymes (GAL1, GAL5, GAL7 and GAL10) covert galactose into glucose-6-phosphate through a series of metabolic reactions. The transcription of the genes involved in these two steps is influenced by the post-transcriptional regulation between GAL4, GAL80, and GAL3 [9]. When galactose is absent, the function of GAL4 for activating the transcription of GAL2, GAL1, GAL5, GAL7 and GAL10 is disable due to the protein-protein interaction between GAL4 and GAL80. However, in the presence of galactose, GAL3 interacts with GAL80 so that GAL4 is released to activate the transcription of genes involved in galactose utilization.

2.5 Benchmark gene expression datasets

The development of algorithms using gene expression data to annotate genomes relies on benchmark expression datasets, where performances of different algorithms can be assessed. Benchmark datasets can be categorized into two types: synthetic datasets and real biological datasets.

2.5.1 Synthetic datasets

Synthetic datasets are generated by simulators, such as SynTReN [130] and GeneNetWeaver [80]. Given the transcription regulatory network of a species, these simulators sample a

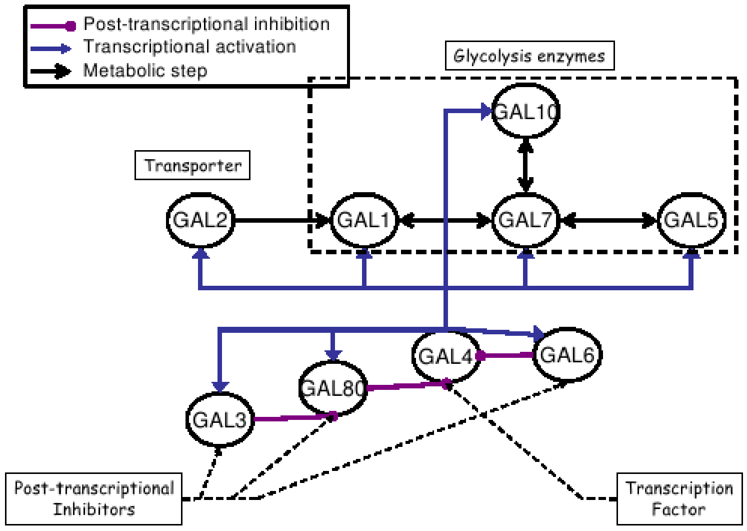

Figure 3: Regulation of galactose utilization in the budding yeast [92]

sub-network from the whole network, and then infer a dynamical model from the sampled sub-network. The inferred dynamical model is capable of generating synthetic expression data, which manifest the expression pattern shown by the genes in the sampled sub-network. Noise is often added into synthetic expression data before they are used to evaluate the performance of algorithms, because all technologies for collecting expression data are subject to random experimental noise.

In order to generate a synthetic gene expression dataset, users first need to specify the size of the sampled network (i.e., the number of genes in the network). Moreover, users need to determine the number of experimental conditions included in the synthetic dataset and the noise intensity level to be added. Synthetic datasets have been widely used to evaluate the performance of algorithms. For example, in [79], synthetic datasets based on the regulatory network of the budding yeast are used to evaluate the performance of algorithms. In Chapter 4, we also test our proposed method on synthetic datasets.

2.5.2 Real biological datasets

In order to become a benchmark, a real biological dataset has to meet the following two requirements. First, the dataset should be relevant to a well studied model organism whose biological processes have been extensively analyzed. This requirement ensures that the biological soundness of the result given by an algorithm in this dataset can be validated. Second, the dataset should consist of expression values collected under a relative large number of conditions, and consequently there are a sufficient number of genes involved in the response given by cells to these conditions. This requirement ensures that the performance of algorithms can be thoroughly evaluated based their results in this dataset. In this subsection, we describe two benchmark datasets: the *Escherichia Coli* dataset [35] and the yeast stress dataset [45].

2.5.2.1 The *Escherichia Coli* dataset

Escherichia Coli (E. coli) is one of the best-studied prokaryotic model organisms. A dataset of gene expression profiles in this species has been assembled [35], and it consists of 445 experiments under various conditions, including growth phases, heat shock, numerous genetic mutations and so on. This dataset has been applied to evaluate the performance of relevance networks, Bayesian networks, linear regression, and module networks for using expression data to infer transcription regulatory networks [35, 86]. We use this dataset to assess our proposed method in Chapter 5. Experimental results of algorithms in this dataset can be validated by records in EcoCyc [65], which is a comprehensive database of biological processes in *E. coli*, and RegulonDB [44], which is a database for experimentally confirmed transcription regulatory interactions in *E. coli*.

Most algorithms achieve better results in this dataset than those obtained in the yeast stress dataset described in Section 2.5.2.2, because *E. coli* has a much simpler transcriptional regulation system than the budding yeast. Consequently, transcription factors and their targets in *E. coli* are highly co-expressed. This facilitates regulatory network learning.

2.5.2.2 The yeast stress dataset

Another widely used benchmark dataset is the yeast stress dataset, which measures the budding yeast's response to a panel of diverse environmental stresses [45]. The conditions covered by the dataset consist of temperature shocks, amino acid starvation, nitrogen source depletion, and so on. Around 900 genes in this dataset show a stereotypical response under many different environmental stresses. This stereotypical response is referred to as the environmental stress response. Genes involved in the environmental stress response are regulated by different transcription factors [45], but they manifest similar expression profiles. Consequently, algorithms using expression data to infer regulatory networks exclude these genes from their analyses [106].

In contrast, there are 2,355 genes in this dataset showing unique response to specific conditions. Many algorithms have been applied to infer transcription factors of these genes, such as [106, 62]. In this thesis, the performance of three proposed methods is also evaluated using this set of genes. *Saccharomyces* Genome Database (SGD) [21] and YEASTRACT [88] are the major reference databases of the budding yeast. Records in these two databases are often used to validate results from different algorithms on the yeast stress dataset.

2.6 Applying gene expression data to annotate genome

In this section, we will describe several major techniques for analyzing expression data, including selecting differentially expressed genes, gene set enrichment analysis, clustering of genes via expression values, and inferring transcription regulatory networks.

2.6.1 Selecting differentially expressed genes

One crucial step in the analysis of expression data is to select differentially expressed genes. Suppose that experimental conditions in a dataset are categorized into two groups: class 1 and class 2. Golub $et\ al.$ [51] proposed to select differentially expressed genes between these two groups by evaluating signal-to-noise statistic values of genes. The signal-to-noise statistic value of a gene g is defined as:

$$P(g) = \frac{[\mu_1(g) - \mu_2(g)]}{[\sigma_1(g) + \sigma_2(g)]},$$

where $[\mu_1(g), \sigma_1(g)]$ and $[\mu_2(g), \sigma_2(g)]$ denote the means and standard deviations of the expression levels of g for the conditions in class 1 and class 2, respectively.

Another similar measurement is t-statistics [72]. The t-statistic value of g is defined as:

$$T(g) = \frac{|\mu_1(g) - \mu_2(g)|}{\sqrt{\frac{\sigma_1(g)^2}{n_1} + \frac{\sigma_2(g)^2}{n_2}}},$$

where n_1 and n_2 denote the numbers of conditions in class 1 and 2, respectively. The other variables are defined as the above equation for $P(g)$. When conditions are clustered into more than two groups, F-statistics can be applied.

In addition, the Wilcoxon rank sum test [129] and empirical Bayesian method [32] have been applied to detect differentially expressed genes from expression data. A common characteristic of the methods described in this subsection is that they are capable of ranking genes into an ordered list. The higher its ranking in this ordered list, the more likely a gene is differentially expressed. Moreover, this ordered list can be used as the input for downstream analyses, such as gene set enrichment analysis.

2.6.2 Gene set enrichment analysis

Given an ordered list of differentially expressed genes, a straightforward method to interpret this list is to verify whether the genes in the top of the list are relevant to the biological difference between condition classes. This is referred to as singular enrichment analysis [57]. The limitation of singular enrichment analysis is that it does not incorporate biological knowledge regarding how genes work together [60].

To overcome this limitation, Subramanian $et\ al.$ [116] designed a statistical method, called gene set enrichment analysis (GSEA), to interpret this ordered list. After genes are grouped into gene sets according to the biological functions they participate in, GSEA is capable of calculating the enrichment score for each gene set. This score can be used to determine whether the members of a gene set are randomly distributed throughout the ordered list or primarily located in the top or bottom. If a gene set is associated with the latter distribution, the genes in this set are predicted to be relevant to the phenotypic distinction between the experimental condition classes.

Suppose that N genes are ranked into an ordered list $L = \langle g_1, ..., g_N \rangle$ according to their degrees of differential expression, $r(g_i) = r_i$, between condition classes. The enrichment score of a gene set S with N_S genes is calculated as

$$ES(S) = \max_{i \in (1,...,N)} \left[P_{\text{hit}}(S,i) - P_{\text{miss}}(S,i) \right],$$

where

$$P_{\text{hit}}(S,i) = \sum_{\substack{g_j \in S \\ j \leq i}} \frac{|r_j|^p}{N_R},$$

$$N_R = \sum_{g_j \in S} |r_j|^p ,$$

and

$$P_{\text{miss}}(S, i) = \sum_{\substack{g_j \notin S \\ j \le i}} \frac{1}{(N - N_S)}.$$

The parameter p determines the weight for correlations of genes with the phenotypic distinction between condition classes, and can be decided by users. Essentially, the enrichment of S denotes the maximum difference between P_{hit} and P_{miss}.

The significance of $ES(S)$ is normally assessed by comparing it with a set of enrichment scores based on permuted condition classes. The percentage of scores more than $ES(S)$ in this set is used as the p-value for S. Given a threshold p-value (e.g., 0.01), several gene sets may be considered to be relevant to the phenotypic distinction between condition classes. Consequently, false discovery rate control or familywise-error rate control [7] is applied to correct multiple comparisons. Compared to false discovery rate, the disadvantage of using familywise-error rate to correct multiple comparisons is that it is overstrict and may lead to no statistically significant gene set [116].

2.6.3 Clustering of genes

Clustering gene expression data is a method widely used in functional annotation. The strategy is to apply clustering algorithms [33] to identify gene clusters, which are groups of genes with similar expression patterns over a range of experimental conditions. If a gene with unknown function is assigned to a cluster dominated by genes involved in some particular biological process, then this gene can be inferred to participate in the same process. Many different clustering methods have been successfully applied to cluster gene expression data. These methods can be broadly categorized into two types: distance-based and model-based.

2.6.3.1 Distance-based methods

Typical examples of distance-based methods include K-means clustering [121] and Self Organizing Map [120]. The common characteristic of distance-based methods is that they rely on similarity metrics to calculate distances between genes. Consequently, genes with close distances are grouped into the same gene cluster. In [33], the similarity between two genes X and Y based on their expression values measured under N conditions is calculated as:

$$S(X,Y) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{X_i - X_{offset}}{\Phi_X}\right)\left(\frac{Y_i - Y_{offset}}{\Phi_Y}\right),$$

where X_i and Y_i denote the expression values of X and Y under the condition i,

$$\Phi_X = \sqrt{\sum_{i=1}^{N}\frac{\left(X_i - X_{offset}\right)^2}{N}},$$

and

$$\Phi_Y = \sqrt{\sum_{i=1}^{N}\frac{\left(Y_i - Y_{offset}\right)^2}{N}}.$$

X_{offset} and Y_{offset} denote reference states for the expression values of X and Y. For example, when X_{offset} and Y_{offset} denote the means of X and Y under these N conditions, $S(X,Y)$ becomes the Pearson correlation coefficient between these two genes. The distance-based methods are simple and computationally efficient. However, they are sensitive to noise, and have difficulty in handling missing data [102].

2.6.3.2 Model-based methods

Instead of using similarity metrics, model-based methods assume that genes in a cluster are drawn from the same probability distribution. Accordingly, the expression values in a dataset are considered to be generated from a mixture of probability distributions. Suppose that $X = \{X_{ij}, i = 1,...,N, j = 1,...,M\}$ denotes expression values of N genes collected under M conditions, and X_{ij} is the expression value of the gene i under the condition j. Furthermore, suppose that the N genes are assigned into K gene clusters. Let $E = (E(i), i = 1,...,N)$ denote the cluster indicator variable, and $E(i) = k$, $1 \leq k \leq K$ denote that the gene i is assigned to the cluster k. In addition, suppose that expression values of genes in the cluster k under the condition j follow a normal distribution with the mean β_{kj} and variance σ_{kj}^2. That is, $X_{ij} \sim N(\beta_{kj}, \sigma_{kj}^2)$ if $E(i) = k$. The likelihood of observing X is proportional to:

$$P\left(X|E,\beta,\sigma^2\right) \propto \prod_{k=1}^{K}\prod_{E(i)=k}\prod_{j=1}^{M}\left(\left(\sigma_{kj}^2\right)^{-1/2} e^{\left(-1/2\sigma_{kj}^2\right)\left(X_{ij}-\beta_{kj}\right)^2}\right).$$

The expectation maximization algorithm [28] and Gibbs sampling [63] can be applied to determine the values of parameters (e.g., E, β, σ^2) that maximize the above likelihood.

The advantage of model-based methods is that they are more resistant to noise than distance-based methods. In addition, gene clusters obtained by model-based methods can be statistically evaluated by the likelihood of observing the data from the learned mixture of probability distributions. Due to the aforementioned advantages, model-based clustering approaches draw more and more attention. Typical examples of model-based methods include the finite mixture model [49] and the Dirichlet process mixture model [102].

2.6.4 Inferring transcription regulatory networks

Gene expression data are widely used to infer transcription regulatory relationships between genes. For example, when genes are grouped into gene clusters based on their expression profiles, if most genes in a learned cluster are known to be regulated by a transcription factor, it is very likely that this transcription factor also regulates the expression of other genes in this cluster. Beside clustering algorithms, Bayesian networks have also achieved promising results in inferring regulatory relationships from gene expression data, and Chapter 3 will review related work in this direction. In this subsection, we will describe how to apply relevance networks and linear regressions to learn regulatory networks.

2.6.4.1 Relevance networks

In a relevance network, each node denotes a gene, and two nodes are connected by an edge only when the mutual information between them is higher than a given threshold [15]. The mutual information between two genes X and Y based on their discretized expression values is defined as:

$$I\left(X;Y\right) = \sum_{i,j} P\left(x_i, y_j\right) \log \frac{p\left(x_i, y_j\right)}{p\left(x_i\right) p\left(y_j\right)},$$

where $P\left(x_i\right)$ denotes the probability that X shows the expression value x_i. The higher mutual information between a transcription factor and a gene, the more likely they have a regulatory relationship. Compared to the Pearson correlation coefficient, the advantage of mutual information is that it does not assume linearity, continuity, or other properties associated with the relationships between expression values of transcription factors and their targets, but it costs more computationally [35].

A limitation of the relevance network method is that its performance deteriorates when some candidate transcription factors are weakly co-expressed with a large number of genes.

To cope with this problem, Faith *et al.* [35] designed the context likelihood of relatedness (CLR) algorithm, which extends the relevance network method by adding a background correction step. The background distribution for the mutual information between gene X and gene Y consists of two sets of mutual information values: the set (MI_X) of mutual information values between X and all other genes, and the set (MI_Y) of mutual information values between Y and all other genes. MI_X and MI_Y can be approximated by normal distributions. Consequently, the corrected mutual information between X and Y can be calculated as:

$$f(Z_X, Z_Y) = \sqrt{Z_X^2 + Z_Y^2},$$

where Z_X and Z_Y denote the z-scores of the mutual information value between X and Y from the normal distributions associated with MI_X and MI_Y, respectively.

Another limitation of the relevance network method is that genes with indirect regulatory relationships may show high mutual information and this increases the number of false positives. For example, if gene X regulates gene Y that regulates gene Z, X and Z probably have enriched mutual information due to the co-expression between them. In [81], the authors applied the data processing inequality to filter high mutual information associated with indirect interactions. The data processing inequality states that if gene X and gene Z interacts only through gene Y, then:

$$I(X;Z) \leq \min\left[I(X;Y), I(Y;Z)\right].$$

Hence, the interaction between X and Z can be eliminated from predictions by the relevance network method if $I(X;Z) < I(X;Y)$ and $I(X;Z) < I(Y;Z)$.

2.6.4.2 Linear regression

In [11, 74], linear regression was applied to infer transcription regulatory networks from gene expression data. Let $Y = (y_1, y_2, ..., y_N)$ denote the expression values of gene y under N conditions. Let $Z = (z_1, z_2, ..., z_p)$ denote a list of p candidate transcription factors for y. Then, the expression value of y under the condition i can be modeled as:

$$y_i = \alpha + \sum_{j=1}^{p} \beta_j z_{ij},$$

where z_{ij} denotes the expression value of the transcription factor z_j under the condition i, and β_j is the regression coefficient of z_j. Transcription factors associated with positive (or negative) coefficients activate (or repress) the expression of y. In addition, the impact of a transcription factor on the regulation of y can be evaluated by the magnitude of its coefficient.

The regression coefficients of candidate transcription factors can be inferred by ordinary least square multivariate regression, but the method tends to assign many transcription factors with non-zero coefficients, and this makes it difficult to interpret the result. To overcome this limitation, L1 shrinkage [126] is often used, which is defined as:

$$\left(\hat{\alpha}, \hat{\beta}\right) = \underset{\alpha,\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} \left(y_i - \alpha - \sum_{j=1}^{p} \beta_j z_{ij} \right)^2 \right\}$$

with the constraint:

$$\sum_{j=1}^{p} |\beta_j| \leq t \sum_{j=1}^{p} |\beta_j{}'| ,$$

where $\beta_j{}'$ denotes the ordinary least squares estimate of the coefficient of z_j, t is called the shrinkage parameter, and its value is in the range of 0 to 1. The optimal value of t can be identified by cross validation [11]. One advantage of linear regression is that it can be used for inferring transcription factors of a set of co-expressed genes (i.e., a regulatory module). It is implemented by regressing the mean of expression values of this set of genes on a list of candidate transcription factors. In Chapter 6, we apply this method on the yeast stress dataset.

Chapter 3

Related work

Many algorithms, such as information-theoretic approaches, ordinary differential equations, and clustering algorithms, have been adopted for learning transcription regulatory networks using gene expression data. In [6, 35], the authors described how to apply them, and compared their strengths and limitations. In this chapter, we will review the work using Bayesian networks.

This chapter is organized as follows. Section 3.1 gives a brief introduction of learning Bayesian networks. Section 3.2 reviews several methods that applied Bayesian networks to infer transcription regulatory networks. Section 3.3 describes how to apply the module network method [106], a special type of Bayesian networks, to learn transcription regulatory networks.

3.1 Introduction to Bayesian networks

3.1.1 Basic structure

Bayesian networks are a combination of probability theory and graph theory. They are very useful to represent probabilistic relationships between multiple interacting entities. A Bayesian network can be represented by a directed acyclic graph (DAG). In the context of transcription regulatory networks, each node denotes a gene, while each edge indicates a regulatory relationship between two genes. Figure 4 shows a Bayesian network representing a portion of transcription regulatory pathways involved in glucose repression in the budding yeast. It describes that the transcription factor, SNF1, regulates its target, MIG1,

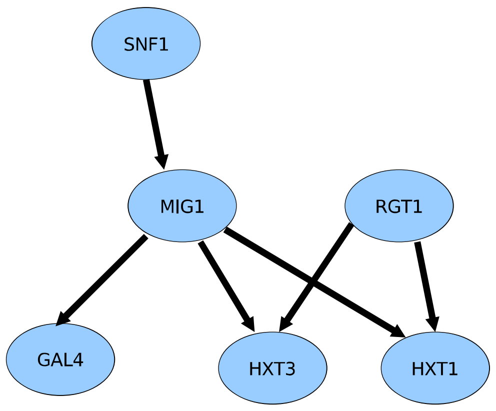

Figure 4: Bayesian network for a portion of transcription regulatory pathways involved in glucose repression in the budding yeast

which is also a transcription factor and regulates the transcription of genes involved in glucose utilization [16]. In addition, the combination of MIG1 and RGT1 regulates the transcription of HXT3 and HXT1 whose functions are to transport glucose into cells [66].

Conditional independence is a critical concept in Bayesian networks. Mathematically, a and b are conditionally independent given c if $p(a, b|c) = p(a|c)p(b|c)$. Essentially, the DAG structure in a Bayesian network encodes conditional independence relations between nodes. That is, a node is conditionally independent of its non-descendants given its parents. This allows efficient inference and learning in Bayesian networks.

Each node in a Bayesian network is associated with a conditional probability distribution that describes its relationship with its parents. For discrete variables, the multinomial distribution is often used, and it can be expressed as a conditional probability table. For continuous variables, the most common choice is the linear Gaussian distribution. That is, each node is associated with a Gaussian distribution whose mean varies linearly with the value of the node's parents and whose standard deviation is fixed.

3.1.2 Parameter learning

The task of parameter learning is to find the parameter θ of the conditional probability distributions associated with the nodes given a network structure S and a set of training data D. It can be computed efficiently under two assumptions: no missing data in the training set (i.e., complete data) and parameter independence [54].

For complete data, the maximum likelihood estimation (MLE) is a typical method for parameter learning in Bayesian networks. This learning method aims to maximize the likelihood of data, i.e. $p(D|\theta)$. The probability of observing a new sample x is estimated by $p(x|\theta_{MLE})$, where θ_{MLE} is calculated as:

$$\theta_{MLE} = \arg\max_{\theta}\ p(D|\theta).$$

This method is not a strict Bayesian approach, because no prior distribution of θ is included. In contrast, the maximum a posteriori (MAP) estimator aims to maximize the posterior distribution, and is defined as:

$$\theta_{MAP} = \arg\max_{\theta}\ p(D|\theta)p(\theta),$$

where $p(\theta)$ denotes the prior distribution of θ. The Dirichlet and Gaussian priors are often used for the multinomial and Gaussian distributions, respectively, because it is straightforward to calculate the corresponding posterior distributions. When the training data contains missing data or hidden variables, the expectation maximization algorithm [28] is often used.

3.1.3 Structure learning

In the previous subsection, we showed how to learn the parameters of a Bayesian network with a known structure S, but in reality, S is often unknown. The common strategy for structure learning is to introduce a statistically motivated scoring function that evaluates each network with respect to the training data, and then to search for the network with the maximal score [42].

The score can be evaluated by the posterior probability of a structure S given the data D:

$$\text{SCORE}(S : D) \ = \ \log P(S|D) \ = \ \log P(D|S) + \log P(S) + C,$$

where C is a constant independent of S; $P(D|S)$ denotes the marginal likelihood averaging the probability of the data over all possible parameter assignments to S, and is defined as:

$$P(D|S) = \int P(D|S, \theta)P(\theta|S)d\theta;$$

and $P(S)$ represents the prior knowledge about the network structure. For example, we may assume that all networks are equally likely if no prior knowledge is available.

The number of possible network structures is exponential to the number of variables, so an exhaustive search is not feasible even for a network with a small number of nodes. Consequently, heuristic methods are applied to decide how to navigate in the search space. The greedy hill-climbing algorithm is one of the most widely used methods. In addition, the simulated annealing and best-first search are also applied. A detailed description of structuring learning in Bayesian networks is included in the tutorial [54] given by Heckerman.

3.2 Applying Bayesian networks to infer transcription regulatory networks

Bayesian networks have yielded promising results in inferring transcription regulatory networks from gene expression data [42, 107]. One advantage of this method is to apply Bayesian probability theory to deal with uncertainty of data. Moreover, the method uses directed acyclic graphs to represent transcription regulatory networks, which facilitates the interpretation of results, i.e., that each gene is regulated by its parents.

Despite the success of Bayesian networks in learning regulatory networks, this method has a major shortcoming. A typical gene expression dataset describes thousands of genes, but at most consists of a few hundreds of instances (experimental conditions). In Bayesian networks, each gene is associated with its own set of parents and a conditional probability distribution, so the number of structural features and distribution parameters to be learned is enormous relative to the amount of available data [87]. Hence, learned models may overfit the data, and consequently do not represent real regulatory relationships. In addition, some subtle patterns may not be detected when training samples are rare. To cope with this problem, several techniques have been proposed, including sparse candidates [43], bootstrapping [40], model averaging [41], and adding prior biological knowledge [59].

3.2.1 Sparse candidates

As described in Section 3.1.3, heuristic search strategies are frequently used to cope with huge search spaces in learning regulatory networks. However, the search process is still very time-consuming when there are thousands of genes. Because a gene only interacts with a very limited number of genes, the sparse candidates algorithm [43] restricts the maximum number of affecting genes for each target gene. This design significantly reduces the number of possible networks.

The sparse candidates algorithm first identifies a relatively small number of candidate parents for each gene, based on simple local statistics, such as correlations between gene expression levels. Then, it restricts the search to networks, where only the candidate parents of a gene can be its parents. In addition, this algorithm is capable of dynamically adjusting the candidate parents for each gene so that the search space is not overly restricted.

3.2.2 Bootstrap-based confidence estimation

When learning Bayesian networks with many variables and a small number of samples, we often find that many networks with largely distinct structures should be considered as a reasonable explanation of the given data. From a Bayesian perspective, this indicates that the posterior probability over models of the given data is not dominated by a single network [42]. Because it is not feasible to list all likely networks in the context of learning regulatory networks, the authors in [40] proposed to extract common edges (features) from them, instead of learning entire network structures.

There are two types of features of interest: Markov relations and order relations. A Markov relation between two genes indicates that one gene is in the Markov blanket of the other. The Markov blanket of a variable in a Bayesian network is the minimal set of variables that shield this variable from the rest of variables. Essentially, this feature represents that these two genes are involved in the same biological process. An order relation between two genes denotes that one gene is an ancestor of the other. This feature indicates that there is a regulatory relationship between these two genes.

The statistical confidence of an extracted feature is evaluated by a bootstrap method. This method first generates a perturbed version of the original dataset by sampling, with replacement, a fixed number of instances. Then, a network is inferred from the perturbed dataset. The above two-step process is repeated multiple times in order to obtain a set of networks. Lastly, the confidence of a feature is measured by the percentage of networks owning this feature in the set of sampled networks. It has been shown that statistical confidences produced by this simple bootstrap method correlate well with the estimates by Bayesian posterior.

3.2.3 Bayesian model averaging

Bayesian model averaging accounts for model uncertainty by averaging over all possible models [75, 55]. The posterior distribution of a quantity of interest, $\triangle$, such as the probability of an edge, conditional on the data, D, is given by:

$$\text{pr}(\triangle|D) = \sum_{k=1}^{K} \text{pr}(\triangle|M_k, D)\text{pr}(M_k|D),$$

where $(M_1, ..., M_K)$ denotes the set of models under consideration. In general, the number of terms in the summation can be impractically large, so the summation is often approximated by using Markov chain Monte Carlo model composition [50].

Let A denote the class of models under consideration. The method constructs a Markov chain $\{M(t), t = 1, 2, ...N\}$ with the state space A and the equilibrium distribution pr$(M_i|D)$. Then, the average:

$$G = \frac{1}{N}\sum_{t=1}^{N} \text{pr}(\triangle|M(t), D)$$

can be used to estimate pr$(\triangle|D)$. In [41], the authors apply Bayesian model averaging to reconstruct regulatory networks from gene expression data, and the experiments show that this method outperforms the bootstrap approach proposed in [40].

3.2.4 Divide-and-conquer approach

In [68], the authors applied a divide-and-conquer approach to infer transcription regulatory networks. This method consists of four steps:

1. Seed genes, which are highly differentially expressed, are identified;

2. Genes closely related with the seed genes based on biological similarities and expression similarities, are grouped into overlapped modules. That is, some genes, called

intermediary genes, are involved in multiple modules. The biological similarity between two genes is measured by the Gene Ontology terms associated with them and their annotations in MIPS [85];

3. A Bayesian network is inferred for each module identified in the previous step; and
4. The networks of individual modules are integrated to form the global network through intermediary genes.

This method increases the ratio of the number of samples to the number of genes, and consequently reduces incorrect dependencies caused by the high dimensionality of data. Experimental results show that it is capable of detecting subtle relationships that traditional whole-set-based approaches often fail to identify.

3.2.5 Adding prior biological knowledge

A possible solution to handle small numbers of samples in gene expression data is to combine them with other biological information. For example, in [59], the authors proposed to use protein-protein interactions to determine the prior distribution of a regulatory network, and then gene expression data were used to infer the detail of this network.

Tamada $et\,al.$ [119] enhanced the above method with an iterative procedure, which alternates between learning Bayesian networks and motif detection. First, they infer a gene regulatory network from gene expression data alone. Then, motifs are detected from the upstream region of genes that are regulated by the same transcription factors. Moreover, the learned motifs are embedded into the prior distribution of networks, and then a new network is inferred from the gene expression data. This process is repeated until the structure of the network does not change considerably. The authors in [108] proposed a similar method using the expectation maximization algorithm.

Evolutionary information has been incorporated into learning regulatory networks from expression data [118]. Given two organisms, the basic idea is to first identify the pairs of orthologous genes between these two organisms by sequence alignment. Then, the network of each organism is inferred from expression data. Lastly, incomplete parts in the network of one organism can be enhanced by checking their counterparts (e.g., orthologous genes) in the other organism's network.

3.3 Applying module networks to infer transcription regulatory networks

3.3.1 Basic structure of module networks

The authors in [105] introduced module networks, which are a special type of Bayesian networks. In a module network, each module represents a set of variables that share (1) a single variable or a set of variables as their parents and (2) local distributions. In the context of transcription regulatory networks, a module (i.e., a regulatory module) is a set of genes that show similar expression values under a given set of experimental conditions. In Figure 5, we compare the representations of a transcription regulatory network with 6 genes in the budding yeast by a Bayesian network and a module network. In the latter, these genes are grouped into 3 modules: M1 consisting of GCN4; M2 consisting of DAL80, GLN3, and MET28; and M3 consisting of DAL2 and DAL3. Since genes in a same module share a local probabilistic model, there are only three models to be learned, which is less than the five required by a Bayesian network.

The local probabilistic model (i.e., the regulation program) of a module consists of one or several genes, which are the regulators of this module. It can be represented by a regression tree. Each internal node of the tree is associated with a set of conditions (i.e., a condition cluster) and a test of the behavior of a particular transcription factor, while each leaf node is associated with a condition cluster and a normal distribution which describes the behaviors of genes in the module under this condition cluster. To assign a condition to a leaf node (i.e., a condition cluster), starting from the root of the regression tree, at each internal node we select the branch by evaluating the behavior of the transcription factor tested in the node under this condition.

Figure 6 shows an example of the regulation program of the module M3 in Figure 5. This regulation program consists of two tests (internal nodes): whether DAL80 is highly expressed and whether GLN3 is highly expressed. Consequently, expression values of the genes in this module are partitioned into three condition clusters: cluster1 where the genes in M3 are down expressed, cluster2 where the genes are not expressed, and cluster3 where the genes are highly expressed.

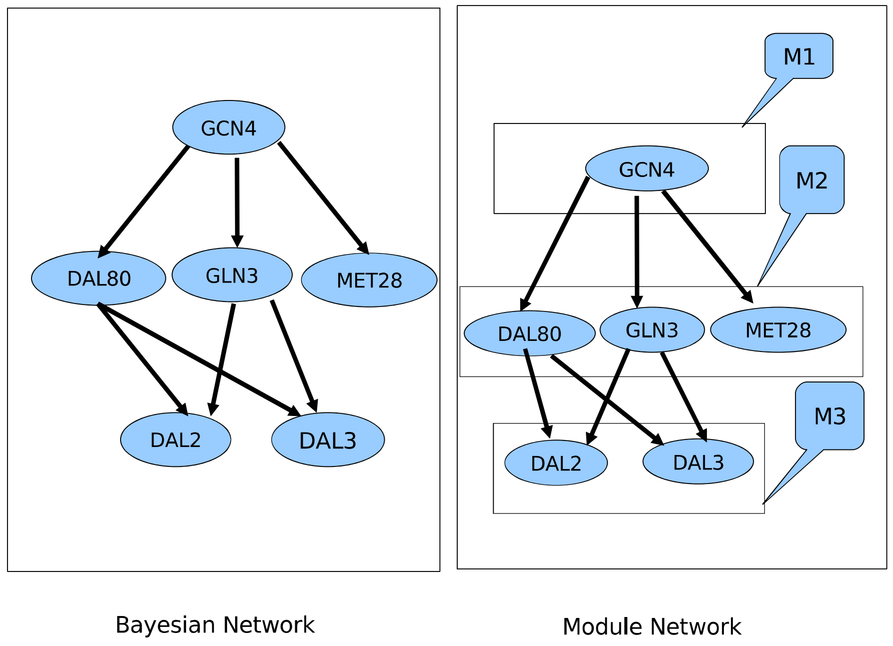

Figure 5: A transcription regulatory network represented by a Bayesian network (left) and a module network (right).

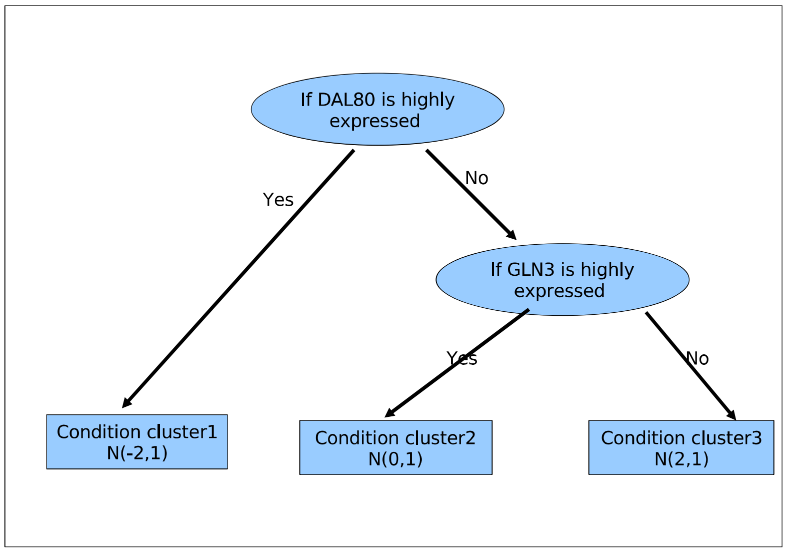
If DAL80 is highly expressed
Yes
No
If GLN3 is highly expressed
Yes
No
Condition cluster1 N(-2,1)
Condition cluster2 N(0,1)
Condition cluster3 N(2,1)

Figure 6: An example of the regulation program of the module M3. $N(\mu, \sigma)$ represents a normal distribution with mean μ and variance σ.

3.3.2 Using the expectation maximization algorithm to learn module networks

To learn module network models, Segal $et\ al.$ [106] applied the Expectation Maximization (EM) algorithm [28], using the Bayesian score [54] to evaluate a model's fit to the data. The learning algorithm is an iterative process, and each iteration consists of two steps: a M-step and an E-step. In each M-step, the algorithm determines the best regulation program (regression tree) for each given gene module, while in each E-step, genes are re-assigned to the module whose regulation program best predicts its behavior. The algorithm stops when the module assignment of genes is not changed between two iterations.

The procedure of a M-step is described as follows:

- Given a list of candidate regulators, start from a regression tree with a single leaf node consisting of all experimental conditions.
- Perform a series of operations that split a leaf node into two leaf nodes using a regulator and a splitting value as the test. The selection of leaf nodes, regulators and splitting values is done to maximize the increase of the Bayesian score.
- In each splitting operation, the selected leaf node becomes an internal node as the parent of the two new leaf nodes, and is associated with the selected regulator and splitting value.
- This process stops when no splitting operation can increase the Bayesian score.

In an E-step, the algorithm assigns each gene to the module whose regulation program best predicts its behavior. Given the regulation program of a module, the likelihood of observing an expression value under a particular condition is determined by the normal distribution associated with the leaf node (i.e., condition cluster), which the condition is assigned to. Consequently, the overall probability that the expression values of a given gene are generated from the regulation program of this module can be calculated by multiplying likelihoods of observing this gene's expression values under individual experimental conditions. Hence, each gene is assigned to the module with the highest probability to generate its expression values.

The module network learning method has yielded promising results in several complex eukaryotic systems, such as the budding yeast [106] and mouse [71]. Figure 7 shows

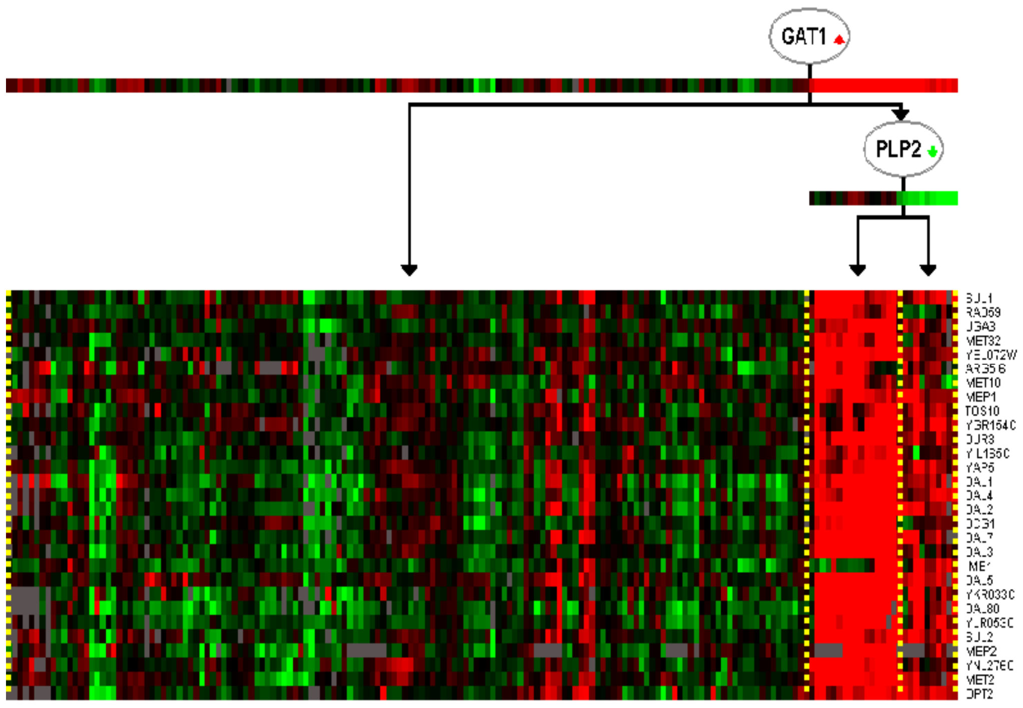

Figure 7: The module of nitrogen catabolite repression in the yeast stress dataset and its regulation program [106]

the module of nitrogen catabolite repression in the yeast stress dataset and its regulation program [106]. The expression values of genes in this module are grouped into three condition clusters, where they are not expressed, highly expressed and moderately expressed, respectively. The regulation program of this module consists of two internal nodes testing the behaviors of transcription factors GAT1 and PLP2, respectively. In addition, several putative regulatory relationships predicted by the algorithm in the yeast stress dataset have been verified by microarray experiments [106].

3.3.3 Applying a two-step-based method to learn module networks

One limitation of the EM-based module network learning algorithm is that it only selects a single module network model that may represent a local maximum in the posterior distribution of models given the data. One possible solution for the issue is to apply sampling-based methods to sample models from the posterior distribution. However, sampling-based methods may show extremely slow convergence rates when learning module networks, because they have to shift between grouping genes into modules (E-steps) and inferring regulation programs of modules (M-steps), which leads to a huge search space of models.

Michoel $et\ al.$ [62] introduced a novel two-step method for learning module networks, which separates clustering genes into modules and learning the regulation program for each module. This design makes it feasible to apply sampling-based algorithms to learn module networks, because the search space of models significantly decreases. Experimental results show that this method outperforms the EM-based learning algorithm. The following subsections will give a briefly description of each step.

3.3.3.1 Clustering genes into modules by Gibbs sampler

In the first stage, the Gibbs sampling algorithm [63], which is based on the weighted Chinese restaurant process [102], is used to cluster genes into modules. One innovative design of the clustering algorithm is to apply two-way clustering of genes and conditions. That is, instead of considering that expression values under different conditions in a module are independent, it clusters conditions into condition clusters, and then assumes that expression values for conditions in a condition cluster are drawn from a same normal distribution. This design significantly reduces the number of parameters to be determined, and consequently shows good convergence even for large datasets.

3.3.3.2 Inferring regulation programs by logistic regression

In the second stage, given a list of candidate transcription factors and gene modules learned in the first stage, the algorithm calculates the confidences (i.e., regulatory scores) for assigning these transcription factors to these modules [62]. The procedure is described as follows:

- Given a gene module, a condition clustering is generated by the Gibbs sampling algorithm [63] based on the expression values of genes in the module. The condition clustering represents a partition of all conditions in the data and consists of several condition clusters, each of which includes a set of conditions.

- Then, the learned condition clustering is represented by a regression tree. Unlike the regression tree described in Section 3.3.1, each internal node in the tree is only associated with a set of conditions, but no transcription factor.

- For a given transcription factor in the list of candidates, at each internal node in the tree, a logistic regression classifier is built to predict the assignments of conditions included in the node to its left or right child node, using the transcription factor as the feature. The classification accuracy of the classifier is used to determine the confidence of assigning the transcription factor to the node.

- Accordingly, the overall confidence (i.e., the regulatory score) for assigning the transcription factor to the module is calculated by summing individual confidences for the transcription factor in all internal nodes of the regression tree.

- All candidate transcription factors are sorted into an ordered list according to their regulatory scores. The higher its ranking, the more likely a candidate transcription factor regulates the module.

Chapter 4

A regression tree-based Gibbs sampler to learn the regulation programs in a transcription regulatory module network

4.1 Introduction

The purpose of this chapter is to demonstrate feasibility of applying a regression tree-based Gibbs sampling algorithm to learn regulation programs in module networks. Given a module, we use a Gibbs sampler to sample regulation programs, i.e., regression trees, from the posterior distribution of regulation programs given the data. A set of tree operations is defined for generating new regression trees from a given tree. We show that the set of tree operations is sufficient to generate a Gibbs sampler with well mixing rate even for large datasets. Based on the frequency with which a regulator appears in the sampled regulation programs and the significance that the regulator shows in the sampled regulation programs, we provide the confidence estimate for the regulatory relationship between the regulator and the module.

The remainder of this chapter is organized as follows. In Section 4.2, we describe how to apply Gibbs sampling to sample regression trees in module networks. In Section 4.3, we present the experimental results in synthetic and real biological data. Section 4.4 concludes this chapter by summarizing the main results.

4.2 Applying Gibbs sampling to learn regulation programs in module networks

4.2.1 Regulation program of a module and its Bayesian score

The regulation program of a given module can be represented by a regression tree, which consists of two types of nodes: internal nodes and leaf nodes [105]. Each internal node, which has a regulator and a splitting value, corresponds to a test of whether the expression value of the regulator for an experimental condition is greater than the splitting value, and has two child nodes: the right child node is chosen when the answer to the test is true; the left child is chosen otherwise. Each leaf node represents a set of experimental conditions where the behaviors (upregulation, no change, or downregulation) of regulators match the context specified by the tests on the path to the leaf node. In addition, each leaf node is associated with a normal distribution to describe the expression level of the module's genes in the experimental conditions associated with the leaf node.

The task of learning the regulation program for a module is to find the program that best explains the change of gene expression level in the module. A regulation program's fit to a module can be evaluated by the Bayesian score [54] of its regression tree, R, which is defined as:

$$S(R) = \sum_{l \in L} S_l, \tag{1}$$

where L represents the set of leaf nodes in R; S_l is the Bayesian score of leaf node l, which is calculated as:

$$S_l = \log \int \int p\left(\mu, \tau\right) \prod_{m \in \varepsilon_l} \prod_{i \in A} p\left(x_{i,m} | \mu, \tau\right) d\mu d\tau, \tag{2}$$

where ε_l and A denote the set of experimental conditions in leaf node l and the set of genes assigned to the module, respectively; $x_{i,m}$ represents the expression level of gene i in experimental condition m; $p\left(x|\mu, \tau\right)$ is a normal distribution with mean μ and precision τ; $p(\mu, \tau) = p(\mu|\tau)p(\tau)$ is a normal-gamma prior distribution over μ and τ with

$$p(\mu|\tau) = \left(\frac{\lambda_0 \tau}{2\pi}\right)^{1/2} e^{-\frac{\lambda_0 \tau}{2}(\mu - \mu_0)^2},$$

$$p(\tau) = \frac{{\beta_0}^{\alpha_0}}{\Gamma(\alpha_0)} \tau^{\alpha_0 - 1} e^{-\beta_0 \tau},$$

$\alpha_0, \beta_0, \lambda_0 > 0$ and $-\infty < \mu_0 < \infty$. In this work, we use the values $\alpha_0, \beta_0, \lambda_0 = 0.1$ and $\mu_0 = 0.0$.

The double integral in Equation (2) can be solved explicitly by

$$T_l^{(n)} = \sum_{i \in A, m \in \epsilon_l} x_{i,m}^n \ (n = 0, 1, 2.),$$

and the result is

$$\begin{aligned} S_l = & -\frac{1}{2} T_l^{(0)} \log(2\pi) + \frac{1}{2} \log \left(\frac{\lambda_0}{\lambda_0 + T_l^{(0)}} \right) \\ & - \log \Gamma(\alpha_0) + \log \Gamma \left(\alpha_0 + \frac{1}{2} T_l^{(0)} \right) \\ & + \alpha_0 \log \beta_0 - \left(\alpha_0 + \frac{1}{2} T_l^{(0)} \right) \log \beta_1 \end{aligned} \tag{3}$$

with

$$\beta_1 = \beta_0 + \frac{1}{2} \left[T_l^{(2)} - \frac{(T_l^{(1)})^2}{T_l^{(0)}} \right] + \frac{\lambda_0 \left(T_l^{(1)} - \mu_0 T_l^{(0)} \right)^2}{2 \left(\lambda_0 + T_l^{(0)} \right) T_l^{(0)}}.$$

Hence, it is straightforward to compute the Bayesian score of the regulation program for a module.

4.2.2 Sampling regression trees by a Gibbs sampler

In [106], the authors used a deterministic search algorithm to learn the regression tree (regulation program) of a module. The major shortcoming of the deterministic algorithm is that its result may only represent one of several possible models.

In order to account for the model uncertainty in the deterministic search algorithm, we may use Bayesian model averaging [75], which is able to calculate the strength of a regulator over all possible regression trees. However, in general it is not feasible to enumerate all regression trees due to the huge number of possible trees. Hence, we use a Gibbs sampling algorithm to sample regression trees from the posterior distribution of regression trees given the data, where the log-likelihood of a regression tree R under the posterior distribution is given by the Bayesian score (Equation (1)).

Following the standard Gibbs sampling framework [73], given the current regression tree R_t, we sample the next regression tree R_{t+1} from the neighborhood of R_t ($\mathrm{nbd}(R_t)$), which consists of R_t and the regression trees generated by modifying R_t with one of following operations:

- splitting one leaf node into two leaf nodes by a regulator and a splitting value, and converting the split leaf node into an internal node associated with the selected regulator and splitting value;
- trimming two leaf nodes connecting to a same internal node and converting the internal node into a leaf node, i.e., the reverse of the splitting operation; and
- replacing the regulator and splitting value in one internal node with another regulator and splitting value.

The probability or transition rate Q_R that a regression tree $R \in \mathrm{nbd}(R_t)$ is selected as R_{t+1} is proportional to the exponential of the Bayesian score difference, i.e.,

$$Q_R = \frac{e^{S(R)-S(R_t)}}{\sum_{R' \in \mathrm{nbd}(R_t)} e^{S(R')-S(R_t)}}.$$

Starting from a randomly generated regression tree, the Gibbs sampling procedure simulates a Markov chain with the posterior distribution as its equilibrium distribution. In other words, after a burn-in period, the probability to sample a particular regression tree R, is proportional to $e^{S(R)}$.

Initial random regression trees for the Gibbs sampling procedure are generated by the following method. For a dataset consisting of N experimental conditions, we generate a random number K, $1 \le K \le \sqrt{N}$. Then, starting from a tree with only one leaf node consisting of all experimental conditions, we randomly split a leaf node into two leaf nodes using a randomly selected regulator and splitting value. This process stops when the regression tree has more than K leaf nodes and this tree is used as the starting point for the Gibbs sampling procedure.

In large datasets, the splitting operation is very time-consuming, because we need to enumerate all possible combinations of a regulator and a splitting value in each leaf node. In this case, we can restrict the maximum number of leaf nodes in regression trees to decrease the sample space. In addition, the replacing operation may also cause the sampling process to be very slow when there are many regulators or experimental conditions. In this situation, we may discretize the expression values of regulators to decrease the number of possible splitting values of each regulator.

4.2.3 The regulatory score of a regulator

At first view, one approach to identify regulators of a module is to select the regulators that frequently appear in the regression trees sampled by the Gibbs sampling procedure. However, the approach does not consider the significance of regulators in regression trees, e.g., how many conditions they regulate in regression trees.

Hence, we introduce a regulatory score $f(y, R)$, for a regulator y in a regression tree R, representing the significance that y shows in R. This score is defined as:

$$f(y, R) = e^{\frac{\sum_{l \in L_y} S_l}{|A| \cdot C_y}} \cdot \frac{C_y}{C}, \tag{4}$$

where S_l is defined as Equation (2); L_y denotes the set of leaf nodes which can be reached from the internal node where y is assigned to in R; $|A|$ denotes the number of genes assigned to the module; C and C_y denote the total number of experimental conditions in the module and the number of experimental conditions assigned to the leaf nodes in L_y, respectively. Essentially, Equation (4) represents the product of the geometric average of the prediction probability associated with the leaf nodes under y and the weight factor, $\frac{C_y}{C}$, which suggests that regulators regulating more experimental conditions are more significant in a regression tree.

4.2.4 The expected value of the regulatory score of a regulator

Given a sequence of regression trees $(R_t, t = 1, 2, ..., N)$ generated by the Gibbs sampling algorithm, the expected value of the regulatory score of a regulator y with respect to the posterior distribution can be approximated as:

$$E(S(y)) \approx \frac{1}{N} \sum_{t=1}^{N} f(y, R_t). \tag{5}$$

Since $E(S(y))$ takes into account the regulatory score of y shown in each regression tree and the posterior probability of each regression tree, it can be used to evaluate the likelihood that y regulates the module. Hence, given a set of regulators, we can rank the regulators by $E(S(y))$ and then select the top-ranked regulators as the most significant regulators.

In this work, we use the method proposed in [63] to test convergence of the Gibbs sampling algorithm. That is, given a set of regulators $\{y_i, i = 1, ..., m\}$, we run two independent Gibbs samplers. Let a_i and b_i denote $E(S(y_i))$ produced by the first Gibbs sampler and second Gibbs sampler, respectively. Then, the correlation measure between the two samplers is defined as:

$$\rho = \frac{|\sum_{i=1}^{m} a_i b_i|}{\sqrt{(\sum_{i=1}^{m} a_i^2)(\sum_{i=1}^{m} b_i^2)}}. \tag{6}$$

If the correlation, ρ, between two Gibbs samplers is 1, then they reach full convergence.

4.3 Experimental results and discussion

In this section, we present the results produced by the regression tree-based Gibbs sampler for synthetic data and real biological data.

4.3.1 Synthetic Data

We require that the number of genes in a simulated network should be close to that in a real biological module and the regulatory relationships in a simulated network should not be obvious. Hence, we used SynTReN [130] to generate simulated datasets for gene networks with 30 genes of which 10-15 act as regulators. The topology of the networks was sub-sampled from the transcriptional network of *Saccharomyces cerevisiae*. All parameters of SynTReN were set to default values. We generated 12 simulated datasets with the above configurations and each dataset included 60 microarray samples for 20 experimental conditions.

In each synthetic dataset, we ran 10 independent Gibbs samplers using the list of true regulators as potential regulators. Starting from a randomly generated regression tree, each Gibbs sampler used 50 iterations for burn in steps, had a sampling step of 10 iterations, and consisted of 100 iterations in total. This configuration is applied in all Gibbs samplers in this work. Then, we calculated the expected value of the regulatory score of each regulator (Equation 5) based on the regression trees sampled by the 10 Gibbs samplers. Lastly, we ranked the regulators according to their expected values. To investigate the convergence property of this sampling procedure, we calculated the correlation measure between the expected values from two sets of 10 Gibbs samplers (Equation 6), which is 0.99, a value indicating that the Gibbs sampling procedure reached convergence.

In this subsection, we compare the ranks produced by the regression tree-based Gibbs sampling with those produced by the deterministic algorithm [106] and LeMoNe [62]. In a good rank of regulators, the more genes a regulator regulates, the higher the rank of this regulator. We use the F-measure [87] to evaluate the ranks produced by different algorithms. Given a rank of regulators in a synthetic gene network, starting from an empty predicted regulator set, regulators are sequentially added to the predicted regulator set according to their order in the rank. As one regulator is included into the predicted regulator set, the regulator is predicted to regulate all genes in the network. Then, based on the true gene network and predicted regulatory relationships, we calculate the corresponding true positive (tp), false positive (fp), false negative (fn) and F-measure for the current predicted regulator set. The F-measure is defined as:

$$F = \frac{2P \times R}{P + R},$$

where

$$P = \frac{\text{tp}}{\text{tp} + \text{fp}} \quad \text{and} \quad R = \frac{\text{tp}}{\text{tp} + \text{fn}}.$$

Note that regulatory relationships via intermediate regulators are also considered as true positives.

4.3.1.1 Regression tree-based Gibbs sampling versus the deterministic algorithm

The deterministic algorithm (see Section 3.3.2) can rank regulators based on the order that they are selected to split leaf nodes. Figure 8 shows the comparison of the F-measure between the regression tree-based Gibbs sampling and the deterministic algorithm. Generally, the regression tree-based Gibbs sampling gives better F-measure. Furthermore, we compared the ranks produced by the two algorithms in each dataset (Table 1).

Figure 9 shows the regulatory network in dataset 4 where the tree-based sampling algorithm outperformed the deterministic algorithm. In the dataset, the tree-based sampling algorithm ranked MBP1_SWI6, SWI4_SWI6, and SPT16 as the top 3 regulators and they regulate 10, 3 and 12 genes, respectively. The deterministic algorithm selected SWI4_SWI6, ACE2 and TUP1 and they each only regulate 3 genes. Since MBP1_SWI6 and SPT16 are the most important regulators, the tree-based sampling algorithm detects the true network structure in the dataset.

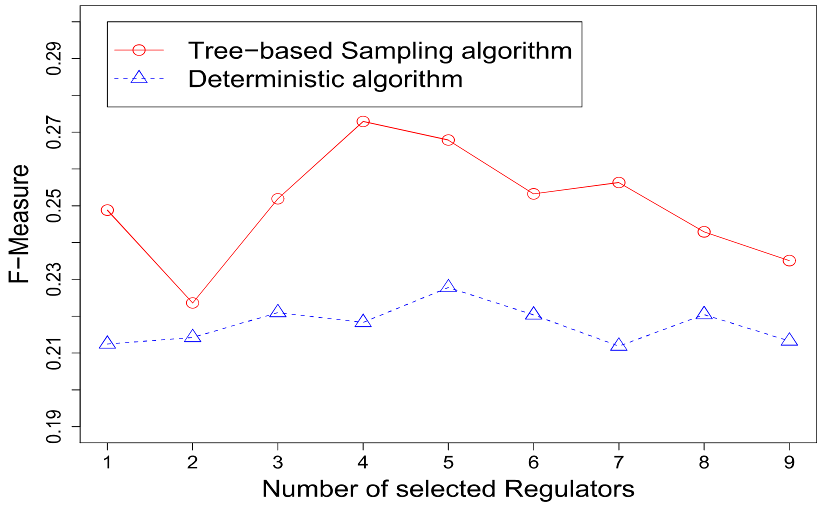

Figure 8: Plot of the average of the F-measure produced by the regression tree-based Gibbs sampling and the deterministic algorithm in 12 synthetic datasets. For each algorithm, the figure shows the average of F-measures obtained by the algorithm in the 12 datasets, when the top i ($i = 1, 2, ..., 9$) regulators in the ranks given by the algorithm are selected.

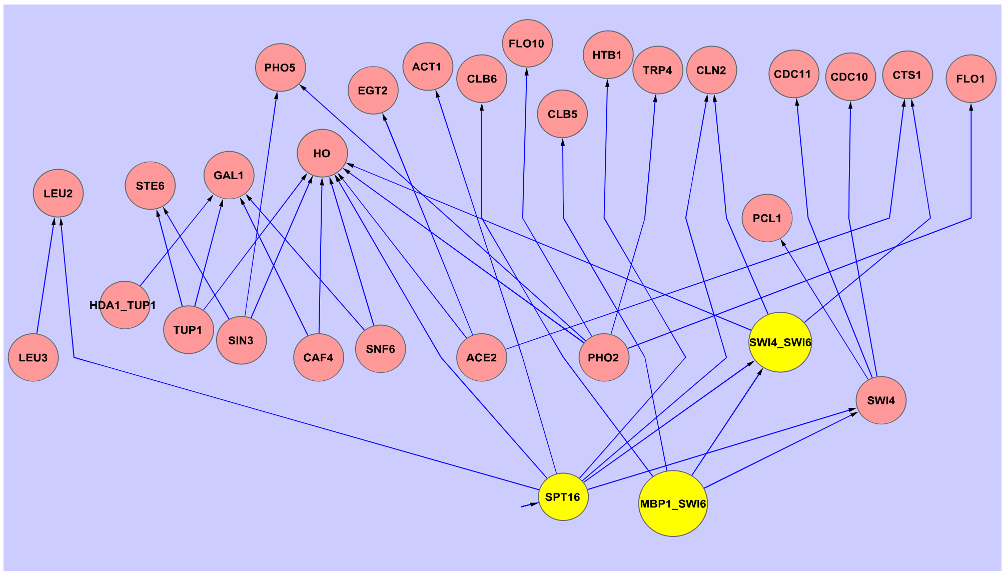

Figure 9: Regulatory network in dataset 4. Each node denotes a gene, while the tail and head of each directed edge denote a transcription factor and a gene regulated by the transcription factor. Three genes (MBP1_SWI6, SWI4_SWI6, and SPT16) are in yellow color because they are going to be further analyzed in the text.

Algorithms	Regression tree-based			Deterministic			LeMoNe		
#Regulators Selected	1	2	3	1	2	3	1	2	3
Dataset 1	0.33	0.27	0.22	0.33	0.27	0.22	0.03	0.04	0.21
Dataset 2	0.38	0.31	0.28	0.38	0.38	0.33	0.38	0.29	0.24
Dataset 3	0.41	0.33	0.33	0.41	0.33	0.33	0.41	0.37	0.32
Dataset 4	0.26	0.24	0.36	0.08	0.11	0.13	0.08	0.11	0.12
Dataset 5	0.17	0.14	0.15	0.17	0.18	0.15	0.17	0.14	0.15
Dataset 6	0.14	0.17	0.27	0.14	0.28	0.26	0.14	0.28	0.33
Dataset 7	0.15	0.12	0.16	0.15	0.12	0.16	0.15	0.12	0.13
Dataset 8	0.17	0.14	0.22	0.17	0.14	0.14	0.17	0.27	0.22
Dataset 9	0.25	0.21	0.35	0.02	0.09	0.26	0.02	0.26	0.35
Dataset 10	0.26	0.32	0.28	0.26	0.26	0.22	0.26	0.32	N/A
Dataset 11	0.41	0.35	0.31	0.41	0.35	0.35	0.41	0.35	0.52
Dataset 12	0.05	0.08	0.08	0.03	0.06	0.09	0.03	0.19	0.18

Table 1: F-measure of the regression tree-based sampling algorithm, deterministic algorithm and LeMoNe in synthetic datasets. In each dataset, for each algorithm, the table shows the corresponding F-measures, when the top one regulator, top two regulators, and top three regulators in the rank of the algorithm are selected, respectively.

4.3.1.2 Regression tree-based Gibbs sampling versus LeMoNe

LeMoNe [62] is an ensemble method, which samples the clustering of experimental conditions and learns fuzzy decision trees as regulation programs. Given a module and a list of regulators, LeMoNe is also able to rank the regulators according to the probabilities that they regulate the module.

In most datasets the regression tree-based Gibbs sampling algorithm and LeMoNe achieved comparable results (Table 1). In dataset 4, the tree-based sampling algorithm outperformed LeMoNe and LeMone's result is similar to that of the deterministic algorithm. However, LeMoNe gave the better result with dataset 12. The regulatory network of this dataset is shown in Figure 10. The tree-based sampling algorithm ranked ALPHA1 and REB1 as the top two regulators, but they both only regulate two genes. LeMoNe assigned A1_ALPHA2 and GAL11 as the top two regulators and they regulate one and nine genes, respectively.

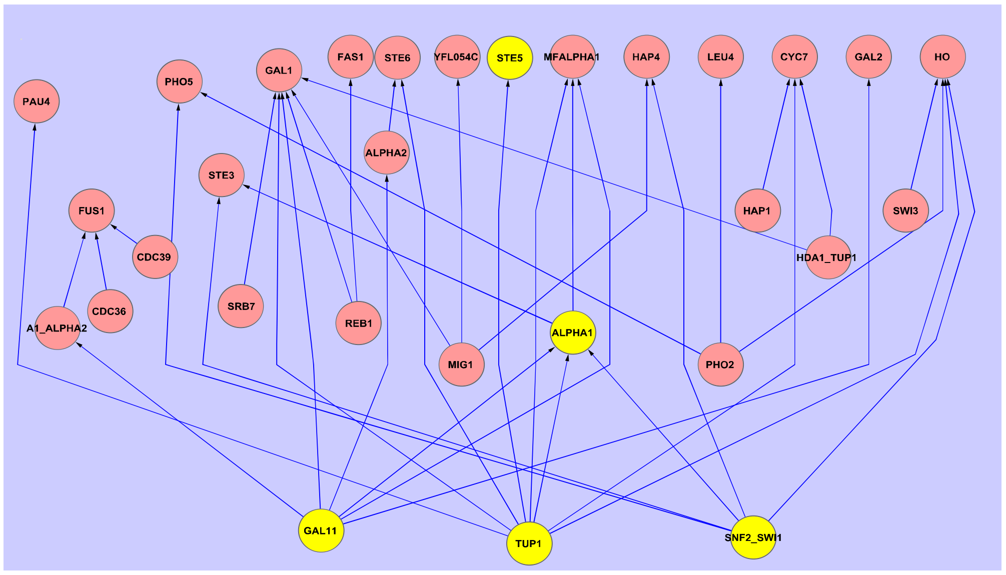

Figure 10: Regulatory network in dataset 12. Each node denotes a gene, while the tail and head of each directed edge denote a transcription factor and a gene regulated by the transcription factor. Five genes (STE5, GAL11, TUP1, SNF2_SWI1, and ALPHA1) are in yellow color because they are going to be further analyzed in the text.

4.3.2 Biological data

In this subsection, we tested the regression tree-based Gibbs sampling algorithm on two real biological modules, module 7 and 11 learned in [63], from the yeast stress dataset [45] consisting of 2355 genes and 173 experimental conditions. The dataset contains a large number of regulators (466) and experimental conditions (173), so the Gibbs sampling procedure may have a slow convergence rate. Hence, we applied the method proposed in [63] to identify how many independent Gibbs sampler runs are required for reaching convergence.

We ran 150 independent Gibbs samplers with module 7. Note that we used the discretized expression values of regulators in the replacing operation when constructing the neighborhood of a regression tree and set the maximum number of leaf nodes in the tree to the square root of the number of the experimental conditions in the dataset, i.e., 14. Then, we calculated the expected value of the regulatory score of each regulator based on the regression trees sampled by k ($k = 1, ..., 50$) samplers. Figure 11 shows the correlation measure (Equation 6) between the expected values of two non-overlapping sets

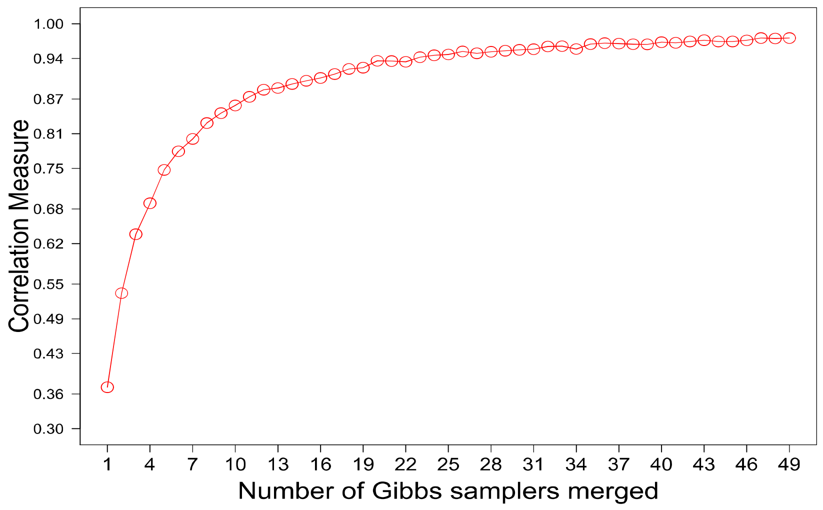

Figure 11: Correlation measure between two sets of k ($k = 1, ..., 50$) Gibbs samplers in module 7. For a given k, we calculated the correlation measure (Equation 6) between the expected values of the regulatory scores of all candidate regulators based on two non-overlapping sets of k Gibbs samplers.

of k ($k = 1, ..., 50$) Gibbs samplers. We observed that all samplers achieved comparable Bayesian scores, but the correlation measure between two individual Gibbs samplers is only around 0.37. This indicates that there are multiple local maxima in the posterior distribution and each sampler can only visit a local maximum. However, the correlation measure between two sets of 20 Gibbs samplers is around 0.94, a value indicating that the Gibbs sampling procedure reached convergence. We observed a similar result of the convergence test with module 11. This implies that the regression tree-based Gibbs sampling has a well mixing rate even for this large dataset.

Based on the above convergence tests, we ran 20 independent Gibbs samplers with module 7 and module 11, respectively. Regulatory relationships recorded in the YEASTRACT [88] database were used to validate the results of the algorithm. Below are the experimental results.

Module 11 consists of 47 genes. Most genes in the module participate in nitrogen catabolite repression or amino acid metabolism. The regression tree-based Gibbs sampling algorithm ranks UGA3, MET28 and GAT1 as the top three regulators, and they are all supported by YEASTRACT. MET28 is a member of the basic leucine zipper DNA binding factor family and encodes a transcription factor that participates in the regulation of sulfur amino acid metabolism. In the module, 8 of 47 genes are known to be regulated by MET28 in YEASTRACT. In addition, the MET28 binding motif, 5'-TCACGTG-3', is detected in the upstream region of 20 genes. GAT1 encodes a transcriptional activator that is involved in the regulation of genes participating in nitrogen catabolite repression. In the module, seven genes are regulated by GAT1 according to the YEASTRACT database and the GAT1 binding motif is found in upstream of 19 genes. UGA3 regulates the transcription of genes, which are required for the utilization of gamma-aminobutyrate as a nitrogen source. In the module, one gene is known to be regulated by UGA3 in YEASTRACT and the UGA3 binding motif, 5'-SGCGGNWTTT-3', is detected in the upstream region of four genes.

Module 7 consists of 30 genes. Most genes in the module participate in respiratory processes. HAP4, GSM1 and USV1 are the top three regulators predicted by the regression tree-based Gibbs sampling algorithm. HAP4 is a well known regulator of respiratory genes and 28 genes in the module are regulated by HAP4 according to the YEASTRACT database. YEASTRACT does not show that any gene in the module is regulated by GSM1 or USV1, but the functions of the two regulators recorded in SGD database [20] are relevant to the respiratory process in the yeast. GSM1 and USV1 are suspected to regulate genes involved in energy metabolism and growth on non-fermentable carbon sources, respectively.

4.3.3 Discussion

The regression tree-based Gibbs sampling, deterministic algorithm and LeMoNe all show bad performance when detecting the regulatory network shown in Figure 10 (dataset 12) and have a common problem that they select an intermediate regulator, which is co-regulated with the module instead of regulating it, as the top regulator. In this subsection we analyze the result produced by the tree-based sampling algorithm to identify the reason.

ALPHA1 is selected as the top regulator by the regression tree-based Gibbs sampling algorithm, but it is actually an intermediate regulator regulated by GAL11, SNF2_SWI1 and TUP1, the most important regulators in the gene network. Due to the regulatory relationships, ALPHA1 co-expresses with the three regulators (Figures 12 to 14). Furthermore, we observed that ALPHA1 also co-expresses with genes regulated by the three regulators to some degree, because correlations between gene expression levels show transitivity.

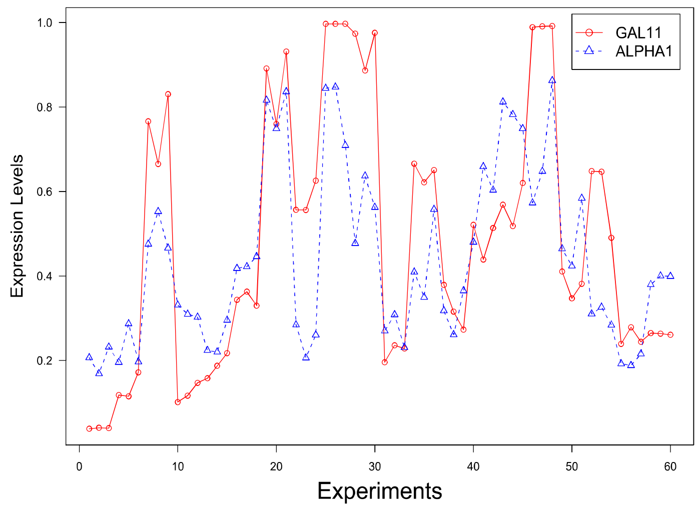

Figure 12: Correlation between the expression profiles of GAL11 and ALPHA1 in dataset 12.

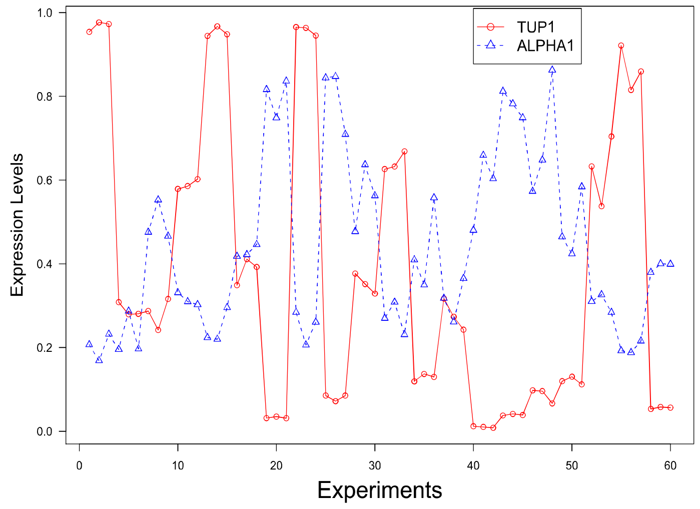

Figure 13: Correlation between the expression profiles of TUP1 and ALPHA1 in dataset 12.

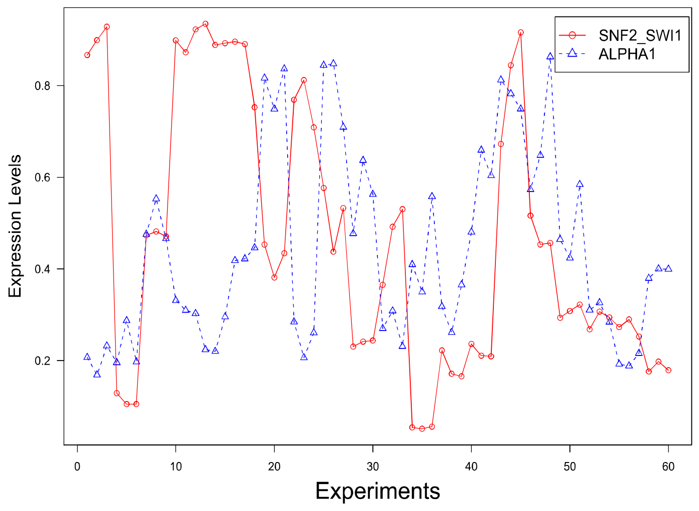

Figure 14: Correlation between the expression profiles of SNF2_SWI1 and ALPHA1 in dataset 12.

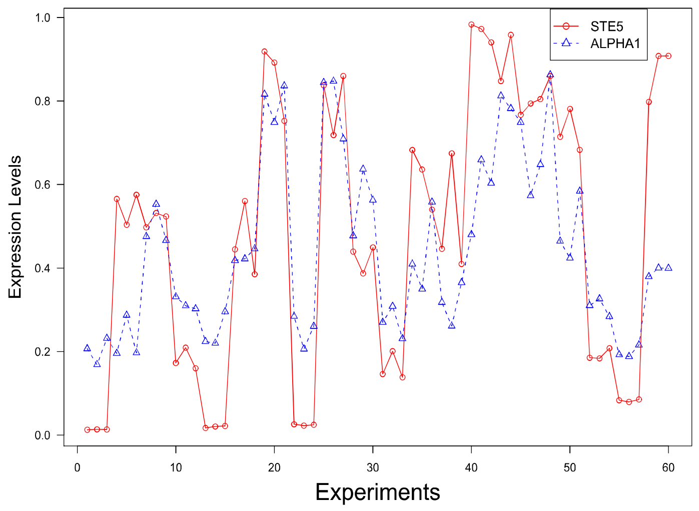

Figure 15: Correlation between the expression profiles of STE5 and ALPHA1 in dataset 12.

For example, Figure 15 shows that ALPHA1 co-expresses with STE5 that is regulated by TUP1.

We consider the correlation between ALPHA1 and STE5 as a fake correlation in contrast to a true correlation between a gene and its regulator. Fake correlations cause ALPHA1 to co-express with most genes in the network, but the tree-based sampling algorithm cannot distinguish between true correlations and fake correlations, so it ranks ALPHA1 as the most significant regulator. This problem is very common. For example, in the network shown in Figure 9, SWI4_SWI6, which is regulated by SPT16 and MBPI_SWI6, is ranked as the second most significant regulator, but it only regulates 3 genes.

4.4 Conclusion and future work

In this chapter, we introduced a Gibbs sampling approach to finding regulation programs in a transcription module network. Regulation programs of regulatory modules are represented as regression trees and a set of tree operations are defined. This set of operations are used by Gibbs samplers to sample regression trees from a posteriori distribution derived from data. Experimental results in synthetic datasets and real biological datasets indicate that this set of tree operations is sufficient to generate a Gibbs sampler with well convergence property even for large datasets. In addition, comparisons were made with two other approaches: the deterministic algorithm [106] and the fuzzy-learning algorithm [62]. The Gibbs sampling approach outperforms the deterministic algorithm based on an F-measure comparison, and achieves comparable results with those given by the fuzzy-learning algorithm.

As shown in Section 4.3.3, solely gene expression value-based methods, including the regression tree-based Gibbs sampling, deterministic algorithm and LeMoNe, cannot distinguish between true correlations and fake correlations. Our future work will concentrate on solving the problem. One solution is to combine biological prior knowledge into the learning algorithms. The method designed in [135] can be used to model prior knowledge and add it into the Bayesian score of regression trees. Another possible solution is to evaluate multiple regulators in each internal node in a regression tree, instead of only selecting a regulator and a splitting value. For example, we may build a classifier that uses several regulators as features to partition experimental conditions into two groups in each internal node.

Computational time

In synthetic datasets, we used a Dell workstation with Intel Pentium 4 processor and 3GB memory. It took around 2 minutes. The yeast stress dataset includes much more conditions that those in synthetic datasets, so we used a HP rackmount server with AMD Opteron processors (x86, 64 bit, dual core) and 16 GB memory. It took around 10 minutes for the proposed method to generate a sampler in each module in the yeast stress dataset on.

Chapter 5

Applying linear models to learn regulation programs in a transcription regulatory module network

5.1 Introduction

A common strategy of most regulation program learning algorithms in module networks [106, 62, 99] is that the regulation program of a module is represented by a regression tree. Each internal node of the tree is associated with a transcription factor and a set of conditions (i.e., a condition cluster), while each leaf node is only associated with a condition cluster. In this way, each internal node represents a contrast between the conditions covered by its left-child and right-child nodes. The confidence of assigning a transcription factor to a particular node is evaluated by the degree of differential expression that that transcription factor manifests in the contrast represented by the node. Accordingly, the overall confidence (i.e., the regulatory score) for assigning a transcription factor to a module is calculated by summing individual confidences for that transcription factor in all internal nodes of the module's tree.

A limitation of regression tree-based regulation program learning is that tree structures can represent a contrast between two condition clusters only if they are assigned to the left-child and right-child of a same internal node. Hence, regression tree-based learning might miss some biologically meaningful contrasts. In addition, the learning assumes that transcription factors are globally co-expressed to some degree with their targets, indicating that they should behave similarly across all experimental conditions in a given dataset. Global expression correlations are strong indications for regulatory relationships, but it is overstrict to demand that all regulatory relationships show this property. An obvious example is that transcription factors that are constitutive genes—cells express at a basal level in all conditions—might be locally co-expressed with their targets in some particular conditions.

In this chapter, we apply linear models to learn the regulation program of a module. Given a condition clustering of the module, instead of building a regression tree, the proposed method extracts the contrast in which the module's genes are most significantly differentially expressed, called the *critical contrast*. The differential expression under the contrast represents an important characteristic of the expression profile of the genes, so the process of learning the regulation program for the module becomes one of identifying transcription factors whose expression profiles are also associated with the characteristic. The effectiveness of the proposed method is demonstrated by applying it to two real biological datasets.

The remainder of this chapter is organized as follows: Section 5.2 describes how to apply linear models to infer regulatory relationships in module networks. Section 5.3 presents experimental results. Section 5.4 summarizes the main results and points to future work.

5.2 Inferring regulatory relationships in module networks by linear models

Given a gene module, we first obtain a condition clustering based on the expression values of genes in the module. The condition clustering represents a partition of all conditions in the data and consists of several condition clusters, each of which includes a set of conditions. Given the condition clustering, the proposed method consists of two tasks: extracting the critical contrast of the condition clustering, and inferring transcription factors based on the contrast. We use linear models to accomplish both tasks. The following subsections describe the details of each task.

5.2.1 Extracting the critical contrast of a condition clustering

The purpose of identifying the critical contrast of a condition clustering is to find, between which two condition clusters, the module's genes are most significantly differentially expressed. Consequently, we define the critical contrast as consisting of two condition clusters: the *extraordinary cluster*, in which the genes show extraordinary behaviors (i.e., extremely high or low expression values); and the *ordinary cluster*, in which the genes show ordinary behaviors.

We measure the differential expression of genes between condition clusters with the linear model described below. Suppose that in a dataset we identified a gene module M in which conditions are partitioned into two clusters: c_1 and c_2. The expression values of genes in M under the condition i can then be represented by the linear model [67]:

$$y_i = \beta_1 X_{i1} + \beta_2 X_{i2} + \varepsilon_i \tag{7}$$

where β_1 and β_2 denote regression coefficients, and ε_i is normally distributed with mean 0 and variance σ^2. X_{i1} and X_{i2} are indicator variables defined as follows:

$$X_{i1} = \begin{cases} 1 & : i \in c_1 \\ 0 & : i \notin c_1 \end{cases} ,$$

$$X_{i2} = \begin{cases} 1 & : i \in c_2 \\ 0 & : i \notin c_2 \end{cases} .$$

In this way, the degree of differential expression of the genes in M between c_1 and c_2 can be determined by the ordinary t-statistic, which is defined as:

$$t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{(n_1-1)s_1^2 + (n_2-1)s_2^2}{n_1+n_2-2}} \sqrt{\frac{n_1+n_2}{n_1 n_2}}} \tag{8}$$

where μ_1 and μ_2 are the means of the expression values of the genes in c_1 and c_2, respectively; s_1 and s_2 are the standard deviations in c_1 and c_2, respectively; n_1 and n_2 denote the numbers of conditions in c_1 and c_2, respectively.

We then apply the following search strategy to identify the critical contrast of a given condition clustering c of M. Suppose that c consists of N condition clusters. First we sort these N condition clusters into an ordered list according to the mean of the expression values in the clusters. Then, we calculate the ordinary t-statistic for the contrast between the unions of the first k condition clusters ($k = 1, 2, ..., N - 1$) and remaining $N - k$ condition clusters in the ordered list. Finally, the contrast with the maximum t-statistic among the $N-1$ contrasts is chosen as the critical contrast of c. Consequently, its associated union of condition clusters with the higher absolute mean of expression values becomes the extraordinary cluster c_e, while the other union becomes the ordinary cluster c_o.

5.2.2 Using moderated t-statistics to select differentially expressed transcription factors

Since the genes in M show different behaviors between c_e and c_o, the task of learning the regulation program of M can be accomplished by identifying transcription factors that are also dramatically differentially expressed between the same two clusters. We may apply ordinary t-statistics as defined in Equation 8 to do the work, but inferences based on the statistics might not be stable when the number of expression values in c_e or c_o is small. This is a likely situation, because we only evaluate the expression values of an individual transcription factor, instead of a set of genes.

To cope with the instability, we use a moderated t-statistic [113, 114], based on a Bayesian hierarchical model, to select differentially expressed transcription factors. Given a transcription factor r, the hierarchical model assumes a prior distribution for the variance of r (σ_r^2), which is defined as:

$$\frac{1}{\sigma_r^2} \sim \frac{1}{d_0 s_0^2} \chi_{d_0}^2 \tag{9}$$

where d_0 and s_0 are estimated by an empirical Bayes approach, and $\chi_{d_0}^2$ represents a Chi-square distribution with d_0 degree of freedom. It can be shown that the posterior mean of σ_r^{-2} is:

$$\tilde{s_r}^2 = \frac{d_0 s_0^2 + (n_e - 1)s_{re}^2 + (n_o - 1)s_{ro}^2}{d_0 + n_e + n_o - 2} \tag{10}$$

where s_{re} and s_{ro} are the standard deviations of the expression values of r in c_e and c_o, and n_e and n_o denote the numbers of conditions in c_e and c_o, respectively. The moderated t-statistic is defined by replacing the pooled variance in Equation 8 by $\tilde{s_r}$:

$$\tilde{t_r} = \frac{\mu_{re} - \mu_{ro}}{\tilde{s_r}\sqrt{\frac{n_e + n_o}{n_e n_o}}} \tag{11}$$

where μ_{re} and μ_{ro} are the means of the expression values of r in c_e and c_o, respectively. The $\tilde{t_r}$ provides more stable inference when the number of conditions is small [113], because it borrows extra information from the ensemble of genes in the dataset by using d_0 and s_0. Furthermore, in order to make $\tilde{t_r}$ comparable with moderated t-statistics based on other condition clusterings, we can normalize $\tilde{t_r}$ by:

$$\tilde{t}_{r\_standardized} = \frac{\tilde{t_r} - \mu_{\tilde{t}}}{s_{\tilde{t}}} \tag{12}$$

where $\mu_{\tilde{t}}$ and $s_{\tilde{t}}$ are the mean and standard deviation of the moderated t-statistics of all candidate transcription factors based on c.

As explained in the subsection 5.2.1, the extraordinary cluster c_e and ordinary cluster c_o may be the union of several condition clusters from the condition clustering c, such that the genes in M can be differentially expressed within c_e and c_o. However, the confidence of the assignment of r to M (Equation 11) is only relevant to the degree of the differential expression that r manifests between c_e and c_o. This indicates that it is not required for r to be globally co-expressed with genes in M in order to obtain a high confidence of it as a transcription factor of M.

5.2.3 The regulatory score for assigning a transcription factor to a module

If the expression values of genes in M can be partitioned into multiple equi-probable condition clusterings, then the overall confidence (i.e., the regulatory score) of the assignment of r may be calculated by summing the individual confidences that r shows in all condition clusterings. Hence, the regulatory score for assigning r to M over a set of condition clusterings C is defined as:

$$Z(r) = \sum_{c \in C} \tilde{t}_{cr\_standardized} \tag{13}$$

where $\tilde{t}_{cr\_standardized}$ is the standardized moderated t-statistic of r, based on a condition clustering c. We can rank all candidate transcription factors according to their regulatory scores as defined in Equation 13. The higher its ranking, the more likely a candidate transcription factor regulates M.

5.3 Experimental results and discussion

5.3.1 Experimental results in the yeast stress dataset

We applied the proposed method to a yeast dataset which measures yeast's response to various stresses, and consists of 173 experimental conditions [45]. In [63] 2355 differentially expressed genes in this dataset were clustered into 69 gene modules. We sampled 10 condition clusterings for each gene module using a Gibbs sampler [62]. Then, given the list of 321 candidate transcription factors prepared in [106], we calculated the regulatory score for assigning a transcription factor to a particular module as defined in Equation 13. Furthermore, the regulatory relationships between 185 transcription factors and 6297 genes recorded in YEASTRACT [88] (released on Apr 27, 2009) were used as the reference database to evaluate predictions given by the linear model.

5.3.1.1 Results for regulation of nitrogen utilization

In the yeast stress dataset, a module for nitrogen utilization was obtained in [62]. This module consists of 47 genes mostly involved in two pathways: the methionine pathway (regulated by MET28 and MET32), and the nitrogen catabolite repression (NCR) system (regulated by GLN3, GZF3, DAL80 and GAT1). Both pathways relate to the process by which the budding yeast uses the best available nitrogen source in the environment [77, 24].

In this module, we sampled a condition clustering with 18 clusters that were ordered descendingly by their means of expression values. As shown in Figure 16, we obtained the maximum ordinary t-statistic (38.98) when we compared the union of the first 3 condition clusters with the remaining clusters in the ordered list. This indicates that the extraordinary cluster of the clustering's critical contrast includes those conditions under nitrogen depletion and amino-acid starvation where using non-preferred nitrogen sources is crucial, while the ordinary cluster consists of the remaining conditions. Accordingly, the critical contrast represents the comparison of the genes' behaviors under preferred and non-preferred nitrogen sources.

Figure 17 shows that the module's genes are dramatically differentially expressed in the contrast. That is, they are only highly expressed under non-preferred nitrogen sources (i.e., the extraordinary cluster). Similar results were obtained for the critical contrasts of the other nine condition clusterings of the module.

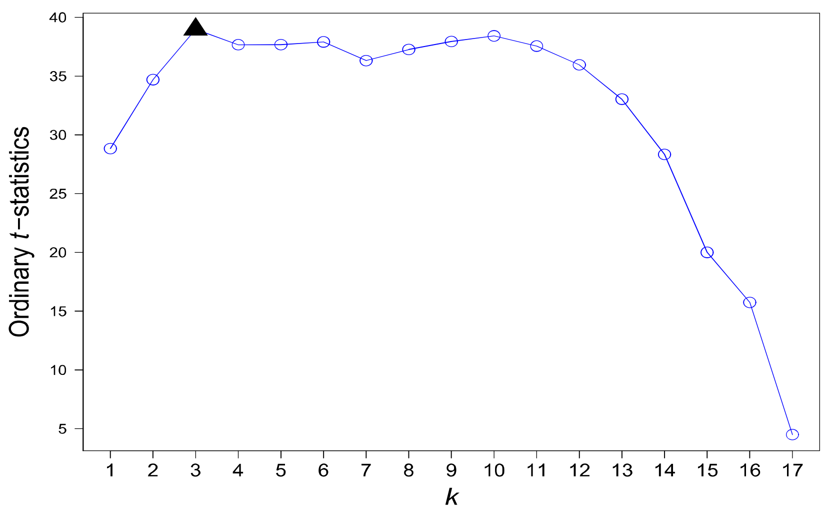

Figure 16: Ordinary t-statistic for the contrast between the union of the first k condition clusters ($k = 1, 2, ..., 17$) and the remaining $18 - k$ clusters. The horizontal axis gives the values of k. The colored triangle represents the largest ordinary t-statistic.

We then ordered candidate transcription factors according to their regulatory scores as defined in Equation 13. Table 2 shows the top ten regulators as ranked by the linear model, which includes most known transcription factors of the NCR process and the methionine pathway.

5.3.1.2 Linear model versus LeMoNe in the NCR process

Furthermore, we compared the predictions for the module studied in the Section 5.3.1.1 given by the linear model and by the LeMoNe regression tree-based method [62]. As shown in Table 2, both methods identify most known transcription factors of the module, but they ranked the transcription factors of the NCR process (denoted with *) differently. DAL80, GLN3, GZF3, and GAT1 are the first, fifth, eighth, and ninth regulators in the rank by the linear model. However, LeMoNe ranks GAT1, DAL80, GZF3 as the first, fourth and fourteenth regulators, and most strikingly, GLN3 is out of the top 100. We next investigated why the two methods show very different confidences on the assignments of GLN3 and GAT1 to this module.

Given the condition clustering we studied in the Section 5.3.1.1, LeMoNe builds a

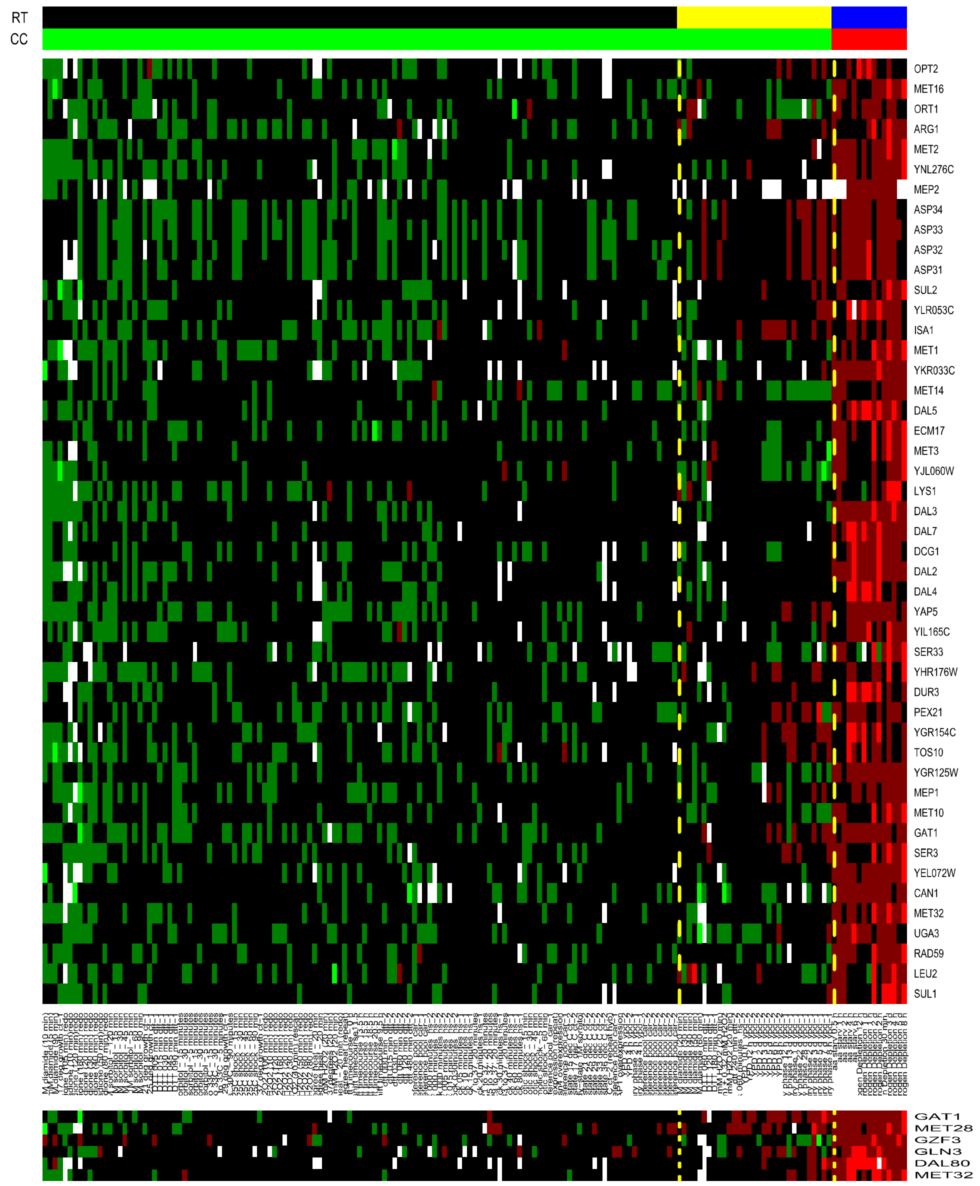

Figure 17: Heatmaps of expression values of genes in the module for nitrogen utilization in the yeast stress dataset (top), and known transcription factors of the module (bottom). In track CC (critical contrast) conditions assigned to the extraordinary and ordinary clusters are colored by red and green, respectively. In track RT (regression tree) conditions assigned to cluster1, cluster2, and cluster3 (detailed in Figure 18) are colored by black, yellow, and blue, respectively.

Algorithm	Linear model		LeMoNe	
Rank	Regulator	Number of genes regulated	Regulator	Number of genes regulated
1	**DAL80***	10	**GAT1***	7
2	MET32	13	MET28	8
3	UGA3	1	MET32	13
4	LYS14	1	**DAL80***	10
5	**GLN3***	18	UGA3	1
6	YAP5	3	THI2	0
7	MET28	8	YAP5	3
8	**GZF3***	6	CMP2	0
9	**GAT1***	7	GCN20	0
10	DAL82	9	INO2	1

Table 2: Top 10 transcription factors for the module of nitrogen utilization in the yeast stress dataset as inferred by the linear model and LeMoNe, and the number of genes they regulate in the module according to records in YEASTRACT. * regulators are known transcription factors of NCR process.

regression tree as shown in Figure 18, in which we focus on three condition clusters: cluster1; cluster2, mainly consisting of conditions in stationary phase; and cluster3, including the conditions under nitrogen depletion and amino acid starvation (i.e., the extraordinary cluster identified by the liner model). As seen in Figure 17, genes in the module are not expressed in cluster1, and slightly expressed in cluster2, but significantly expressed in cluster3. We speculate that the conditions in cluster2 represent a transition from utilizing preferred nitrogen sources to non-preferred nitrogen sources. During the transition, preferred nitrogen sources become less and less available, such that NCR related genes are expressed to some degree, but the expressed amount is much less than that under non-preferred nitrogen sources (e.g., conditions in cluster3).

As shown in Figures 19 and 20, the major distinction between the expression profiles of GAT1 and GLN3 is that GAT1 is significantly differentially expressed between cluster1 and cluster2 (p-value = 9.675e-09 for the two sample t-test) that is similar to NCR related genes, but compared to GAT1, GLN3 is much less significant (p-value = 0.08 for the two sample t-test). We consider that the distinction is due to their expressions being controlled by different mechanisms. In the presence of a preferred nitrogen source, cells express

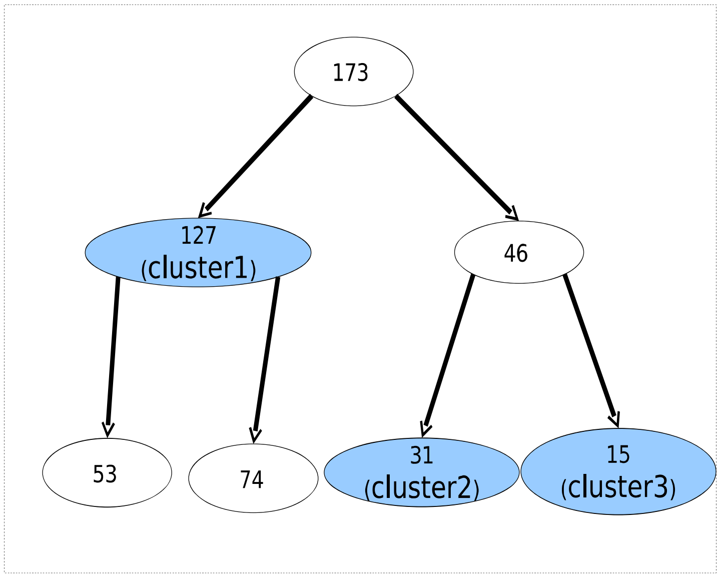

Figure 18: Top 3 levels of the LeMoNe's regression tree built on a condition clustering of the module for nitrogen utilization. A condition cluster is represented by a circle containing its number of conditions. Three of the clusters have been given labels for easy reference in the text.

GLN3 at a basal level, but it is located in the cytoplasm due to the protein-protein interaction with URE2. When preferred nitrogen sources are absent and only a non-preferred nitrogen source is available, GLN3 is translocated from the cytoplasm to the nucleus where GLN3 activates the transcription of GAT1 and NCR related genes [24]. Hence, GLN3 shows a constitutive expression profile, and as a result it is only locally co-expressed with NCR related genes in cluster3. In contrast, GAT1's activity is basically controlled by transcriptional regulation, so its expression profile is essentially very close to that of a typical NCR related gene [23], and it is globally co-expressed with its targets.

The confidence of assigning a transcription factor to the module by LeMoNe is mainly determined by the degree of the transcription factor's differential expression in the contrast between cluster1, and the union of cluster2 and cluster3 (i.e., the contrast represented by the root node of the regression tree in Figure 18). GAT1 is more significantly differentially expressed in the contrast (p-value $<$ 2.2e-16 for the two sample t-test) than GLN3 (p-value = 9.043e-06 for the two sample t-test), and consequently LeMoNe ranks GAT1 much higher than GLN3.

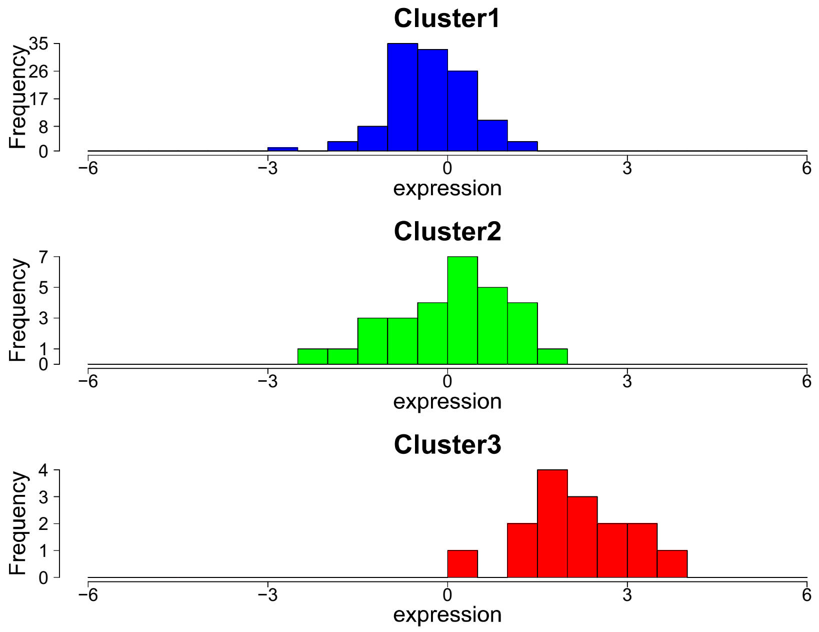

Figure 19: Histogram of expression values of GLN3 in condition clusters cluster1, cluster2 and cluster3.

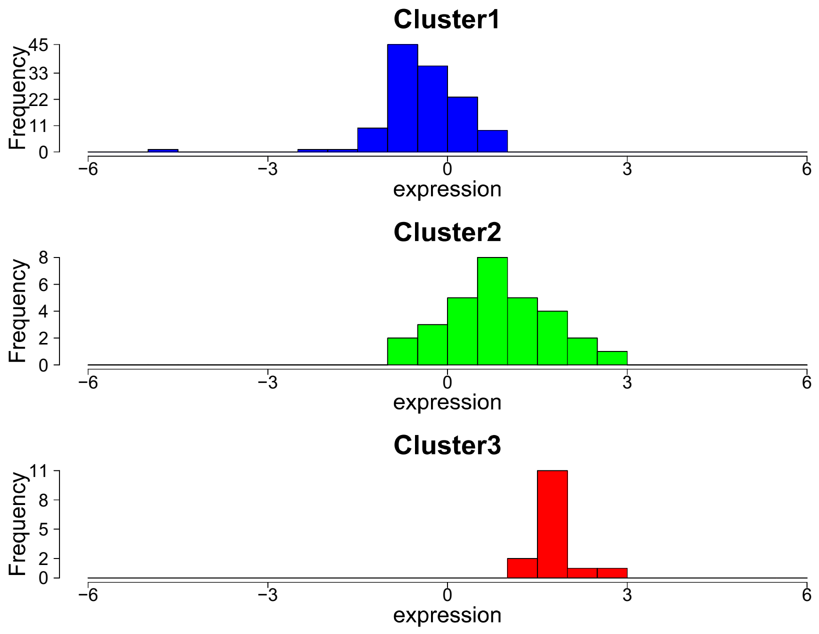

Figure 20: Histogram of expression values of GAT1 in condition clusters cluster1, cluster2 and cluster3.

On the other hand, as explained in Section 5.3.1.1, the linear model searches for transcription factors that are differentially expressed in the contrast between cluster3, and the union of cluster1 and cluster2 (i.e., between conditions under non-preferred nitrogen sources and the other conditions). Genes in the module are involved in the process by which the yeast uses the best available nitrogen source, so the differential expression in this contrast is the most important property of the genes' expression profile. But as the contrast can not be directly represented by LeMoNe's regression tree, it gives low confidence for the assignment of two known regulators (GLN3 and GZF3) to the module.

The experimental results in the module suggest that LeMoNe and the linear model rely on different contrasts to infer transcription factors. As a result, they show distinct preference in detecting regulatory relationships. Their different preferences in assignment of regulators are also demonstrated by their results in the dataset as shown in the follows.

5.3.1.3 Results over the entire yeast stress dataset

Looking deeper, we compared the regulator *gene-wise* performance of the proposed method and LeMoNe over the entire yeast stress dataset. We applied each method to the dataset to calculate the regulatory score for assigning a regulator to a module. Then we ordered all of the method's regulatory scores between 321 candidate transcription factors and 69 modules in descending order. This lead to a ranked list of 22,149 regulator-module interactions for the method. Furthermore, in order to convert regulator module-wise predictions into regulator gene-wise predictions, we made the simplifying assumption that the regulator of each module-wise prediction regulates all genes in the module.

Following the aforementioned strategy, the top 200 regulator module-wise predictions from the linear model yielded 4993 regulator gene-wise predictions. The closest number of gene-wise predictions yielded by LeMoNe (5021) are produced by its top 191 module-wise predictions. Taking these gene-wise predictions from LeMoNe and the linear model, we get Figure 21 showing the precision versus recall curves for these two methods. The precision and recall of the top i regulator gene-wise predictions from a method are defined as:

$$precision_{gene-wise}(i) = \frac{TP(i)}{i}, \tag{14}$$

and

$$recall_{gene-wise}(i) = \frac{TP(i)}{P}, \tag{15}$$

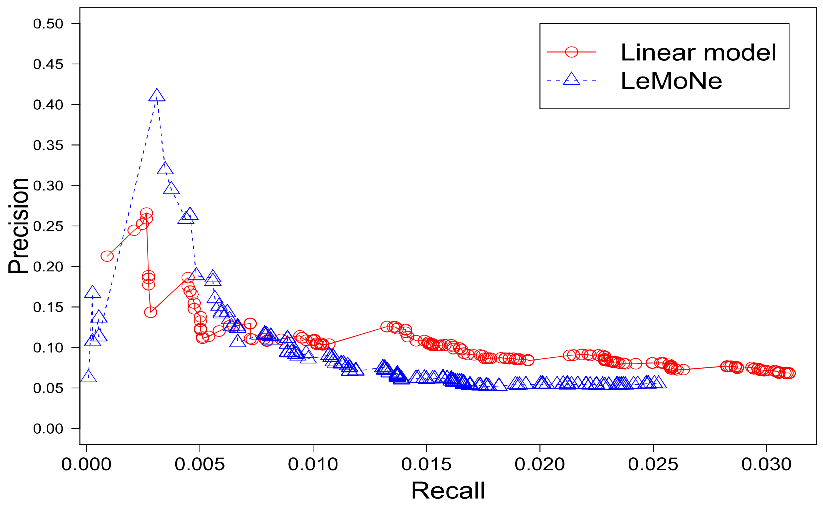

Figure 21: Precision versus recall curves for the linear model and LeMoNe in the yeast stress dataset. Precisions and recalls are defined as Equations 14 and 15, respectively.

where $TP(i)$ represents the number of regulator-gene interactions in the top i predictions recorded in YEASTRACT, and P gives the total number of interactions recorded in YEASTRACT.

In general, LeMoNe and the proposed method achieve comparable performance in the dataset (with areas under the curves of 0.0028 versus 0.0033), but we observe that they retrieve very different parts of the transcriptional regulatory networks in the yeast. For example, in Table 3 which shows the top 10 predictions given by two methods, the only overlapped prediction is the assignment of DAL80 to module 51. In LeMoNe it is the second prediction, while in the linear model it is the fourth prediction. The difference is because LeMoNe and the linear model depend on distinct contrasts to infer regulators of modules (i.e., select differentially expressed transcription factors). The difference also suggests that combining the predictions given by these two methods might be a promising direction.

5.3.2 Experimental results in the *Escherichia Coli (E. coli)* dataset

The *E. coli* dataset [35] contains expression values for 4345 genes under 189 conditions. In this dataset, we first clustered 1882 genes with standard deviations more than 0.5 into

Algorithm	Linear model		LeMoNe	
Rank	Regulator	Module	Regulator	Module
1	DAL80	11	PDR3	13
2	MET32	11	DAL80	51
3	PHD1	36	USV1	28
4	DAL80	51	HAP4	30
5	DAL82	48	IME4	46
6	UGA3	11	HAP4	7
7	ACA1	48	XBP1	10
8	DAL80	40	TOS8	24
9	LYS14	11	GAT1	11
10	GLN3	11	GAL80	41

Table 3: Top 10 inferred regulatory relationships by the linear model and LeMoNe in the yeast stress dataset

70 gene modules using 100 independent Gibbs sampler runs [63]. Then, we sampled 10 condition clusterings for each gene module. Furthermore, using the list of 316 transcription factors prepared in [86] as the candidate transcription factors of these modules, we calculated the regulatory score for assigning each transcription factor in this list to a particular module as defined in Equation 13. Last, we evaluated the results by the regulatory relationships recorded in RegulonDB [44].

5.3.2.1 Linear model versus LeMoNe in the flagellum chemotaxis system

E. coli is capable of using its chemotaxis sensory system to detect environmental signals. These detected signals are then used to direct its flagellar motors so that it can move towards the source of nutrient in environments [34]. The transcription of genes involved in the chemotaxis and flagella system in *E. coli* is mainly regulated by three transcription factors: FLHC, FLHD and FLIA [86]. FLHC and FLHD are the master regulators of the system because FLIA is regulated by them [110].

In the *E. coli* dataset, we identified three modules (3, 24 and 36) related to chemotaxis and flagellar system. The linear model predicts FLHC, FLHD and FLIA as the top 3 regulators for each of these modules, but LeMoNe misses FLHD in all of these modules (Tables 4-6).

We further analyzed the results of these two algorithms in module 3. In this module, we

Top 3 regulators as ranked by the linear model			
Rank	1	2	3
Regulator	FLIA	FLHC	FLHD
Number of genes regulated	33	30	30

Top 3 regulators as ranked by LeMoNe			
Rank	1	2	3
Regulator	FLIA	TRER	EVGA
Number of genes regulated	33	0	0

Table 4: Top 3 transcription factors for module 3 in the *E. coli* dataset as inferred by the linear model and LeMoNe, and the number of genes they regulate in the module according to records in RegulonDB. The module consists of 44 genes.

Top 3 regulators as ranked by the linear model			
Rank	1	2	3
Regulator	FLIA	FLHC	FLHD
Number of genes regulated	5	6	6

Top 3 regulators as ranked by LeMoNe			
Rank	1	2	3
Regulator	CELD	FLIA	FLHC
Number of genes regulated	0	5	6

Table 5: Top 3 transcription factors for module 24 in the *E. coli* dataset as inferred by the linear model and LeMoNe, and the number of genes they regulate in the module according to records in RegulonDB. The module consists of 12 genes.

Top 3 regulators as ranked by the linear model			
Rank	1	2	3
Regulator	FLHD	FLIA	FLHC
Number of genes regulated	1	2	1

Top 3 regulators as ranked by LeMoNe			
Rank	1	2	3
Regulator	FLIA	CYSB	FLHC
Number of genes regulated	2	0	1

Table 6: Top 3 transcription factors for module 36 in the *E. coli* dataset as inferred by the linear model and LeMoNe, and the number of genes they regulate in the module according to records in RegulonDB. The module consists of 4 genes.

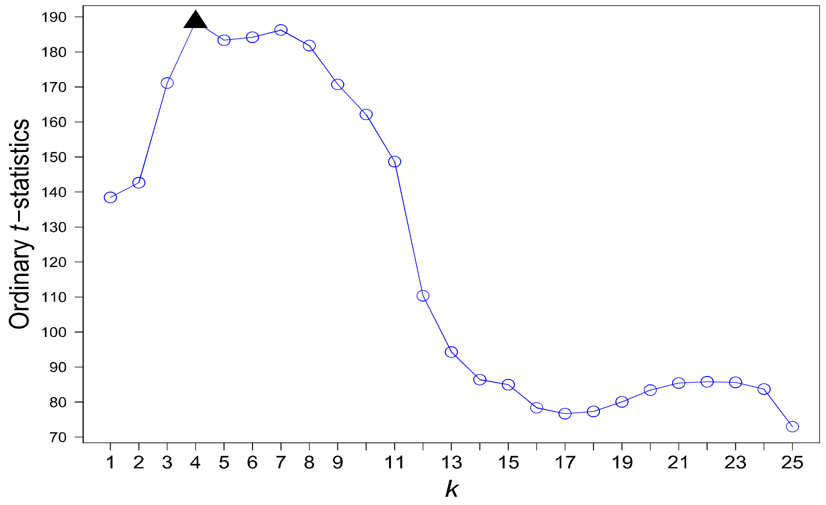

Figure 22: Ordinary t-statistic for the contrast between the union of the first k condition clusters ($k = 1, 2, ..., 25$) and the remaining $26 - k$ clusters. The horizontal axis gives the values of k. The colored triangle represents the largest ordinary t-statistic.

sampled a condition clustering with 26 clusters, and they were sorted into an ordered list by their means of expression values. We obtained the maximum ordinary t-statistic (188.60) when we compared the union of the first 4 condition clusters with the remaining clusters in the ordered list (Figure 22). The structure of the regression tree built by LeMoNe in the module is similar to that of the tree shown in Figure 18, but cluster1, cluster2 and cluster3 consist of 149, 13, and 27 conditions, respectively. LeMoNe searches for transcription factors differentially expressed in the contrast between cluster1, and the union of cluster2 and cluster3. FLHD and FLHC are the 25th and 14th in the ordered list given by LeMoNe in this module, and they are assigned much less confidence for regulating this module than that of FLIA (1st regulator in LeMoNe's ordered list). This is because the coexpression of FLHD and FLHC with the genes in this module is significantly lower than that for FLIA [86]. In other words, unlike FLIA, FLHD and FLHC are not globally co-expressed with their targets. However, FLHD and FLHC are strongly differentially expressed in the critical contrast used by the linear model (i.e., the contrast between cluster3 and the union of cluster1 and cluster2) (Figure 23), and consequently they are assigned high confidence for regulating this module by the linear model.

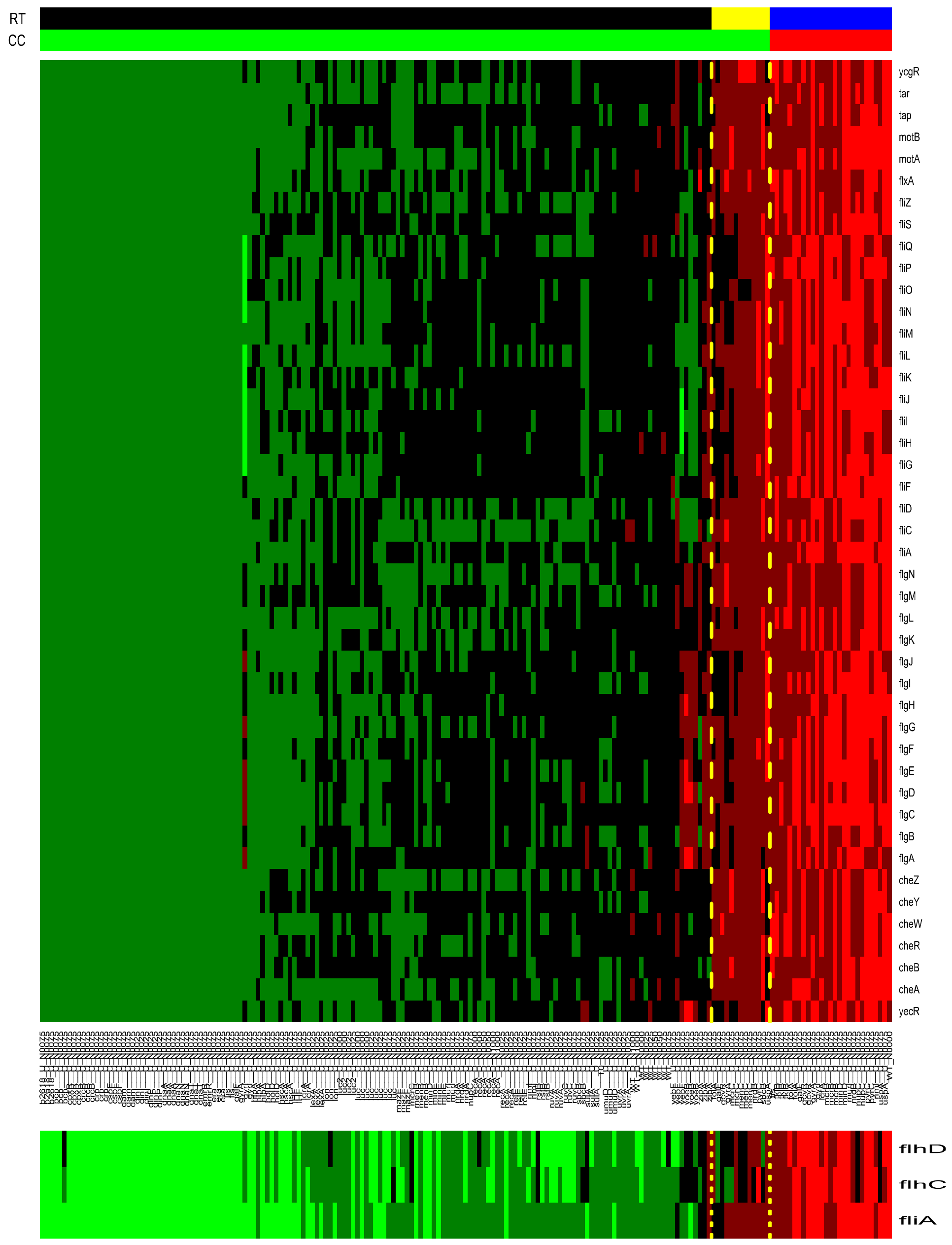

Figure 23: Heatmaps of expression values of genes in module 3 in the *E. coli* dataset (top), and known transcription factors of the module (bottom). In track CC (critical contrast) conditions assigned to the extraordinary and ordinary clusters are colored by red and green, respectively. In track RT (regression tree) conditions assigned to cluster1, cluster2, and cluster3 (detailed in Figure 18) are colored by black, yellow, and blue, respectively. Note that the numbers of conditions included in cluster1, cluster2 and cluster3 are 149, 13, and 27, respectively, instead of those shown in Figure 18

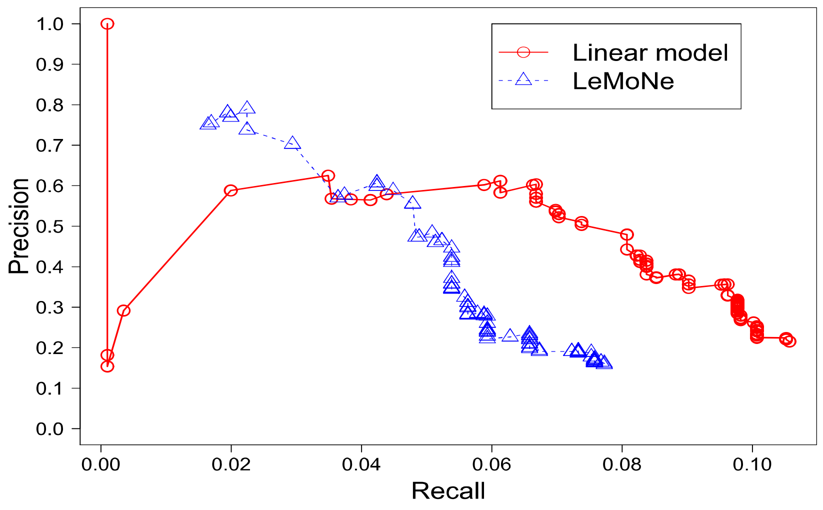

Figure 24: Precision versus recall curves for the linear model and LeMoNe in the *E. coli* dataset. Precisions and recalls are defined as Equations 14 and 15, respectively.

5.3.2.2 Results over the entire *E. coli* dataset

Similar to the comparison in Section 5.3.1.3, we evaluated the performance of the linear model and LeMoNe using precision versus recall curves based on their results in the entire *E. coli* dataset (Figure 24). The top 100 regulator module-wise predictions from the linear model were transformed to 985 regulator gene-wise predictions. The closest number of gene-wise predictions from LeMoNe are produced by its top 95 module-wise predictions. The precision and recall of the top i regulator gene-wise predictions from a method are defined as Equations 14 and 15. The linear model yields slightly better performance than LeMoNe.

5.4 Conclusion and future work

In this chapter, we proposed to apply linear models to infer regulators in transcriptional module networks. Given a gene module, the proposed method identifies the critical contrast of this module, which consists of two condition clusters. Since an important characteristic of genes in this module is that they are significantly differentially expressed between these two clusters, the proposed method searches for transcription factors also associated with this characteristic as the regulators of this module. The novelty of this strategy is that it does not rely on regression trees, which have been widely used to infer regulators of gene modules. Based on the analysis of two modules in the yeast and *E. coli*, we show that the structure of regression trees may restrict regression tree-based methods from identifying some regulatory relationships. In addition, the proposed simple linear model is capable of achieving comparable results with LeMoNe, a well known regression tree-based algorithm, based on their overall performance on two real biological datasets.

One direction for future work is to extend the linear model to use ensemble methods [18]. In the module for nitrogen utilization in the yeast stress dataset, the highest t-statistic value (38.98) is achieved when k=3 (Figure 16). However, when $k = i$ ($i = 4, 5, ..., 10$), the corresponding t-statistic is only slightly smaller than 38.98. This indicates that the genes in this module are also highly differentially expressed in these contrasts. In addition, similar results are observed in module 3 in the *E. coli* dataset (Figure 22). Hence, it may be promising to calculate regulatory scores of transcription factors based on their differential expression in an ensemble of multiple contrasts.

Another direction for future work is to keep each experimental condition individually, instead of clustering conditions into condition clusters. In other words, each condition is considered to be a condition cluster. This simplification can significantly reduce the computational workload of the proposed method, because condition clustering is very time-consuming.

Computational time

It takes around 20 minutes for the proposed linear model to identify critical contrasts and infer transcription factors of all modules in the yeast stress dataset on a Dell workstation with Intel Pentium 4 processor and 3GB memory. The *E. coli* dataset requires 30 minutes.

Chapter 6

An integrative approach to infer regulation programs in a transcription regulatory module network

6.1 Introduction

Many techniques have been applied to infer the regulation program of a given gene module, such as logistic regression [62], t-statistics [100], Gibbs sampler [99], and linear regression [11]. A common characteristic of these methods is that they are able to calculate the confidence (i.e., regulatory score) for the assignment of a transcription factor to a gene module, which is referred to as a regulator-module interaction. Consequently, their results can be sorted into an ordered list of regulator-module interactions according to their regulatory scores. The higher the ranking of a regulator-module interaction in the ordered list given by a method, the more confidence this method assigns to the interaction. In addition, since these methods resort to distinct techniques, they show very different biases in detecting regulatory relationships. For example, in the nitrogen utilization module in the budding yeast, LeMoNe [62] favors regulatory relationships where transcription factors and genes are globally co-expressed, while the LIMMA-based method [100] favors regulatory relationships where transcription factors and genes are locally co-expressed. This suggests that integrating results from different regulation program learning algorithms can be a promising direction in better inferring regulatory networks.

In this chapter, we extend our previous work [100] by integrating its results with those given by two other learning algorithms [62, 11]. The integration methods we select are union, intersection, and weighted rank aggregation [95]. Experimental results indicate that the union and weighted rank aggregation methods produce more accurate predictions than those given by individual algorithms, whereas the intersection method does not yield any improvement in the accuracy of predictions.

The rest of this chapter is organized as follows: Section 6.2 describes the dataset, integration methods, and regulation program learning algorithms studied in this chapter. Section 6.3 presents experimental results. Section 6.4 summarizes the main results and discusses future work.

Note that the assignment of a regulator to a module is associated with a p-value for each regulation program learning algorithm, and the p-value is required by the weighted rank aggregation method to integrate results from different learning algorithms. In contrast, the assignment is also associated with a p-value based on records in the reference database, YEASTRACT. The p-value is calculated by the hypergeometric distribution and is used to evaluate the performance of individual learning algorithms and integration methods.

6.2 System and method

6.2.1 Data set and reference database

The yeast stress dataset has been used as a benchmark to validate the performance of module network learning algorithms [106, 62]. In previous work [106, 63], 2355 differentially expressed genes in the dataset were selected and these genes were clustered into 69 gene modules. In this work, we apply three algorithms [62, 11, 100] to infer regulators of these modules using a list of 321 transcription factors prepared by Segal $et\ al.$ [106] as candidate transcription factors. Then, we integrate the results of these algorithms by methods described in Section 6.2.2. The regulatory relationships recorded in YEASTRACT [88] (released on Apr 27, 2009) are used as the reference database to validate results given by individual algorithms and our integration methods.

6.2.2 Integration methods

We apply union, intersection, and weighted rank aggregation integration methods to integrate results from different regulation program learning algorithms. The union and intersection methods are straightforward. The former determines the ranking of a regulator-module interaction using the highest ranking given by all candidate learning algorithms. In contrast, the latter determines the ranking of an interaction using the lowest ranking. For example, given a regulator-module interaction, which is the 1st, 3rd and 5th items in rankings given by three individual learning algorithms, respectively, the union method assigns 1st as its ranking, while the intersection method assigns 5th as its ranking. After determining the ranking of each interaction, these methods can each produce an ordered list of regulator-module interactions by sorting interactions by their rankings.

In comparison, the weighted rank aggregation method [95] is much more computationally intensive than the union and intersection methods. Given a set of learning algorithms M, this integration algorithm searches for an ordered list δ^* that is simultaneously as close as possible to the list produced by each algorithm in M. Let $L_m = (A_1^m, A_2^m, ..., A_k^m)$ represents an ordered list of k regulator-module interactions produced by the algorithm m. Let $r^m(A)$ denote the rank of the interaction A under m. Finally, let $m(i)$ $(i = 1, 2, ..., k)$ denote the p-value (weight) that algorithm m assigns to the interaction ranked at the ith position in the ordered list. This can be represented by the following minimization problem:

$$\delta^* = \text{ arg min } \Phi(\delta),$$

where

$$\Phi(\delta) = \sum_{m \in M} d(\delta, L_m) \tag{16}$$

represents the sum of the distances between an ordered list δ and the lists from all algorithms. The distance between δ and L_m is determined by the weighted Spearman's footrule distance:

$$d(\delta, L_m) = \sum_{A \in L_m \cup \delta} \left| m\left(r^\delta(A)\right) - m\left(r^m(A)\right) \right| \times \left| r^\delta(A) - r^m(A) \right|.$$

To determine δ^*, we apply the cross-entropy Monte Carlo algorithm [96]. This algorithm represents an ordered list by a random matrix whose entries are 0 or 1. The matrix follows two constraints: the sum of each column is 1 and the sum of each row is at most 1. Given a current random matrix (ordered list), a new matrix is generated by shuffling entries in the current matrix subject to above two constraints. This sampled matrix is used to update entries in the current matrix so that the value of Equation 16 is reduced. This sampling procedure repeats utile it reaches convergence.

6.2.3 Regulation program learning algorithms

We select LeMoNe [62], Inferelator [11], and the LIMMA-based method [100], as candidate regulation program learning algorithms. In this subsection, we describe how to apply these algorithms to the yeast stress dataset. In addition, in order to apply the weighted rank aggregation to integrate their results, for each algorithm, we also define how to calculate the p-value for the assignment of a regulator to a module.

6.2.3.1 LeMoNe

For each gene module in the yeast stress dataset, LeMoNe [62] sampled 10 regression trees, and then calculated regulatory scores for assigning transcription factors to this module based on these trees. Regulatory scores of all regulator-module interactions were downloaded from the supplementary website of [62].

We calculate the LeMoNe-based p-value for the assignment of a regulator r to a module as follows. First, given a regression tree T of this module, we define the p-value of the split with r and a splitting value z at an internal node t in T (i.e., $p\text{-value}_{(t)}(r, z)$) as the probability of observing a split with a higher average prediction probability than this split at the node t. The average prediction probability of a split is defined as in Equation 4 of [62]. Then, the p-value for assigning r to t is defined as:

$$p\text{-value}_{(t)}(r) = \frac{w_t}{|Z|} \sum_{z \in Z} p\text{-value}_{(t)}(r, z)$$

where w_t is the number of experimental conditions in t divided by the total number of conditions in the data, and Z represents the set of possible splitting values for r in t. Furthermore, given a set of regression trees F, the LeMoNe-based p-value for assigning r to this module can be calculated as:

$$p\text{-value}(r) = \frac{1}{|F|} \sum_{T \in F} \sum_{t \in T} p\text{-value}_{(t)}(r)$$

6.2.3.2 Inferelator

Inferelator [11] uses linear regression and variable selection to identify transcription factors of gene modules. In each gene module in the yeast stress dataset, we fit a linear model to the mean of the module's genes in each condition using the 321 candidate transcription factors as predictor variables. The regulatory score for assigning a regulator to the module is decided by the absolute value of the regulator's regression coefficient in the fitted model.

The Inferelator-based p-value for the assignment of a regulatory to a module is defined as follows. First, we permute the values of the expression value matrix from the row direction (gene). Second, we apply Inferelator to the permuted dataset using the original gene modules. Third, we fit the distribution of nonzero coefficients obtained from the permuted dataset by the Weibull distribution defined as:

$$pdf(x) = \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k} \tag{17}$$

with $k = 0.889$ and $\lambda = 0.015$ (Figure 25). Last, we define the Inferelator-based p-value for a regulator-module interaction with a regulatory score S as the probability of observing a value more than S from the Weibull distribution (Equation 17).

6.2.3.3 LIMMA-based method

In our previous [100], moderated t-statistics proposed in LIMMA [113] were applied to infer transcription factors of gene modules. For each gene module in the yeast dataset, ten condition clusterings were sampled by a two-way clustering algorithm [63]. Then the regulatory score for assigning a transcription factor to this module was calculated by summing the transcription factor's standardized moderated t-statistics based on the sampled condition clusterings.

We next describe how to define the method's p-value for the assignment of a transcription factor to a module. First, we randomly generate ten condition clusterings, each of which consisted of two clusters. We then calculate the regulator score for each candidate transcription factor based on these randomly generated clusterings. Moreover, we record the regulatory score of a randomly selected transcription factor. The above process is repeated to obtain 100,000 randomly generated regulatory scores. Last, the probability density function of these randomly generated scores is approximated by the stretched

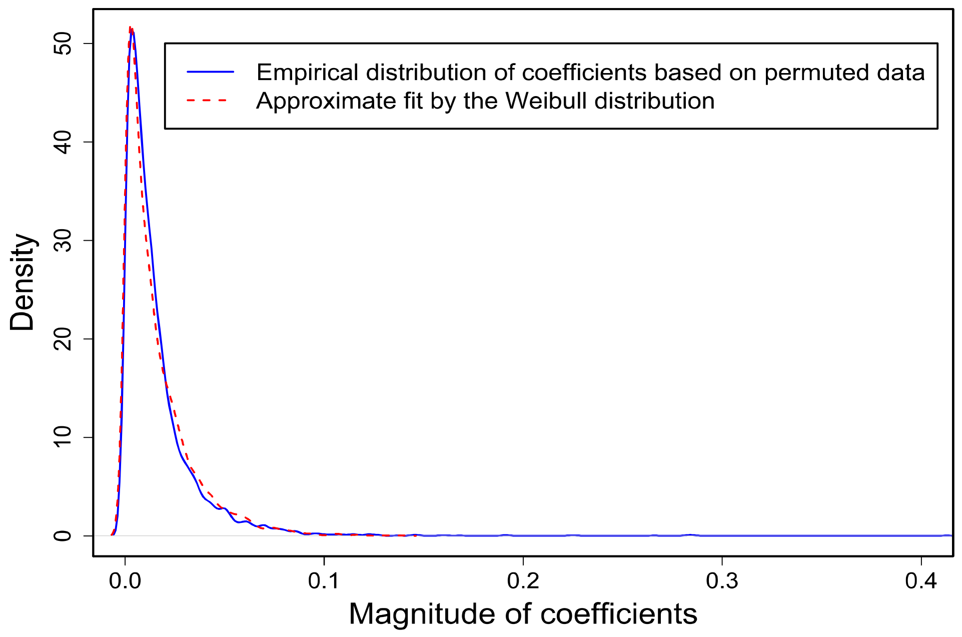

Figure 25: Probability density function of coefficients (regulatory scores) based on permuted data and the approximated fit by the Weibull distribution. The solid line denotes the empirical probability density function of regression coefficients obtained by Inferelator in permuted expression data, while the dotted line denotes the probability density function of the Weibull distribution (Equation 17) with $k = 0.889$ and $\lambda = 0.015$.

exponentials [115] defined as:

$$pdf(x) = \begin{cases} h_{max} \exp\left[-b_r(x - x_{max})^{c_r}\right] & \text{for } x \geq x_{max} \\ h_{max} \exp\left[-b_l(x_{max} - x)^{c_l}\right] & \text{for } x < x_{max} \end{cases} \tag{18}$$

with $h_{max} = 0.127$, $b_r = 0.024$, $b_l = 0.083$, $c_r = 2.45$, $c_l = 1.70$ and $x_{max} = -0.050$. As shown in Figure 26, the approximated fit is very close to the empirical distribution of the randomly generated regulatory scores, so the p-value for a regulator-module interaction with a regulatory score S can be defined as the probability of observing a value more than S from the approximated fit.

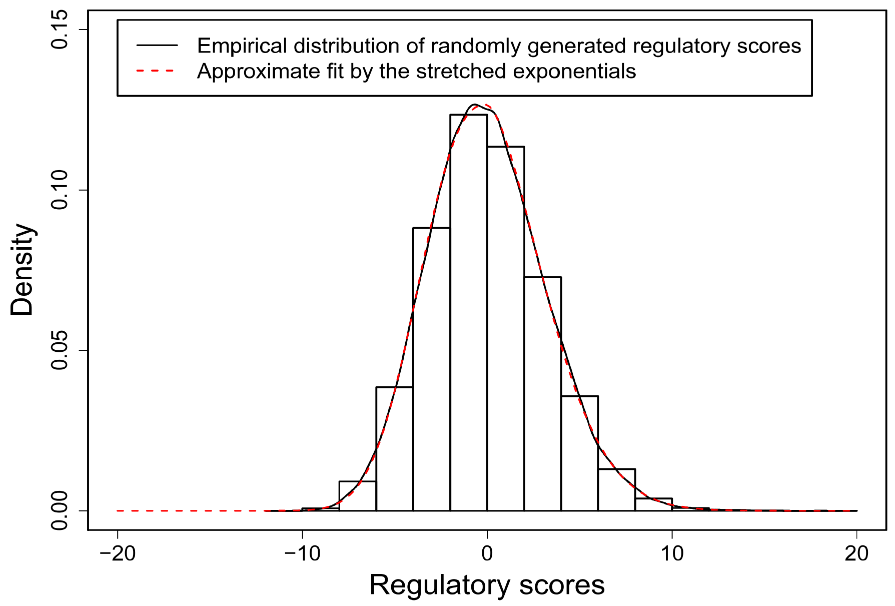

Figure 26: Probability density function of randomly generated regulatory scores and the approximated fit by the stretched exponentials. The solid line denotes the empirical probability density function of regulatory scores obtained by the LIMMA-based method based on randomly generated condition clusterings, while the dotted line denotes the stretched exponentials (Equation 18) with $h_{max} = 0.127$, $b_r = 0.024$, $b_l = 0.083$, $c_r = 2.45$, $c_l = 1.70$ and $x_{max} = -0.050$.

6.3 Experimental results and discussion

6.3.1 Results of individual learning algorithms

We applied each regulation program learning algorithm described in Section 6.2.3 to calculate the regulatory score for assigning a regulator to a module. Then we sorted all of its regulatory scores between 321 candidate transcription factors and 69 modules in the descending order. This led to an ordered list of 22,149 regulator-module interactions for each method.

In addition, for each regulator-module interaction, we used the hypergeometric distribution to calculate the p-value of this interaction, using regulatory relationships in YEASTRACT as the reference database. This p-value is based on the number of genes regulated by the regulator in the dataset, the number of genes regulated by the regulator in the module, and the number of genes in the module. Note that the p-values defined in Section 6.2.3 are used as weights by the weighted rank aggregation algorithm, while the p-values based the hypergeometric distribution defined here are used to determine if regulator-module interactions are true positives.

Moreover, for a given ordered list of regulator-module interactions, we define the precision of the top i items in this ordered list as:

$$p(i) = \frac{T(i)}{i}, \tag{19}$$

where $T(i)$ denotes the number of interactions with p-values less than 0.05 in the top i items (i.e., the number of true positives among these i interactions).

In Figure 27, we show the precisions of the top i regulator-module interactions ($i = 1, 2, ..., 100$) in the ordered lists obtained by Inferelator, LeMoNe and the LIMMA-based method. When less than 20 interactions are selected, the LIMMA-based method outperforms the other two methods. However, Inferelator and LeMoNe outperform the LIMMA-based method when the number of selected interactions is in the range of 20 and 50. In addition, when more than 50 interactions are selected, the three methods show similar performance in the yeast dataset.

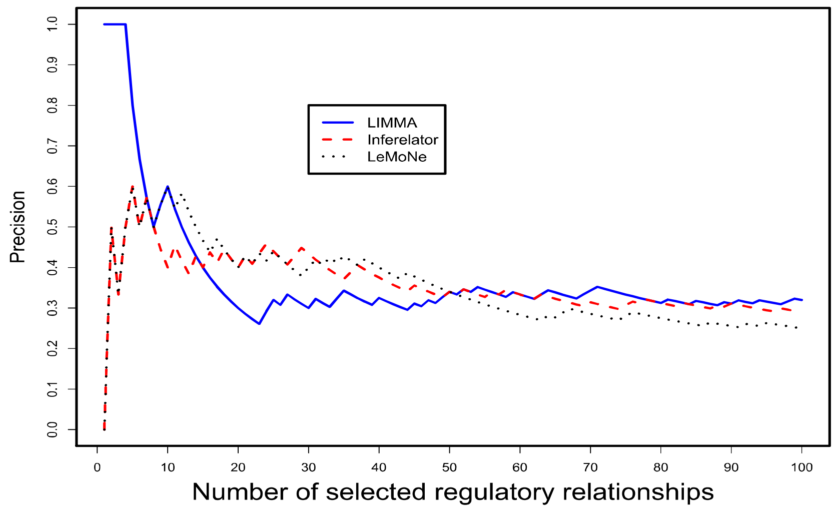

Figure 27: Comparison of precision of three candidate learning algorithms. For each algorithm, the figure shows the precision (Equation 19) when the top i ($i = 1, 2, ..., 100$) regulator-module interactions in the rank given by the algorithm are selected.

6.3.2 Results for the weighted rank aggregation

The weighted rank aggregation method searches for a synthesized list that is simultaneously as close as possible to the ordered lists from LeMoNe, Inferelator, and the LIMMA-based method. However, it is not feasible to directly apply this integration method on a list with 22,149 interactions due to the extensive computational workload. Hence, we resort to a tradeoff by integrating the top k ($k \leq 22,149$) interactions in the ordered lists given by these algorithms. This is, for a given ordered list and k, interactions ranked lower than k (i.e., $k+1, k+2, ..., 22,149$), are associated with a same weight (p-value) of one. The larger k, the closer the list produced by the rank aggregation is to the lists given by the three candidate algorithms, but the rank aggregation costs more computation time. For example, it takes 12 and 48 hours for $k = 75$ and $k = 100$ on a HP rackmount server with AMD Opteron processors (x86, 64 bit, dual core) and 16 GB memory.

In order to select a proper value for k in the yeast dataset, we applied the rank aggregation ten times for $k = 25, 50, 75, 100$, respectively. For each k, this led to 10 ordered lists, and we calculated the average of the precisions of the top i ($i = 1, 2, ..., k$) interactions in these ten lists. As shown in Figure 28, when k increases from 25, to 50 and then to 75, the precisions obtained by the rank aggregation method are improved, but the precisions at

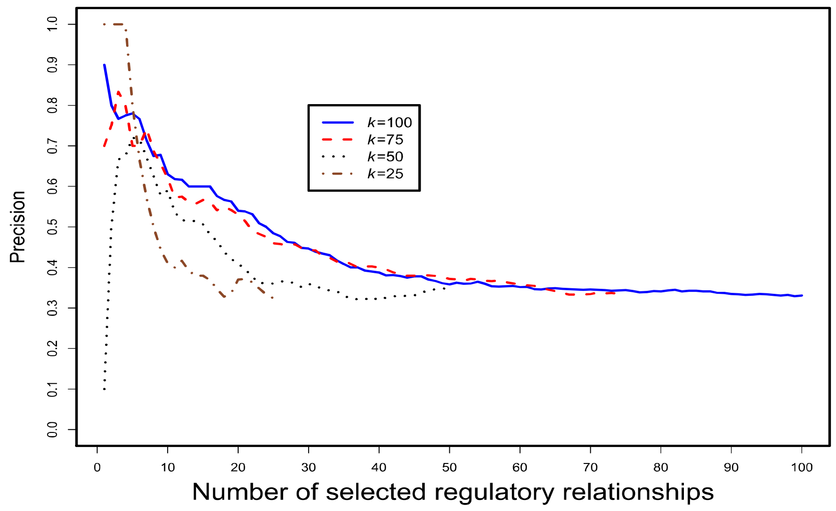

Figure 28: Comparison of precision given by the rank aggregation at k = 25, 50, 75, and 100.

$k = 75$ and $k = 100$ are about the same. This indicates that after k reaches 75, considering more interactions from the ordered lists of the candidate algorithms can no longer improve the performance of the rank aggregation method. Hence, k is set to 100 in our tests using the yeast dataset.

6.3.3 Comparison of integration methods and individual algorithms

In this subsection, we compare the performance of integration methods with individual algorithms. In order to make the comparison clear, for a given i ($i = 1, 2, ..., 100$), we define the *baseline* precision as that obtained by selecting the maximum of the precisions of the top i interactions given by all individual algorithms. That is, given a set of individual learning algorithms M, it is determined as:

$$p^*(i) = \max_{m \in M} \left(\mathrm{p}_m(i)\right), \tag{20}$$

where $\mathrm{p}_m(i)$ denotes the precision of top i interactions in the ordered list given by algorithm m (Equation 19). Note that baseline precisions represent an upper optimistic bound that can not be achieved by individual algorithms as we can only use one of them at a time. Hence, even if the precisions obtained by an integration method are only comparable to baseline precisions, it still shows that this integration method yield a better overall performance than those of individual algorithms.

We compare the precision of the top 100 predictions from the three integration methods with baseline precisions (Figure 29). The union and rank aggregation methods generate better or similar results compared to the baseline precisions. In addition, somewhat surprisingly, the union method, which has a lower computational cost than rank aggregation, archives comparable results as given by rank aggregation. The first twenty interactions from union and rank aggregation are shown in Tables 7 and 8, respectively.

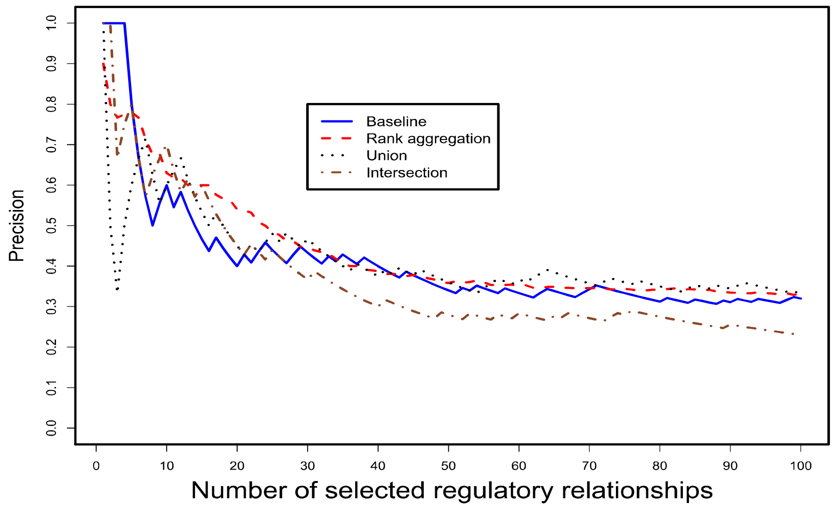

Figure 29: Comparison between precisions of integration methods and baseline precisions. For each of the three integration methods, the figure shows the precision (Equation 19) when the top i ($i = 1, 2, ..., 100$) regulator-module interactions in the rank given by the integration method are selected. For a given i, the baseline precision denotes the maximum of precisions obtained by individual learning algorithms (Equation 20).

On the other hand, we observe that baseline precisions are generally better than precisions given by the intersection method. The intersection method sorts interactions by their lowest rankings from all candidate algorithms, so it tends to assign interactions with moderate confidences from all algorithms with high ranks. We speculate that this may have affected its performance. For example, as shown in Table 9, its first 20 interactions include several that are not highly ranked by any algorithm, such as the twelfth interaction (RDS2 to module 16) ranked 219th, 125th and 273rd by the LIMMA-based method, LeMoNe, and Inferelator, respectively; and the eighteenth interaction (DAL81 to module 58) ranked as

			Ranks from individual algorithms		
rank	Regulator-module	p-value	LIMMA	LeMoNe	Inferelator
1	**DAL80-11***	3.47e-10	1	130	88
	IME4-46	1.00e+00	48	1	2906
	HAP1-13	1.00e+00	3076	169	1
4	**HAP4-7***	1.67e-30	50	9	2
	MET32-11*	1.21e-13	2	30	7
	DAL80-51*	0.00e+00	4	2	10281
7	**PHD1-36***	5.20e-03	3	342	5
	HAP4-30	5.33e-02	23	3	32
	MET32-27	1.00e+00	9680	3702	3
10	**TOS8-24***	1.45e-02	49	4	111
	GAT1-59	2.55e-03	802	5059	4
12	**XBP1-10***	1.08e-02	59	5	602
	DAL82-48	1.00e+00	5	49	7771
14	UGA3-11	2.35e-01	6	509	59
	USV1-28	1.00e+00	190	6	10281
	SKO1-57	1.00e+00	4663	10026	6
17	**GAT1-11***	4.24e-05	34	7	28
	ACA1-48	1.00e+00	7	1048	7773
19	PDR3-13	3.90e-01	12	8	10281
	DAL80-40	1.00e+00	8	367	27

Table 7: Top twenty regulator-module interactions as given by the union method. * records represent true positives at the p-value threshold of 0.05.

			Ranks from individual algorithms		
rank	Regulator-module	p-value	LIMMA	LeMoNe	Inferelator
1	**HAP4-7***	1.67e-30	50	9	2
2	**GAT1-11***	4.24e-05	34	7	28
3	**MET28-11***	5.92e-10	27	24	16
4	HAP4-30	5.33e-02	23	3	32
5	**MET32-11***	1.21e-13	2	30	7
6	**DAL80-51**	0.00e+00	4	2	10281
7	DAL81-6	6.67e-01	21	14	1193
8	PDR3-13	3.90e-01	12	8	10281
9	**PHD1-36***	5.20e-03	3	342	5
10	IME4-46	1.00e+00	48	1	2906
11	**TOS8-24***	1.45e-02	49	4	111
12	**GAL80-41***	7.45e-09	7107	10	52
13	DAL81-55	4.17e-01	14	16	285
14	**MET32-40***	1.00e-02	52	5329	37
15	**CUP2-10***	1.12e-02	553	32	18
16	YAP5-51	1.00e+00	37	746	40
17	**XBP1-10***	1.08e-02	59	5	602
18	YAP6-25	8.63e-02	26	26	5444
19	UGA3-40	1.00e+00	22	33	93
20	UGA3-11	2.35e-01	6	509	59

Table 8: Top twenty regulator-module interactions as given by the weighted rank aggregation method. * records represent true positives at the p-value threshold of 0.05.

			Ranks from individual algorithms		
rank	Regulator-module	p-value	LIMMA	LeMoNe	Inferelator
1	**MET28-11***	5.92e-10	27	24	16
2	**MET32-11***	1.21e-13	2	30	7
3	HAP4-30	5.33e-02	23	3	32
4	**GAT1-11***	4.24e-05	34	7	28
5	**HAP4-7***	1.67e-30	50	9	2
6	OAF1-22	5.37e-02	73	45	87
7	UGA3-40	1.00e+00	22	33	93
8	**TOS8-24***	1.45e-02	49	4	111
9	**DAL80-11***	3.47e-10	1	130	88
10	**GAL4-22***	7.44e-03	155	85	115
11	MET32-51	1.00e+00	28	224	113
12	RDS2-16	1.00e+00	219	125	273
13	**YRR1-24***	4.06e-04	273	21	230
14	DAL81-55	4.17e-01	14	16	285
15	**GZF3-11***	8.34e-04	31	188	297
16	TOS8-49	5.55e-02	87	308	85
17	SWI4-55	3.67e-01	39	63	310
18	DAL81-58	1.00e+00	199	310	190
19	BAS1-63	1.00e+00	33	88	312
20	IME4-3	4.52e-01	317	58	245

Table 9: Top twenty regulator-module interactions as given by the intersection method. * records represent true positives at the p-value threshold of 0.05.

	Integration methods			Individual algorithms		
	Rank aggregation	Union	Intersection	LIMMA	Inferelator	LeMoNe
Area under curve	42.63	41.12	36.09	36.72	35.79	35.18

Table 10: Comparison of areas under precision curves for the top 100 predictions given by the integration methods and individual learning algorithms. The precision curves for the integration methods are shown in Figure 29, while the precision curves for the individual learning algorithms are shown in Figure 27.

199th, 310th and 190th by the LIMMA-based method, LeMoNe, and Inferelator, respectively.

We also compare areas under precision curves for the top 100 predictions given by the integration methods and individual learning algorithms (Table 10). The union and weighted rank aggregation methods achieve better results than those from the individual learning algorithms, but the intersection method only yields a comparable result with the individual learning algorithms. These results indicate that we should be cautious to apply the intersection method to integrate results from algorithms of different natures.

6.4 Conclusion and future work

In this chapter, a meta-learner approach was applied to infer transcription factors of co-expressed gene modules in a yeast stress dataset, with the regulatory relationships recorded in YEASTRACT as the gold standard. We integrated the predictions of three existing inference techniques [62, 11, 100] by three different methods: union, intersection and weighted rank aggregation. Experimental results show that integrated predictions based on union or rank aggregation have higher precision than any of the individual methods. The justification of this work is that the results generated by different algorithms are not identical and often have clearly different influences from the datasets used. The experiments confirm our expectation that integrating the output of several algorithms results in higher quality predictions. To the best of our knowledge, this is the first such an attempt. Consequently, this work may point out a promising direction for module network learning.

An interesting extension of this work is to investigate if integrating results from more algorithms can lead to even better performance. In particular, we expect that when more algorithms are combined, we may see significant difference between the union and weighted rank aggregation methods.

The experiments in this work are conducted on a yeast dataset, and results are validated by the regulatory relationships recorded in YEASTRACT, which does not represent a complete reference database of the regulatory network in the yeast. Hence, another direction for future work is to perform experiments on expression data from other species (e.g., *E. coli* [35]) to verify if results are consistent with those we obtained in the yeast dataset. In addition, we are interested in performing experiments on synthetic datasets (e.g., DREAM [97]), where complete reference networks are available.

Computational time

It takes around 5 minutes for the union and intersection integration methods to generate the rank of regulator-module interactions on the yeast stress dataset on a Dell workstation with Intel Pentium 4 processor and 3GB memory. The weighted rank aggregation method requires much more computational resource than union and intersection, so we ran it on a HP rackmount server with AMD Opteron processors (x86, 64 bit, dual core) and 16 GB memory. It takes around 48 hours for the integration method on the yeast stress dataset.

Chapter 7

Inferring regulatory relationships in fungal species by module networks

Aspergillus niger, *Sporotrichum thermophile*, and *Phanerochaete chrysosporium* are important organisms for industry, because they are capable of producing many useful enzymes. The genomes of the species have been sequenced, but little of their regulatory networks have been identified. In this chapter, we apply module networks to infer gene clusters and their regulators in the three fungal species. The experimental results may help biologists to understand how their regulatory networks work. In addition, some results may be validated by wet lab experiments in the Centre for Structural and Functional Genomics at Concordia University.

7.1 Results in *Aspergillus niger*

Aspergillus niger (A. niger) is an important organism used in biotechnology, as a host to produce enzymes for many industries, such as food, beverage, textile, and agriculture [26]. The genome of *A. niger* has been sequenced and annotated [93]. In this section, we apply the module network method [106] to infer regulatory relationships in this fungus.

7.1.1 Experimental results and Discussion

7.1.1.1 Regulatory modules and their validation

We applied the module network method implemented in Genomica [106] to infer regulatory relationships within 229 differentially expressed sequence tags (ESTs) [109] under the conditions of xylose, maltose and glycerol. The dataset consists of 3 experimental conditions, so the initial number of modules and the maximum number of regulators for each module were decided to be 26 and 2. The default values were applied for the other parameters in Genomica.

After merging modules, Genomica learned 15 gene modules. Table 11 shows details of the 15 modules. Column Regulator1 denotes the regulator assigned to the root node of the regulatory tree of each module, while column Regulator2 denotes the regulator assigned to the internal node in the second level of the regulatory tree. However, in some modules (e.g., module 1), regulatory trees only consist of one internal node.

In order to evaluate the learned modules, we analyzed the GO annotation of ESTs in each module using BiNGO [76]. The latest annotation of *A. niger*, Joint Genome Institute (JGI) version 3.0 [29], was released in June 2008 [93]. 104 of 229 differentially expressed ESTs were annotated with GO terms in this release. We only considered GO terms annotating more than 3 genes and evaluated modules with more than 3 GO annotated genes. Consequently, modules 2, 7 , 10 and 11 were discarded. Under the p-value threshold of 0.05, no module represented significant functional enrichments. Hence, each module was named by the GO term with the smallest p-value that genes in the module were obtained in the GO enrichment analysis. For example, module 0 is named as carbohydrate transport module, because carbohydrate transport is the most significantly enriched GO term in the module (p-value = 0.14). One third (4 genes) of annotated genes in this module are annotated with this GO term.

Furthermore, we applied Gibbs Motif Sampler [124] to detect binding sites of transcription factors with lengths from 6 to 10 base pairs and maximum posteriori values more than 0 in the upstream region of ESTs in each module. The motif, TTCTTC, was identified from module 3. Table 16 in Appendix lists the module assignment of 229 ESTs.

No.	Module name	#Genes (#genes with GO annotation)	p-value (GO coherence)	Regulator1	Regulator2	Motif detected
0	carbohydrate transport	17 (12)	14% (33%)	Asn_07474	Asn_02714	
1	metabolic process	17 (8)	13% (100%)	Asn_04815		
2	N/A	2 (0)	N/A (N/A)	Asn_04815		
3	electron transport	16 (10)	52% (20%)	Asn_02714		TTCTTC
4	cellular biosynthetic process	14 (11)	21% (18%)	Asn_02714	Asn_07474	
5	alcohol metabolic process	26 (7)	41% (28%)	Asn_02714	Asn_07474	
6	cellular protein metabolic process	17 (5)	29% (40%)	Asn_02714		
7	N/A	8 (0)	N/A (N/A)	Asn_02714		
8	carbohydrate metabolic process	20 (9)	32% (44%)	Asn_04815		
9	biosynthetic process	14 (8)	35% (38%)	Asn_04815		
10	N/A	2 (1)	N/A (N/A)	Asn_02714	Asn_07474	
11	N/A	7(1)	N/A(N/A)	Asn_04815		
12	macromolecule metabolic process	39 (15)	53% (40%)	Asn_04815		
13	carbohydrate transport	17 (9)	68% (33%)	Asn_07474	Asn_04815	
14	regulation of macromolecule metabolic process	13 (6)	19% (33%)	None		

Table 11: Summary of inferred modules in *A. niger*. The GO coherence of each module is measured as the percentage of genes annotated by the GO term assigned as the name of the module. The columns Regulator1 and Regulator2 denote the regulators assigned to the root node and the internal node in the second level of the regulatory tree of each module, respectively.

7.1.1.2 Discussion

The gene expression data were collected under three different sugar conditions, so we expected to identify transcription regulatory relationships between genes involved in sugar metabolism. The inferred modules match our expectation, since they participate in various processes in metabolism, such as carbohydrate transport and carbohydrate metabolic process. However, although most modules are large enough to show functional enrichment, we do not find any module with functional enrichment below the p-value threshold of 0.05. Moreover, the GO terms with the lowest p-values of these modules are very general GO terms. For example, the most specific terms are "electron transport" (module 3) and "regulation of macromolecule metabolic process" (module 14), which are at the fifth layer in the biological process ontology. This problem is partially due to the fact that only a small fraction of ESTs (104 of 229) are annotated with GO terms and most annotated terms are very general. When new releases of *A. niger* annotations are available, we expect this situation to improve.

In this experiment, we used 10 genes as the candidates of transcription factors. *creA* [103] and *xlnR* [53], two known transcription factors in *A. niger*, were not included, because they were not in the list of differentially expressed ESTs. Adding these two known transcription factors into the experiments would not change the results, because Genomica relies on the correlation between the expression levels of transcription factors and their regulated genes to detect regulatory relationships. Hence, if the expression levels of some candidate regulators do not change significantly, they will not be selected to explain the change of expression levels of genes in modules. This is also demonstrated by the result that the three ESTs, Asn_04815, Asn_07474 and Asn_02714, which are the most significantly differentially expressed among the 10 candidate regulators, regulate all learned modules. On the other hand, we speculate that the small number of experimental conditions covered by the dataset may lead to this result.

7.1.2 Methods

7.1.2.1 Collecting gene expression data

We collected the gene expression data for *A. niger* under the conditions of xylose, maltose and glycerol, using cDNA microarray. For each condition, we collected 6 samples, so the total number of arrays is 18. The probes on these arrays were retrieved from the expressed sequence tag database [109] in fungal genomics project in Concordia university. LIMMA [114] package was used to select differentially expressed ESTs. We set the cut-off p-value to 0.001, and 229 ESTs were selected under this threshold. These differentially expressed ESTs included several enzymes involved in xylose consumption [27], such as *xyrA* (probable NAD(P)H-dependent D-xylose reductas), *xlnD* (4-beta-D-xylan xylohydrolase) and *aguA* (alpha-D-glucuronoside).

7.1.2.2 Selecting candidate transcription factors

We downloaded all genes annotated with the GO term "transcription regulation" in *A. niger* from the UniProt database [70]. Then, we blasted the 229 differentially expressed ESTs against these transcription factors. 9 ESTs have hits with E-values less than 0.001. These differentially expressed ESTs were also blasted against transcription factors in the budding yeast obtained from SGD [20]. 5 ESTs have hits with E-values less than 0.001. Combining these two results from BLAST together, we obtained a set of 10 ESTs, which were used as candidate transcription factors in this experiment. Table 12 shows detail about these candidate transcription factors.

7.2 Results in *Sporotrichum thermophile*

Sporotrichum thermophile (S. thermophile) is a thermophilic fungus species. One of its most important characteristics is that it is capable of secreting enzymes that can efficiently decompose lignocellulose. These enzymes are critical for the development of technologies using biomass to generate energy. In this section, we apply the module network method [62] to infer regulatory relationships in this fungus based on expression data collected under varied carbon sources.

7.2.1 Experimental results and discussion

Using a Gibbs sampling-based clustering algorithm [63], we identified 13 modules consisting of 436 genes (Table 17 in Appendix). In order to evaluate the learned modules, we analyzed the InterPro protein domain annotations of genes in each module using BiNGO [76] (Table 13). Note that under the p-value threshold 0.05, the modules 4, 9, 10, 12, and 14

No.	EST	UniProt Hit	GO from UniProt	SGD Hit	Description in SGD
1	Asn_02714	A2QRX0	Regulation of transcription, DNA-dependent (GO:0006355)		
2	Asn_07607	A2RA34	Regulation of transcription, DNA-dependent (GO:0006355)	YBR240C	Zinc finger protein of the Zn(II)2Cys6 type, probable transcriptional activator of thiamine biosynthetic genes
3	Asn_03498	A2QF35			
4	Asn_04815	A2QUL7	1. Regulation of sequence-specific DNA binding transcription factor activity (GO:0051090); 2. Transcription initiation, DNA-dependent (GO:0006352)		
5	Asn_01382	A5AA99	Regulation of transcription, DNA-dependent (GO:0006355)		
6	Asn_07224	A2R5X0	Regulation of transcription, DNA-dependent (GO:0006355)		
7	Asn_04275	A2QGW8	Regulation of transcription, DNA-dependent (GO:0006355)	YPL248C	DNA-binding transcription factor required for the activation of the GAL genes in response to galactose; repressed by Gal80p and activated by Gal3p
8	Asn_00534	A2QZ26		YKL062W	Transcriptional activator related to Msn2p; activated in stress conditions, which results in translocation from the cytoplasm to the nucleus; binds DNA at stress response elements of responsive genes, inducing gene expression
9	Asn_04368	A2Q7Y4	Regulation of transcription, DNA-dependent (GO:0006355)		
10	Asn_07474			YIR023W	Positive regulator of genes in multiple nitrogen degradation pathways; contains DNA binding domain but does not appear to bind the dodecanucleotide sequence present in the promoter region of many genes involved in allantoin catabolism

Table 12: Candidate transcription factors in *A. niger*. The column "UniPro Hit" denotes the hit of each EST against the UniProt database, while the next column shows the GO annotation of the hit in the UniProt database. The column "SGD Hit" denotes the hit of each EST against transcription factors in the budding yeast, while the next column shows the description of the hit in the SGD database.

Module No	#Genes in the module	Enriched InterPro domain	#Genes annotated with the enriched domain	p-value	Inferred regulator
1	73	IPR011332 Ribosomal protein, zinc-binding domain	4	1.1043e-07	N/A
2	36	IPR006424 Glyceraldehyde-3-phosphate dehydrogenase, type I	1	0.029434	Spoth1|86826
3	56	IPR007125 Histone core	3	0.0015421	Spoth1|44208
4	30	N/A	N/A	N/A	N/A
5	20	IPR011050 Pectin lyase fold/virulence factor	6	4.5571e-11	Spoth1|83795
7	17	IPR001547 Glycoside hydrolase, family 5	2	0.0055073	Spoth1|111567
8	24	IPR011583 Chitinase II	2	0.0022806	Spoth1|110916
9	58	N/A	N/A	N/A	N/A
10	11	N/A	N/A	N/A	Spoth1|103539
12	32	N/A	N/A	N/A	N/A
13	31	IPR002068 Heat shock protein Hsp20	2	0.0018607	Spoth1|111817
14	28	N/A	N/A	N/A	Spoth1|114828
15	20	IPR005103 Glycoside hydrolase, family 61	5	4.6656e-08	Spoth1|80133

Table 13: Summary of inferred modules in *S. thermophile*

do not show enrichment for any annotation. We infer the regulators of each module using LeMoNe [62] (Table 13). Note that LeMoNe does not obtain results with high confidence for the modules 1, 4, 9, and 12.

This dataset consists of expression values measured under 10 carbon sources. Consequently, several identified gene modules are related to biological processes for utilizing particular carbon sources in metabolism. The module 5 consists of 20 genes, and 6 of them are associated the InterPro annotation “PR011050”. This term denotes pectin lyases which are capable of degrading the pectic components of the plant cell wall [84]. In addition, most genes in this module show expression profiles consistent with the annotation. That is, they are highly expressed under pectin (Figure 30). Hence, we expect that genes with

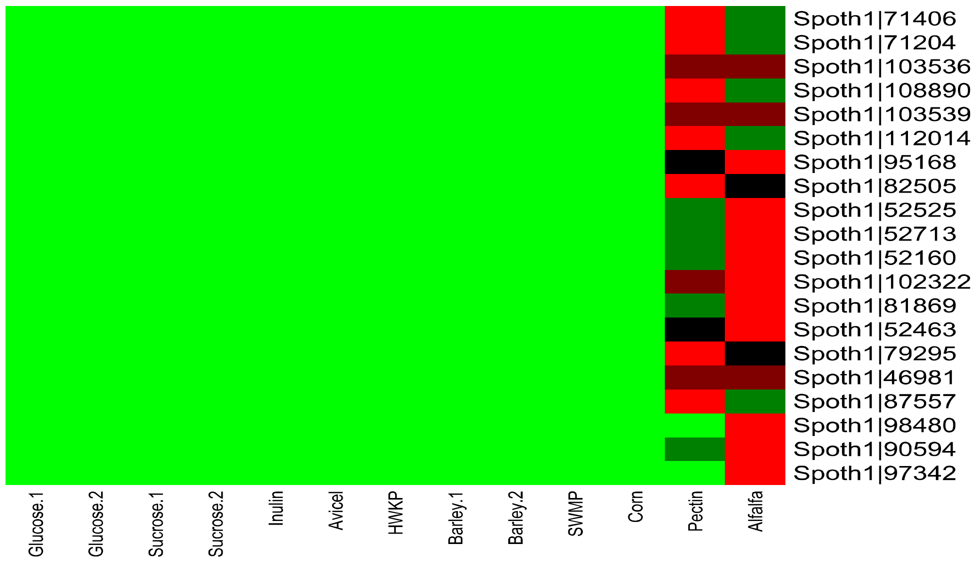

Figure 30: Heatmap of expression values of genes in module 5 in the *S. thermophile* dataset.

unknown functions in the module are very likely to also participate in degrading pectin. This hypothesis is worth of validating by lab experiments.

The most enriched annotations for the modules 7 and 15 are "PR001547" (glycoside hydrolase family 5) and "PR005103" (glycoside hydrolase family 61), respectively. Both terms denote enzymes that hydrolyse the glycosidic bond between two or more carbohydrates, or between a carbohydrate and a non-carbohydrate moiety. However, the family 61 only consists of endoglucanase enzymes, while the family 5 covers enzymes with a wide range of activities, such as xylanase, endoglucanase, and endoglycoceramidase. Accordingly, the expression profiles of genes in these two modules are different. Genes in the module 7 are strongly expressed under softwood mechanical pulp and slightly expressed under barley (Figure 31), while genes in the module 15 are strongly expressed under barley and slightly expressed under softwood mechanical pulp (Figure 32). Web lab experiments may be performed to identity the biological function of genes in the two modules.

7.2.2 Methods

7.2.2.1 Collecting gene expression data

RNA-seq technology was used to measure the expression values of 8,486 genes under 10 carbon sources: glucose, sucrose, inulin, pectin, avicel, hardwood kraft pulp (HWKP),

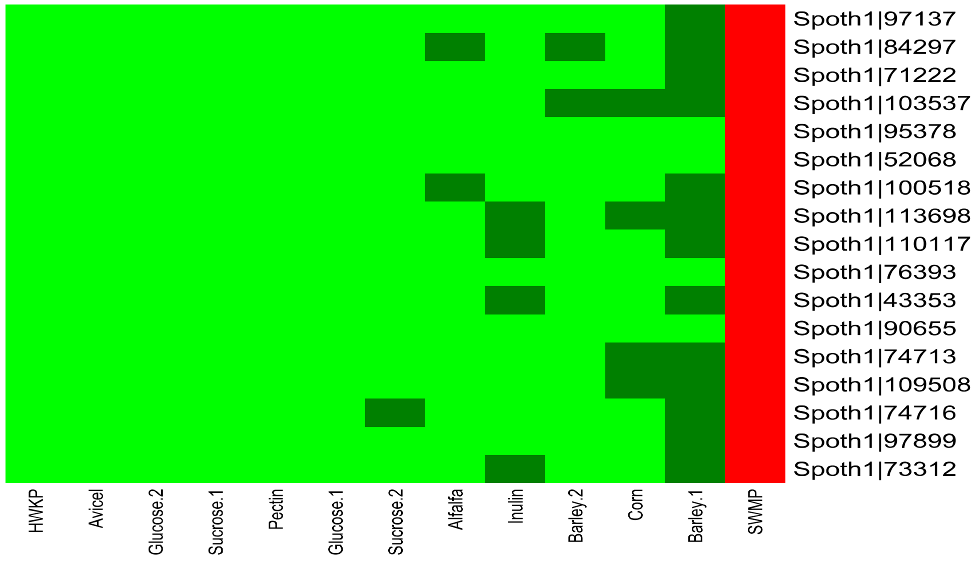

Figure 31: Heatmap of expression values of genes in module 7 in the *S. thermophile* dataset.

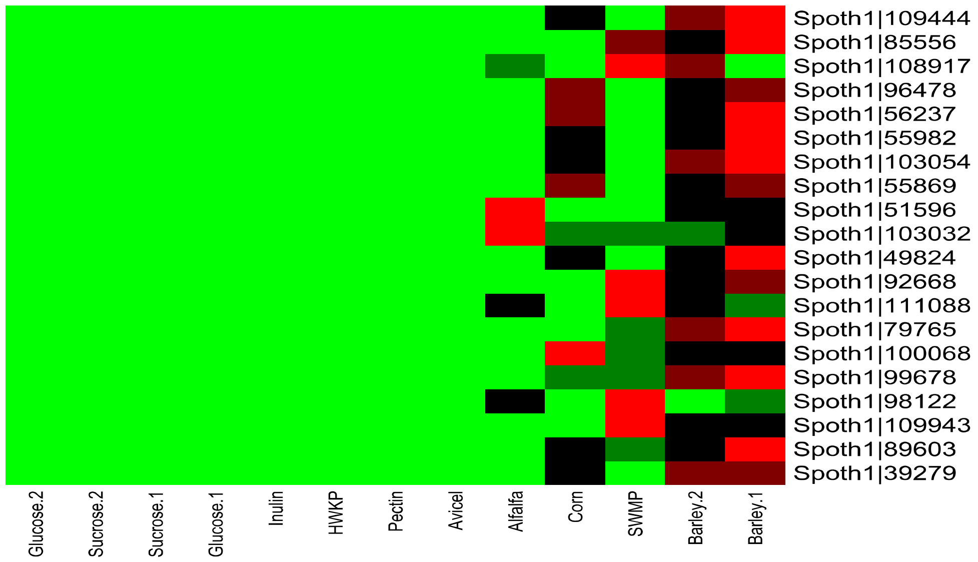

Figure 32: Heatmap of expression values of genes in module 15 in the *S. thermophile* dataset.

softwood mechanical pulp (SWMP), corn, alfalfa, and barley. We collected two samples under glucose, sucrose, and barley, respectively, while we collected one sample under each of the other carbon sources. Hence, this dataset consists of 13 samples. Cufflinks [128] was applied to convert raw counts into FPKM values. 5,853 out of 8,468 genes are associated with protein domain annotation in the JGI *S. thermophile* website [30]. The annotation was used in the gene set enrichment analysis, because it has a better quality than GO annotation in the species.

We removed genes that are not significantly differentially expressed in this dataset (standard deviation less than 50). This led to a dataset with only 911 genes. 699 out of the 911 genes are annotated with protein domain annotation. The expression values of each gene was normalized by subtracting its mean and dividing by its standard deviation.

7.2.2.2 Applying LeMoNe to infer regulatory relationships

We applied a Gibbs sampling clustering algorithm [63] to infer gene clusters (modules) in this dataset. 50 Gibbs samplers were sampled with the default parameters. Since the Gibbs sampling is a non-deterministic algorithm, we calculated the correlation between two runs with 50 independent samplers, which was 0.99, a value indicating that the Gibbs sampling procedure reached convergence.

The clustering algorithm obtained 24 modules. We discarded modules with more than 100 genes or less than 5 genes, because they were not likely to represent biologically coherent modules. Consequently, 13 modules consisting of 436 genes were kept for further analysis. Then, we applied LeMoNe [62], which is based on logistic regression and has been described in Section 3.3.3, to infer regulators of these modules. For each module, 10 condition clusterings were sampled with default parameters. In addition, a list of 870 candidate transcription factors were downloaded from the JGI *S. thermophile* website [30] by searching the keyword "Transcription factor".

7.3 Results in *Phanerochaete chrysosporium*

Phanerochaete chrysosporium is a white rot fungus, which is able to efficiently degrade lignin in wood to carbon dioxide and thereby gain access to the carbohydrate polymers of plant cell walls [82]. Moreover, the fungus is capable of degrading explosive contaminants,

Module	Number of genes in this module	Inferred regulator
module 1	85	Phchr1|03235
module 2	48	Phchr1|00149
module 3	45	Phchr1|03235
module 4	45	Phchr1|00419
module 5	51	Phchr1|03235
module 6	31	Phchr1|03235
module 7	52	Phchr1|05523
module 8	27	Phchr1|06278
module 9	75	Phchr1|03235
module 10	16	Phchr1|06278
module 11	18	Phchr1|11322
module 12	16	Phchr1|00149
module 13	13	Phchr1|05523
module 14	10	Phchr1|01493
module 15	97	Phchr1|03235
module 17	20	Phchr1|00419
module 18	42	Phchr1|03235
module 19	23	Phchr1|03235

Table 14: Summary of inferred modules and their transcription factors in the *Phanerochaete chrysosporium* dataset

pesticides and toxic waste. In this section, we apply the LIMMA-based method (described in Chapter 5) to infer regulatory relationships in this fungus based on expression data collected under different levels of nitrogen.

7.3.1 Experimental results and discussion

We identified 18 modules consisting of 714 genes (Table 18 in Appendix). Table 14 shows the inferred transcription factor of each module. Most inferred modules are differentially expressed under different levels of nitrogen, such as module 4 and module 7. Genes in module 4 are highly expressed under the high nitrogen condition (Figure 33), while genes in module 7 are highly expressed under the low nitrogen condition (Figure 34). Genes in these two modules are more expressed after 4 days than after 2 days. Hence, we speculate that the genes may be involved in the response to the change of nitrogen levels in the environment. Our hypothesis can be further validated by lab experiments. Tables 19 and 20 in Appendix show their annotation.

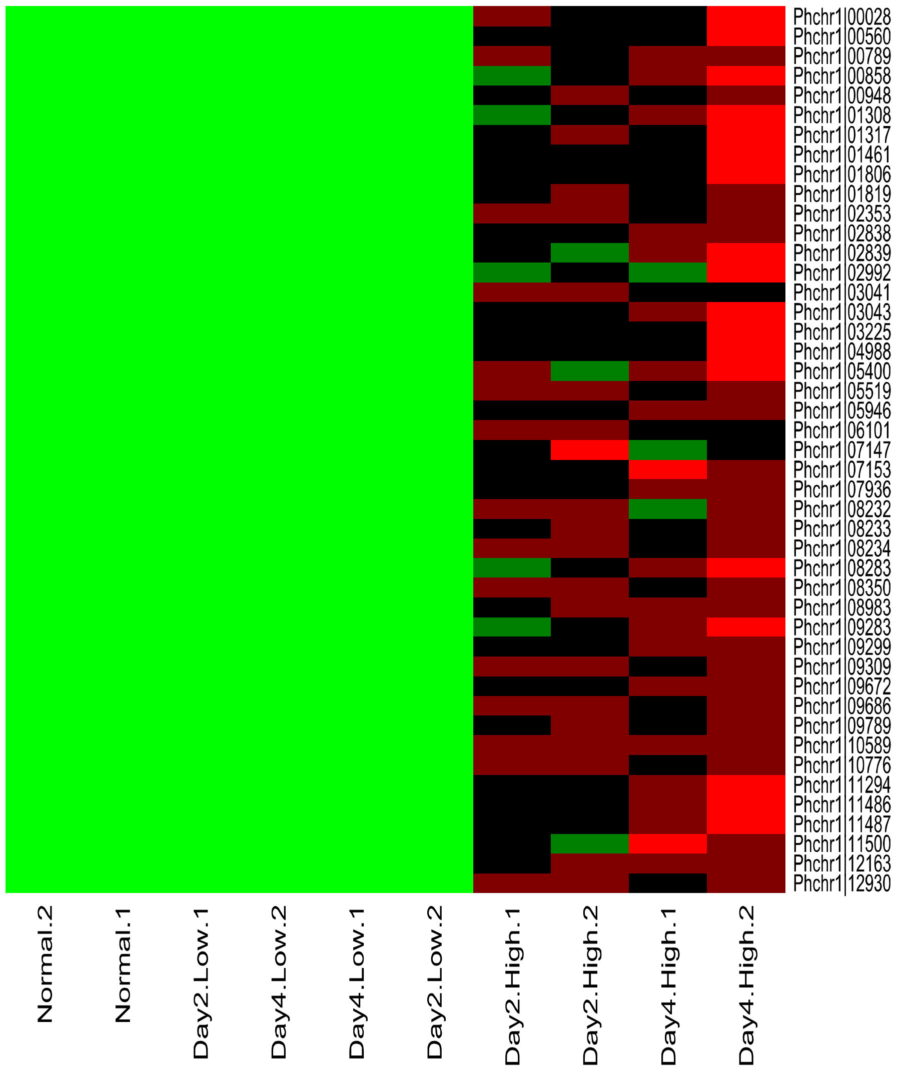

Figure 33: Heatmap of expression values of genes in module 4 in the *Phanerochaete chrysosporium* dataset. Day2.High and Day4.High denote samples collected after 2 days and 4 days under the high nitrogen condition, respectively. Day2.Low and Day4.Low denote samples collected after 2 days and 4 days under the low nitrogen condition, respectively.

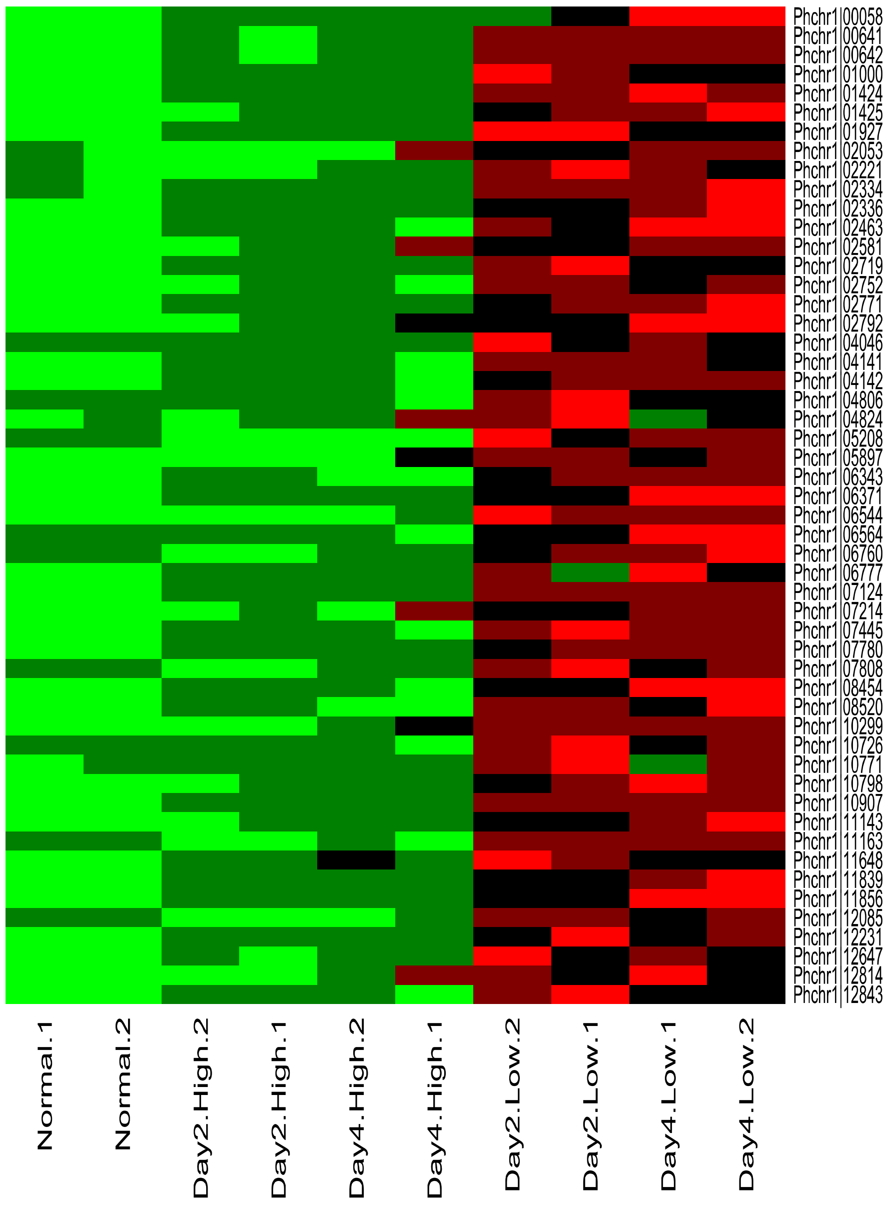

Figure 34: Heatmap of expression values of genes in module 7 in the *Phanerochaete chrysosporium* dataset. Day2.High and Day4.High denote samples collected after 2 days and 4 days under the high nitrogen condition, respectively. Day2.Low and Day4.Low denote samples collected after 2 days and 4 days under the low nitrogen condition, respectively.

Another interesting module is module 9. Genes in the module are highly expressed under the normal condition, but not expressed under either the high nitrogen condition or the low nitrogen condition (Figure 35). We speculate that the genes may be involved in the biological process of utilizing nitrogen under the normal condition. Table 21 in Appendix shows their annotation.

7.3.2 Methods

7.3.2.1 Collecting gene expression data and candidate transcription factors

RNA-seq technology was used to measure the expression values of 12,998 genes in *Phanerochaete chrysosporium*. Under each of high nitrogen and low nitrogen conditions, the expression values of the genes were measured after 2 days and 4 days, respectively. In addition, their expression values under the normal condition were also measured. We collected two replicates of each measurement, so the dataset consisted of 10 samples. Cufflinks [128] was applied to convert raw counts into FPKM values. The annotation of the genes was downloaded from the website of the centre for structural and functional genomics at Concordia university. The annotation is based on blasting the genes against well-annotated proteins from other fungi in the SwissProt database. 5,503 out of 12,998 genes have hits under the E-value threshold 1E-9.

We removed genes with standard deviations less than 100. This led to a dataset with only 861 genes. 498 out of the 861 genes are associated with annotation. The expression values of each gene was normalized by subtracting its mean and dividing by its standard deviation.

7.3.2.2 Applying the LIMMA-based method to infer regulatory relationships

We applied the Gibbs sampling clustering algorithm [63] to infer gene clusters (modules) in this dataset. 100 Gibbs samplers were sampled with the default parameters. The clustering algorithm obtained 23 modules. Modules with more than 100 genes or less than 5 genes were discarded. Consequently 18 modules were kept. We applied the LIMMA-based method (described in Chapter 5) to infer regulators of the modules. 44 genes were selected as candidate transcription factors according to their annotations (Table 15).

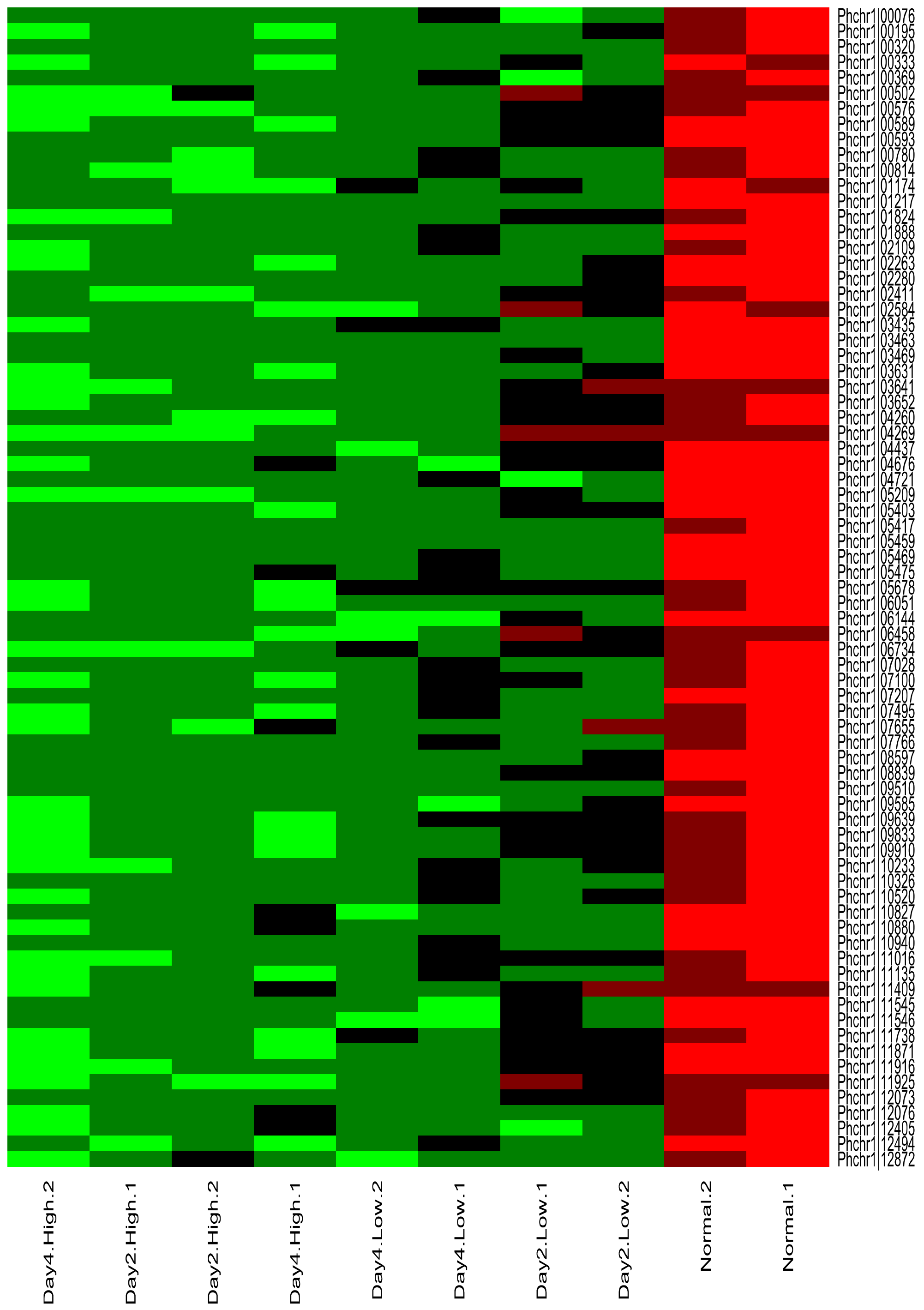

Figure 35: Heatmap of expression values of genes in module 9 in the *Phanerochaete chrysosporium* dataset. Day2.High and Day4.High denote samples collected after 2 days and 4 days under the high nitrogen condition, respectively. Day2.Low and Day4.Low denote samples collected after 2 days and 4 days under the low nitrogen condition, respectively.

No	Transcription factor	Annotation
1	Phchr1|00149	Transcription initiation factor TFIID subunit 7
2	Phchr1|00276	RNA polymerase II transcription factor B subunit 3
3	Phchr1|00412	Transcription elongation factor 1
4	Phchr1|00419	Transcription initiation factor TFIID subunit 13
5	Phchr1|00600	Transcription factor SPT8
6	Phchr1|00602	Transcription elongation factor spt-6
7	Phchr1|01409	Transcription initiation factor TFIID subunit 11
8	Phchr1|01493	Transcription elongation factor S-II
9	Phchr1|01730	Transcription factor SOX-30
10	Phchr1|02172	Transcription initiation factor TFIID subunit 5
11	Phchr1|02491	Transcription factor tau subunit sfc4
12	Phchr1|03145	Putative transcription factor C16C4.22
13	Phchr1|03186	Transcription initiation factor TFIID subunit 2
14	Phchr1|03235	pH-response transcription factor pacC/RIM101
15	Phchr1|03236	Putative transcription initiation factor TFIID 111 kDa subunit
16	Phchr1|03300	Transcription factor TFIIIB component B" homolog
17	Phchr1|03705	AP-1-like transcription factor
18	Phchr1|04286	Transcription factor IWS1
19	Phchr1|05320	Transcription initiation factor TFIID subunit 9
20	Phchr1|05421	Transcription elongation factor SPT4
21	Phchr1|05470	Transcription factor atf1
22	Phchr1|05523	General transcription factor 3C polypeptide 5
23	Phchr1|05796	RNA polymerase I-specific transcription initiation factor rrn3
24	Phchr1|06013	Transcription initiation factor IIB
25	Phchr1|06216	Transcription factor 25
26	Phchr1|06278	Nuclear transcription factor Y subunit gamma
27	Phchr1|06800	Transcription factor IIIB 60 kDa subunit
28	Phchr1|07516	Transcription initiation factor TFIID subunit 10b
29	Phchr1|07651	Transcription factor bHLH140
30	Phchr1|08302	Transcription initiation factor IIF subunit beta
31	Phchr1|09400	Probable transcription initiation factor IIA small chain
32	Phchr1|09418	Transcription elongation factor SPT5
33	Phchr1|09734	Nuclear transcription factor Y subunit beta
34	Phchr1|09896	Transcription factor prr1
35	Phchr1|09897	Transcription factor prr1
36	Phchr1|10102	Transcription initiation factor IIA large subunit
37	Phchr1|10195	Transcription initiation factor TFIID subunit 12
38	Phchr1|10350	RNA polymerase II transcription factor B subunit 4
39	Phchr1|10425	General transcription factor IIH subunit 1
40	Phchr1|10698	TFIIH basal transcription factor complex p47 subunit
41	Phchr1|10844	RNA polymerase II transcription factor B subunit 2
42	Phchr1|11322	General transcription factor IIE subunit 1
43	Phchr1|11863	Transcription initiation factor TFIID subunit 6
44	Phchr1|12375	Transcription factor steA

Table 15: Candidate transcription factors in *Phanerochaete chrysosporium*

Computational time

It takes around 30 minutes for Genomica on the *A. niger* dataset on a Dell workstation with Intel Pentium 4 processor and 3GB memory. It takes around 1 minute for LeMoNe to generate one Gibbs sampler for clustering genes on the *S. thermophile* dataset and the *Phanerochaete chrysosporium* dataset on a HP rackmount server with AMD Opteron processors (x86, 64 bit, dual core) and 16 GB memory. 100 samplers were generated for each dataset. It takes around 5 minutes for LeMoNe to infer transcription factors of gene modules on the *S. thermophile* dataset on the Dell workstation. It takes 2 minutes for the linear model to infer transcription factors of gene modules on the *Phanerochaete chrysosporium* dataset on the Dell workstation.

Chapter 8

Conclusions and future work

Cells have a complex mechanism that controls the expression of genes so that they are able to express varied combinations of genes in response to environmental changes or genetic perturbations. A major part of this mechanism is fulfilled by transcription factors, which are able to bind to the upstream region of genes and then influence their expression levels. The transcriptional regulatory relationships between genes and their transcription factors can be represented by a network, called a transcription regulatory network.

Gene expression data are widely used to infer transcription regulatory networks. However, many transcription factors and their targets do not show correlated expression profiles, because the activity of the transcription factors may be regulated by post-translational modifications or by protein-protein interactions. Hence, we can not rely on expression data to reconstruct the entire regulatory network.

Many algorithms have been proposed for inferring transcription regulatory networks from gene expression data. Particularly, module networks [105], which are a special type of Bayesian networks, are shown to be promising. In a module network, a regulatory module is a set of genes that show similar expression profiles and are regulated by a shared set of regulators (i.e., the regulation program of the module). The regulation program of a module is a set of transcription factors that regulate the transcription of the genes in the module. This thesis concentrates on designing algorithms for inferring the regulation program of a given module.

First, we proposed a regression tree-based Gibbs sampling algorithm (Chapter 4). We show that in synthetic datasets the proposed sampling algorithm achieves better results than the deterministic algorithm [106]. Moreover, most predictions made by the sampling algorithm in the yeast stress dataset [45] are supported by known regulatory relationships in YEASTRACT [88]. Second, we proposed to apply a linear model to infer regulators in module networks (Chapter 5). The effectiveness of the method was demonstrated by the experiments in the yeast stress dataset and the *E. coli* dataset [35]. In addition, we showed that the proposed method can detect regulatory relationships where transcription factors and their targets are locally co-expressed. The regulatory relationships are often neglected by regression tree-based algorithms. Third, we proposed to integrate results from different regulation program learning algorithms (Chapter 6). For this, we combined results given by three algorithms: LeMoNe [62], the LIMMA-based method [100], and Inferelator [11] on the yeast stress dataset. The experiments show that the union and weighted rank aggregation integration methods produce better precisions than those obtained by individual algorithms.

Despite the success of the module network method, it has shortcomings. The most critical limitation is that not all significant regulatory relationships can be embedded in modules. The limitation is due to two reasons. First, some transcription factors regulate a very small number of genes or even just one gene. Consequently, their targets are not likely to be grouped into individual modules. As a result, the module network method shows a poor performance in detecting targets of the transcription factors no matter which regulation program learning algorithm is applied. Second, the result of the module network method is normally interpreted as that the inferred transcription factor of a module regulates all genes in the module. However, the interpretation is sometimes problematic and may lead to a large number of false positives. The genes involved in the same biological process (i.e., a module) may be regulated by different transcription factors, because they may play distinct roles in the process.

To overcome the limitation, a straightforward solution is to apply individual gene-based algorithms, such as CLR [35]. However, many condition contrasts are not obvious when genes are not organized into modules, and consequently individual gene-based algorithms fail to detect some regulatory relationships that module-based methods identify [86]. Most regulatory network learning algorithms only work with either modules or genes, but not both. The further work of this thesis is to perform analysis using module-based methods as well as individual gene-based methods to investigate way to select between them in a given dataset.

Bibliography

[1] `http://www.454.com/`. [Online; accessed 08-Jun-2010].

[2] `http://www.appliedbiosystems.com`. [Online; accessed 08-Jun-2010].

[3] `http://www.illumina.com`. [Online; accessed 08-Jun-2010].

[4] D. Abdulrehman, P. T. Monteiro, M. C. Teixeira, N. P. Mira, A. B. Loureno, S. C. dos Santos, T. R. Cabrito, A. P. Francisco, S. C. Madeira, R. S. Aires, A. L. Oliveira, I. S-Correia, and A. T. Freitas. YEASTRACT: providing a programmatic access to curated transcriptional regulatory associations in *Saccharomyces cerevisiae* through a web services interface. *Nucleic Acids Research*, 39(suppl 1):D136–D140, 2011.

[5] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[6] M. Bansal, V. Belcastro, A. Ambesi-Impiombato, and D. di Bernardo. How to infer gene networks from expression profiles. *Molecular Systems Biology*, 3:78–87, 2007.

[7] Y. Benjamini and D. Yekutieli. The control of the false discovery rate in multiple testing under dependency. *Annals of Statistics*, 29:1165–1188, 2001.

[8] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and D. L. Wheeler. GenBank. *Nucleic Acids Research*, 35(suppl_1):D21–25, 2007.

[9] P. J. Bhat and T. V. S. Murthy. Transcriptional control of the GAL/MEL regulon of yeast *Saccharomyces cerevisiae*: mechanism of galactose-mediated signal transduction. *Molecular Microbiology*, 40(5):1059–1066, 2001.

[10] G. Bindea, B. Mlecnik, H. Hackl, P. Charoentong, M. Tosolini, A. Kirilovsky, W.-H. Fridman, F. Pags, Z. Trajanoski, and J. Galon. ClueGO: a Cytoscape plug-in

to decipher functionally grouped gene ontology and pathway annotation networks. *Bioinformatics*, 25(8):1091–1093, 2009.

[11] R. Bonneau, D. J. Reiss, P. Shannon, M. Facciotti, L. Hood, N. S. Baliga, and V. Thorsson. The Inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets *de novo*. *Genome Biology*, 7(5):R36, 2006.

[12] M. R. Brent. How does eukaryotic gene prediction work. *Nature Biotechnology*, 25:883–885, 2007.

[13] M. J. Buck and J. D. Lieb. ChIP-chip: considerations for the design, analysis, and application of genome-wide chromatin immunoprecipitation experiments. *Genomics*, 83(3):349 – 360, 2004.

[14] J. Bullard, E. Purdom, K. Hansen, and S. Dudoit. Evaluation of statistical methods for normalization and differential expression in mRNA-Seq experiments. *BMC Bioinformatics*, 11(1):94, Feb. 2010.

[15] A. J. Butte, I. S. Kohane, and I. S. Kohane. Mutual information relevance networks: Functional genomic clustering using pairwise entropy measurements. *Pacific Symposium on Biocomputing*, 5:415–426, 2000.

[16] M. Carlson. Regulation of glucose utilization in yeast. *Current Opinion in Genetics & Development*, 8(5):560 – 564, 1998.

[17] M. Carlson. Glucose repression in yeast. *Current Opinion in Microbiology*, 2(2):202 – 207, 1999.

[18] L. E. Carvalho and C. E. Lawrence. Centroid estimation in discrete high-dimensional spaces with applications in biology. *Proceedings of the National Academy of Sciences*, 105(9):3209–3214, 2008.

[19] R. Caspi, T. Altman, J. M. Dale, K. Dreher, C. A. Fulcher, F. Gilham, P. Kaipa, A. S. Karthikeyan, A. Kothari, M. Krummenacker, M. Latendresse, L. A. Mueller, S. Paley, L. Popescu, A. Pujar, A. G. Shearer, P. Zhang, and P. D. Karp. The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of pathway/genome databases. *Nucleic Acids Research*, 38(suppl 1):D473–D479, 2010.

[20] J. Cherry, C. Adler, C. Ball, S. Chervitz, S. Dwight, E. Hester, Y. Jia, G. Juvik, T. Roe, M. Schroeder, S. Weng, and D. Botstein. SGD: *Saccharomyces* Genome Database. *Nucleic Acids Research*, 26(1):73–79, 1998.

[21] K. R. Christie, S. Weng, R. Balakrishnan, M. C. Costanzo, K. Dolinski, S. S. Dwight, S. R. Engel, B. Feierbach, D. G. Fisk, J. E. Hirschman, E. L. Hong, L. Issel-Tarver, R. Nash, A. Sethuraman, B. Starr, C. L. Theesfeld, R. Andrada, G. Binkley, Q. Dong, C. Lane, M. Schroeder, D. Botstein, and J. M. Cherry. *Saccharomyces* Genome Database (SGD) provides tools to identify and analyze sequences from *Saccharomyces cerevisiae* and related sequences from other organisms. *Nucleic Acids Research*, 32(suppl-1):D311–314, 2004.

[22] G. Cochrane, P. Aldebert, N. Althorpe, M. Andersson, W. Baker, A. Baldwin, K. Bates, S. Bhattacharyya, P. Browne, A. van den Broek, M. Castro, K. Duggan, R. Eberhardt, N. Faruque, J. Gamble, C. Kanz, T. Kulikova, C. Lee, R. Leinonen, Q. Lin, V. Lombard, R. Lopez, M. McHale, H. McWilliam, G. Mukherjee, F. Nardone, M. P. G. Pastor, S. Sobhany, P. Stoehr, K. Tzouvara, R. Vaughan, D. Wu, W. Zhu, and R. Apweiler. EMBL Nucleotide Sequence Database: developments in 2005. *Nucleic Acids Research*, 34(suppl_1):D10–15, 2006.

[23] J. A. Coffman, R. Rai, T. Cunningham, V. Svetlov, and T. G. Cooper. Gat1p, a GATA family protein whose production is sensitive to nitrogen catabolite repression, participates in transcriptional activation of nitrogen-catabolic genes in *Saccharomyces cerevisiae*. *Molecular and Cellular Biology*, 16(3):847–858, 1996.

[24] T. S. Cunningham, R. Rai, and T. G. Cooper. The level of DAL80 expression down-regulates GATA factor-mediated transcription in *Saccharomyces cerevisiae*. *Journal of Bacteriology*, 182(23):6584–6591, 2000.

[25] F. De Bona, S. Ossowski, K. Schneeberger, and G. Rtsch. Optimal spliced alignments of short sequence reads. *Bioinformatics*, 24(16):i174–i180, 2008.

[26] R. P. de Vries and J. Visser. *Aspergillus* enzymes involved in degradation of plant cell wall polysaccharides. *Microbiology and Molecular Biology Reviews*, 65(4):497–522, 2001.

[27] R. P. de Vries, J. Visser, and L. H. de Graaff. CreA modulates the XlnR-induced expression on xylose of *Aspergillus niger* genes involved in xylan degradation. *Research in Microbiology*, 150(4):281 – 285, 1999.

[28] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[29] DOE Joint Genome Institute. Joint Genome Institute *Aspergillus niger*. `http://genome.jgi-psf.org/Aspni5/Aspni5.home.html`, 2009. [Online; accessed 10-Aug-2009].

[30] DOE Joint Genome Institute. Joint Genome Institute *Sporotrichum thermophile*. `http://genome.jgi-psf.org/Spoth1/Spoth1.home.html`, 2010. [Online; accessed 08-Jul-2010].

[31] S. R. Eddy. A probabilistic model of local sequence alignment that simplifies statistical significance estimation. *PLoS Computational Biology*, 4(5):e1000069, 2008.

[32] B. Efron, R. Tibshirani, J. D. Storey, and V. Tusher. Empirical Bayes analysis of a microarray experiment. *Journal of the American Statistical Association*, 96(456):1151–1160, 2001.

[33] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, 1998.

[34] T. Emonet, C. M. Macal, M. J. North, C. E. Wickersham, and P. Cluzel. AgentCell: a digital single-cell assay for bacterial chemotaxis. *Bioinformatics*, 21(11):2714–2721, 2005.

[35] J. J. Faith, B. Hayete, J. T. Thaden, I. Mogno, J. Wierzbowski, G. Cottarel, S. Kasif, J. J. Collins, and T. S. Gardner. Large-scale mapping and validation of *Escherichia coli* transcriptional regulation from a compendium of expression profiles. *PLoS Biology*, 5(1):54–66, 2007.

[36] R. D. Finn, J. Tate, J. Mistry, P. C. Coggill, S. J. Sammut, H.-R. Hotz, G. Ceric, K. Forslund, S. R. Eddy, E. L. L. Sonnhammer, and A. Bateman. The Pfam protein families database. *Nucleic Acids Research*, 36(suppl 1):D281–D288, 2008.

[37] P. Flicek, M. R. Amode, and D. Barrell. Ensembl 2011. *Nucleic Acids Research*, 39(suppl 1):D800–D806, 2011.

[38] L. Florea, G. Hartzell, Z. Zhang, G. M. Rubin, and W. Miller. A computer program for aligning a cDNA sequence with a genomic DNA sequence. *Genome Research*, 8(9):967–974, 1998.

[39] N. Friedman. Inferring cellular networks using probabilistic graphical models. *Science*, 303(5659):799–805, 2004.

[40] N. Friedman, M. Goldszmidt, and A. Wyn. Data analysis with Bayesian networks: A bootstrap approach. In *Proceedings of 15th Conference on Uncertainty in Artificial Intelligence*, pages 206–215, 1999.

[41] N. Friedman and D. Koller. Being Bayesian about network structure. a Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50(1-2):95–125, 2003.

[42] N. Friedman, M. Linial, and I. Nachman. Using Bayesian networks to analyze expression data. *Journal of Computational Biology*, 7(3-4):601–620, 2000.

[43] N. Friedman, I. Nachman, and D. Pe'er. Learning Bayesian network structure from massive datasets: The "Sparse Candidate" algorithm. In *Proceedings of 15th Conference on Uncertainty in Artificial Intelligence*, pages 206–215, 1999.

[44] S. Gama-Castro, V. Jimenez-Jacinto, M. Peralta-Gil, A. Santos-Zavaleta, M. I. Penaloza-Spinola, B. Contreras-Moreira, J. Segura-Salazar, L. Muniz-Rascado, I. Martinez-Flores, H. Salgado, C. Bonavides-Martinez, C. Abreu-Goodger, C. Rodriguez-Penagos, J. Miranda-Rios, E. Morett, E. Merino, A. M. Huerta, L. Trevino-Quintanilla, and J. Collado-Vides. RegulonDB (version 6.0): gene regulation model of *Escherichia coli* K-12 beyond transcription, active (experimental) annotated promoters and Textpresso navigation. *Nucleic Acids Research*, 36(Database issue):D120–124, 2008.

[45] A. P. Gasch, P. T. Spellman, C. M. Kao, O. Carmel-Harel, M. B. Eisen, G. Storz, D. Botstein, and P. O. Brown. Genomic expression programs in the response of yeast cells to environmental changes. *Molecular and Cellular Biology*, 11(12):4241–4257, 2000.

[46] Gene Ontology Consortium. Creating the gene ontology resource: design and implementation. *Genome Research*, 11(8):1425–1433, 2001.

[47] Gene Ontology Consortium. The Gene Ontology (GO) database and informatics resource. *Nucleic Acid Research*, 32(Database issue):D258–D261, 2004.

[48] D. Gershon. Microarray technology: An array of opportunities. *Nature*, 416(6883):885–891, 2002.

[49] D. Ghosh and A. M. Chinnaiyan. Mixture modelling of gene expression data from microarray experiments. *Bioinformatics*, 18(2):275–286, 2002.

[50] P. Giudici and R. Castelo. Improving Markov Chain Monte Carlo model search for data mining. *Machine Learning*, 50(1):127–158, January 2003.

[51] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, 1999.

[52] M. Harbers and P. Carninci. Tag-based approaches for transcriptome research and genome annotation. *Nature Methods*, 2(7):495–502, 2005.

[53] A. A. Hasper, J. Visser, and L. H. De Graaff. The *Aspergillus niger* transcriptional activator XlnR, which is involved in the degradation of the polysaccharides xylan and cellulose, also regulates d-xylose reductase gene expression. *Molecular Microbiology*, 36(1):193–200, 2000.

[54] D. Heckerman. A tutorial on learning with Bayesian networks. In *Learning in graphical models*, pages 301–354. MIT Press, 1999.

[55] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky. Bayesian model averaging: A tutorial. *Statistical Science*, 14(4):382–401, 1999.

[56] R. A. Holt and S. J. Jones. The new paradigm of flow cell sequencing. *Genome Research*, 18(6):839–846, 2008.

[57] D. W. Huang, B. T. Sherman, and R. A. Lempicki. Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists. *Nucleic Acids Research*, 37(1):1–13, 2009.

[58] S. Hunter, R. Apweiler, T. K. Attwood, A. Bairoch, A. Bateman, D. Binns, P. Bork, U. Das, L. Daugherty, L. Duquenne, R. D. Finn, J. Gough, D. Haft, N. Hulo, D. Kahn, E. Kelly, A. Laugraud, I. Letunic, D. Lonsdale, R. Lopez, M. Madera, J. Maslen, C. McAnulla, J. McDowall, J. Mistry, A. Mitchell, N. Mulder, D. Natale, C. Orengo, A. F. Quinn, J. D. Selengut, C. J. A. Sigrist, M. Thimma, P. D. Thomas, F. Valentin, D. Wilson, C. H. Wu, and C. Yeats. InterPro: the integrative protein signature database. *Nucleic Acids Research*, 37(suppl 1):D211–D215, 2009.

[59] S. Imoto, T. Higuchi, T. Goto, K. Tashiro, S. Kuhara, and S. Miyano. Combining microarrays and biological knowledge for estimating gene networks via Bayesian networks. In *Proceedings of the IEEE Computer Society Conference on Bioinformatics*, 2003.

[60] R. A. Irizarry, C. Wang, Y. Zhou, and T. P. Speed. Gene set enrichment analysis made simple. *Statistical Methods in Medical Research*, 18(6):565–575, 2009.

[61] D. Jiang, C. Tang, and A. Zhang. Cluster analysis for gene expression data: a survey. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1370– 1386, 2004.

[62] A. Joshi, R. De Smet, K. Marchal, Y. Van de Peer, and T. Michoel. Module networks revisited: computational assessment and prioritization of model predictions. *Bioinformatics*, 25(4):490–496, 2009.

[63] A. Joshi, Y. Van de Peer, and T. Michoel. Analysis of a Gibbs sampler method for model-based clustering of gene expression data. *Bioinformatics*, 24(2):176–183, 2008.

[64] P. D. Karp, S. Paley, and P. Romero. The Pathway Tools software. *Bioinformatics*, 18(suppl 1):S225–S232, 2002.

[65] I. M. Keseler, J. Collado-Vides, A. Santos-Zavaleta, M. Peralta-Gil, S. Gama-Castro, L. Muiz-Rascado, C. Bonavides-Martinez, S. Paley, M. Krummenacker, T. Altman, P. Kaipa, A. Spaulding, J. Pacheco, M. Latendresse, C. Fulcher, M. Sarker, A. G. Shearer, A. Mackie, I. Paulsen, R. P. Gunsalus, and P. D. Karp. EcoCyc: a comprehensive database of *Escherichia coli* biology. *Nucleic Acids Research*, 39(suppl 1):D583–D590, 2011.

[66] J.-H. Kim and M. Johnston. Two glucose-sensing pathways converge on Rgt1 to regulate expression of glucose transporter genes in *Saccharomyces cerevisiae*. *Journal of Biological Chemistry*, 281(36):26144–26149, 2006.

[67] M. H. Kutner, J. Neter, C. J. Nachtsheim, and W. Li. *Applied Linear Statistical Models*. Boston : McGraw-Hill Irwin, 2005.

[68] P. H. Lee and D. Lee. Modularized learning of genetic interaction networks from biological annotations and mRNA expression data. *Bioinformatics*, 21(11):2739–2747, 2005.

[69] T. I. Lee, N. J. Rinaldi, F. Robert, D. T. Odom, Z. Bar-Joseph, G. K. Gerber, N. M. Hannett, C. T. Harbison, C. M. Thompson, I. Simon, J. Zeitlinger, E. G. Jennings, H. L. Murray, D. B. Gordon, B. Ren, J. J. Wyrick, J.-B. Tagne, T. L. Volkert, E. Fraenkel, D. K. Gifford, and R. A. Young. Transcriptional regulatory networks in *Saccharomyces cerevisiae*. *Science*, 298(5594):799–804, 2002.

[70] R. Leinonen, F. G. Diez, D. Binns, W. Fleischmann, R. Lopez, and R. Apweiler. UniProt archive. *Bioinformatics*, 20(17):3236–3237, 2004.

[71] J. Li, Z. J. Liu, Y. C. Pan, Q. Liu, X. Fu, N. G. Cooper, Y. Li, M. Qiu, and T. Shi. Regulatory module network of basic/helix-loop-helix transcription factors in mouse brain. *Genome Biology*, 8(11):R244, 2007.

[72] H. Liu, J. Li, and L. Wong. A comparative study of feature selection and classification methods using gene expression profiles and proteomic patterns. *Genome Informatics*, 13:51–60, 2002.

[73] J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer, 2004.

[74] A. Madar, A. Greenfield, E. Vanden-Eijnden, and R. Bonneau. DREAM3: Network inference using dynamic context likelihood of relatedness and the Inferelator. *PLoS ONE*, 5(3):e9803, 03 2010.

[75] D. Madigan, J. York, and D. Allard. Bayesian graphical models for discrete data. *International Statistical Review*, 63(2):215–232, 1995.

[76] S. Maere, K. Heymans, and M. Kuiper. BiNGO: a Cytoscape plugin to assess over-representation of gene ontology categories in biological networks. *Bioinformatics*, 21(16):3448–3449, 2005.

[77] B. Magasanik and C. A. Kaiser. Nitrogen regulation in *Saccharomyces cerevisiae*. *Gene*, 290(1-2):1 – 18, 2002.

[78] W. H. Majoros, M. Pertea, and S. L. Salzberg. TigrScan and GlimmerHMM: two open source *ab initio* eukaryotic gene-finders. *Bioinformatics*, 20(16):2878–2879, 2004.

[79] D. Marbach, R. J. Prill, T. Schaffter, C. Mattiussi, D. Floreano, and G. Stolovitzky. Revealing strengths and weaknesses of methods for gene network inference. *Proceedings of the National Academy of Sciences*, 107(14):6286–6291, 2010.

[80] D. Marbach, T. Schaffter, C. Mattiussi, and D. Floreano. Generating realistic in silico gene networks for performance assessment of reverse engineering methods. *Journal of Computational Biology*, 16(2):229–239, 2009.

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

[82] D. Martinez, L. Larrondo, N. Putnam, M. D. Sollewijn, M. D. Sollewijn Gelpke, K. Huang, J. Chapman, K. G. Helfenbein, P. Ramaiya, J. C. Detter, F. Larimer, P. M. Coutinho, B. Henrissat, R. Berka, and D. Cullen. Genome sequence of the lignocellulose degrading fungus *Phanerochaete chrysosporium* strain RP78. *Nature biotechnology*, 22(6):695–700, 2004.

[83] V. Matys, O. V. Kel-Margoulis, E. Fricke, I. Liebich, S. Land, A. Barre-Dirrie, I. Reuter, D. Chekmenev, M. Krull, K. Hornischer, N. Voss, P. Stegmaier,

B. Lewicki-Potapov, H. Saxel, A. E. Kel, and E. Wingender. TRANSFAC and its module TRANSCompel: transcriptional gene regulation in eukaryotes. *Nucleic Acids Research*, 34(Database issue):D108–110, 2006.

[84] O. Mayans, M. Scott, I. Connerton, T. Gravesen, J. Benen, J. Visser, R. Pickersgill, and J. Jenkins. Two crystal structures of pectin lyase A from *Aspergillus reveal* a pH driven conformational change and striking divergence in the substrate-binding clefts of pectin and pectate lyases. *Structure*, 5(5):677 – 689, 1997.

[85] H. W. Mewes, D. Frishman, U. Gldener, G. Mannhaupt, K. Mayer, M. Mokrejs, B. Morgenstern, M. Mnsterktter, S. Rudd, and B. Weil. MIPS: a database for genomes and protein sequences. *Nucleic Acids Research*, 30(1):31–34, 2002.

[86] T. Michoel, R. De Smet, A. Joshi, Y. Van de Peer, and K. Marchal. Comparative analysis of module-based versus direct methods for reverse-engineering transcriptional regulatory networks. *BMC Systems Biology*, 3(1):49, 2009.

[87] T. Michoel, S. Maere, E. Bonnet, A. Joshi, Y. Saeys, T. Van den Bulcke, K. Van Leemput, P. van Remortel, M. Kuiper, K. Marchal, and Y. Van de Peer. Validating module network learning algorithms using simulated data. *BMC Bioinformatics*, 8(Suppl 2):S5, 2007.

[88] P. T. Monteiro, N. D. Mendes, M. C. Teixeira, S. d'Orey, S. Tenreiro, N. P. Mira, H. Pais, A. P. Francisco, A. M. Carvalho, A. B. Lourenco, I. Sa-Correia, A. L. Oliveira, and A. T. Freitas. YEASTRACT-DISCOVERER: new tools to improve the analysis of transcriptional regulatory associations in *Saccharomyces cerevisiae*. *Nucleic Acids Research*, 36(suppl_1):D132–136, 2008.

[89] A. Mortazavi, B. A. Williams, K. McCue, L. Schaeffer, and B. Wold. Mapping and quantifying mammalian transcriptomes by RNA-Seq. *Nature Methods*, 5(7):621–8, 2008.

[90] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–53, 1970.

[91] K. Okubo, H. Sugawara, T. Gojobori, and Y. Tateno. DDBJ in preparation for overview of research activities behind data submissions. *Nucleic Acids Research*, 34(suppl_1):D6–9, 2006.

[92] D. Pe'er. *From gene expression to molecular pathways*. PhD thesis, The Hebrew University, 2003.

[93] H. J. Pel, J. H. de Winde, D. B. Archer, P. S. Dyer, G. Hofmann, P. J. Schaap, G. Turner, R. P. de Vries, R. Albang, K. Albermann, M. R. Andersen, J. D. Bendtsen, J. A. E. Benen, M. van den Berg, S. Breestraat, M. X. Caddick, R. Contreras, M. Cornell, P. M. Coutinho, E. G. J. Danchin, A. J. M. Debets, P. Dekker, P. W. M. van Dijck, A. van Dijk, L. Dijkhuizen, A. J. M. Driessen, C. d'Enfert, S. Geysens, C. Goosen, G. S. P. Groot, P. W. J. de Groot, T. Guillemette, B. Henrissat, M. Herweijer, J. P. T. W. van den Hombergh, C. A. M. J. J. van den Hondel, R. T. J. M. van der Heijden, R. M. van der Kaaij, F. M. Klis, H. J. Kools, C. P. Kubicek, P. A. van Kuyk, J. Lauber, X. Lu, M. J. E. C. van der Maarel, R. Meulenberg, H. Menke, M. A. Mortimer, J. Nielsen, S. G. Oliver, M. Olsthoorn, K. Pal, N. N. M. E. van Peij, A. F. J. Ram, U. Rinas, J. A. Roubos, C. M. J. Sagt, M. Schmoll, J. Sun, D. Ussery, J. Varga, W. Vervecken, P. J. J. van de Vondervoort, H. Wedler, H. A. B. Wosten, A.-P. Zeng, A. J. J. van Ooyen, J. Visser, and H. Stam. Genome sequencing and analysis of the versatile cell factory *Aspergillus niger* CBS 513.88. *Nature Biotechnology*, 25(2):221–231, 2007.

[94] J. Pevsner. *Bioinformatics and Functional Genomics, 2nd Edition*. Wiley-Blackwell, 2009.

[95] V. Pihur, S. Datta, and S. Datta. Weighted rank aggregation of cluster validation measures: a Monte Carlo cross-entropy approach. *Bioinformatics*, 23(13):1607–1615, 2007.

[96] V. Pihur, S. Datta, and S. Datta. RankAggreg, an R package for weighted rank aggregation. *BMC Bioinformatics*, 10(1):62, 2009.

[97] R. J. Prill, D. Marbach, J. Saez-Rodriguez, P. K. Sorger, L. G. Alexopoulos, X. Xue, N. D. Clarke, G. Altan-Bonnet, and G. Stolovitzky. Towards a rigorous assessment

of systems biology models: The DREAM3 challenges. *PLoS ONE*, 5(2):e9202, 02 2010.

[98] K. D. Pruitt, T. Tatusova, and D. R. Maglott. NCBI Reference Sequence (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic Acids Research*, 33(suppl 1):D501–D504, 2005.

[99] J. Qi, T. Michoel, and G. Butler. A regression tree-based Gibbs sampler to learn the regulation programs in a transcription regulatory module network. In *Proceedings of 2010 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 206–215, 2010.

[100] J. Qi, T. Michoel, and G. Butler. Applying linear models to learn regulation programs in a transcription regulatory module network. In *Proceedings of 9th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, pages 37–47, 2011.

[101] J. Qi, T. Michoel, and G. Butler. An integrative approach to infer regulation programs in a transcription regulatory module network. In *Proceedings of the 2011 ACM Conference on Bioinformatics, Computational Biology and Biomedicine*. ACM, 2011.

[102] Z. S. Qin. Clustering microarray gene expression data using weighted Chinese restaurant process. *Bioinformatics*, 22(16):1988–1997, 2006.

[103] G. J. G. Ruijter, S. A. Vanhanen, M. M. C. Gielkens, P. J. I. van de Vondervoort, and J. Visser. Isolation of *Aspergillus niger creA* mutants and effects of the mutations on expression of arabinases and L-arabinose catabolic enzymes. *Microbiology*, 143(9):2991–2998, 1997.

[104] H.-J. Schller. Transcriptional control of nonfermentative metabolism in the yeast *Saccharomyces cerevisiae*. *Current Genetics*, 43:139–160, 2003. 10.1007/s00294-003-0381-8.

[105] E. Segal, D. Pe'er, A. Regev, D. Koller, and N. Friedman. Learning module networks. *Journal of Machine Learning Research*, 6:557–588, 2005.

[106] E. Segal, M. Shapira, A. Regev, D. Pe'er, D. Botstein, D. Koller, and N. Friedman. Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data. *Nature Genetics*, 34(2):166–176, June 2003.

[107] E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller. Rich probabilistic models for gene expression. *Bioinformatics*, 17(suppl_1):S243–252, 2001.

[108] E. Segal, R. Yelensky, and D. Koller. Genome-wide discovery of transcriptional modules from DNA sequence and gene expression. *Bioinformatics*, 19(suppl.1):i273–282, 2003.

[109] N. Semova, R. Storms, T. John, P. Gaudet, P. Ulycznyj, X. Min, J. Sun, G. Butler, and A. Tsang. Generation, annotation, and analysis of an extensive *Aspergillus niger* EST collection. *BMC Microbiology*, 6(1):7, 2006.

[110] W. Shi, M. Bogdanov, W. Dowhan, and D. R. Zusman. The *pss* and *psd* genes are required for motility and chemotaxis in *Escherichia coli*. *Journal of Bacteriology*, 175(23):7711–7714, 1993.

[111] S. Sinha. `http://www.cs.uiuc.edu/homes/sinhas/img/DAILYILLINI.jpg`, 2011. [Online; accessed 12-Jan-2011].

[112] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[113] G. K. Smyth. Linear models and empirical Bayes methods for assessing differential expression in microarray experiments. *Statistical Applications in Genetics and Molecular Biology*, 3:Article3, 2004.

[114] G. K. Smyth. Limma: linear models for microarray data. In *Bioinformatics and Computational Biology Solutions using R and Bioconductor*, pages 397–420. Springer, 2005.

[115] G. Stolovitzky, R. J. Prill, and A. Califano. Lessons from the DREAM2 challenges. *Annals of the New York Academy of Sciences*, 1158:159–195, March 2009.

[116] A. Subramanian, P. Tamayo, V. K. Mootha, S. Mukherjee, B. L. Ebert, M. A. Gillette, A. Paulovich, S. L. Pomeroy, T. R. Golub, E. S. Lander, and J. P.

Mesirov. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences*, 102(43):15545–15550, 2005.

[117] P. A. C. 't Hoen, Y. Ariyurek, H. H. Thygesen, E. Vreugdenhil, R. H. A. M. Vossen, R. X. de Menezes, J. M. Boer, G.-J. B. van Ommen, and J. T. den Dunnen. Deep sequencing-based expression analysis shows major advances in robustness, resolution and inter-lab portability over five microarray platforms. *Nucleic Acids Research*, 36(21):e141, 2008.

[118] Y. Tamada, H. Bannai, S. Imoto, T. Katayama, M. Kanehisa, and S. Miyano. Utilizing evolutionary information and gene expression data for estimating gene networks with Bayesian network models. *Journal of Bioinformatics and Computational Biology*, 3(6):1295–1313, December 2005.

[119] Y. Tamada, S. Kim, H. Bannai, S. Imoto, K. Tashiro, S. Kuhara, and S. Miyano. Estimating gene networks from gene expression data by combining Bayesian network model with promoter element detection. *Bioinformatics*, 19(suppl.2):ii227–236, 2003.

[120] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. S. Lander, and T. R. Golub. Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. *Proceedings of the National Academy of Sciences*, 96(6):2907–2912, 1999.

[121] S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church. Systematic determination of genetic network architecture. *Nature Genetics*, 22(3):281–5, 1999.

[122] A. Tefferi, M. E. Bolander, S. M. Ansell, E. D. Wieben, and T. C. Spelsberg. Primer on medical genomics part. III: Microarray experiments and data analysis. *Mayo Clinic Proceedings*, 77(9):927–940, 2002.

[123] M. C. Teixeira, P. Monteiro, P. Jain, S. Tenreiro, A. R. Fernandes, N. P. Mira, M. Alenquer, A. T. Freitas, A. L. Oliveira, and I. S-Correia. The YEASTRACT database: a tool for the analysis of transcription regulatory associations in *Saccharomyces cerevisiae*. *Nucleic Acids Research*, 34(suppl 1):D446–D451, 2006.

[124] W. Thompson, E. C. Rouchka, and C. E. Lawrence. Gibbs Recursive Sampler: finding transcription factor binding sites. *Nucleic Acids Research*, 31(13):3580–3585, 2003.

[125] W. A. Thompson, L. A. Newberg, S. Conlan, L. A. McCue, and C. E. Lawrence. The Gibbs Centroid Sampler. *Nucleic Acids Research*, 35(Web Server issue):W232–237, 2007.

[126] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):pp. 267–288, 1996.

[127] C. Trapnell, L. Pachter, and S. L. Salzberg. TopHat: discovering splice junctions with RNA-Seq. *Bioinformatics*, 25(9):1105–1111, 2009.

[128] C. Trapnell, B. A. Williams, G. Pertea, A. Mortazavi, G. Kwan, M. J. van Baren, S. L. Salzberg, B. J. Wold, and L. Pachter. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology*, 5:511–5, 2010.

[129] V. G. Tusher, R. Tibshirani, and G. Chu. Significance analysis of microarrays applied to the ionizing radiation response. *Proceedings of the National Academy of Sciences*, 98(9):5116–5121, 2001.

[130] T. Van den Bulcke, K. Van Leemput, B. Naudts, P. van Remortel, H. Ma, A. Verschoren, B. De Moor, and K. Marchal. SynTReN: a generator of synthetic gene expression data for design and analysis of structure learning algorithms. *BMC Bioinformatics*, 7(1):43, 2006.

[131] V. E. Velculescu, L. Zhang, B. Vogelstein, and K. W. Kinzler. Serial analysis of gene expression. *Science*, 270(5235):484–487, 1995.

[132] H. Wang and F. Azuaje. Gene expression correlation and gene ontology-based similarity: An assessment of quantitative relationships. In *Proceedings of IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 25–31. IEEE Computer Society, 2004.

[133] Y. Wang, X.-S. Zhang, and Y. Xia. Predicting eukaryotic transcriptional cooperativity by Bayesian network integration of genome-wide data. *Nucleic Acids Research*, 37(18):5943–5958, 2009.

[134] Z. Wang, M. Gerstein, and M. Snyder. RNA-Seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics*, 10:57–63, 2009.

[135] A. V. Werhli and D. Husmeier. Reconstructing gene regulatory networks with Bayesian networks by combining expression data with multiple sources of prior knowledge. *Statistical Applications in Genetics and Molecular Biology*, 6(1), 2007.

[136] X. Xie, J. Lu, E. J. Kulbokas, T. R. Golub, V. Mootha, K. Lindblad-Toh, E. S. Lander, and M. Kellis. Systematic discovery of regulatory motifs in human promoters and 3 prime UTRs by comparison of several mammals. *Nature*, 434(7031):338–345, 2005.

[137] T. Zeng and J. Li. Maximization of negative correlations in time-course gene expression data for enhancing understanding of molecular pathways. *Nucleic Acids Research*, 38(1):e1, 2010.

Appendices

No	EST	Module Number	JGI Hit
1	Asn_00397	0	Aspni3|56553
2	Asn_00518	0	Aspni3|55604
3	Asn_00785	0	Aspni3|209668
4	Asn_01386	0	Aspni3|209397
5	Asn_01461	0	Aspni3|206434
6	Asn_01933	0	Aspni3|139271
7	Asn_04357	0	Aspni3|52535
8	Asn_04434	0	Aspni3|56782
9	Asn_04494	0	Aspni3|55668
10	Asn_06125	0	Aspni3|174330
11	Asn_07154	0	Aspni3|210730
12	Asn_07911	0	Aspni3|56628
13	Asn_08161	0	Aspni3|52520
14	Asn_08350	0	Aspni3|53284
15	Asn_08528	0	Aspni3|53978
16	Asn_08765	0	Aspni3|51764
17	Asn_10089	0	Aspni3|52520
18	Asn_00163	1	Aspni3|57436
19	Asn_00421	1	Aspni3|205670
20	Asn_00473	1	#N/A
21	Asn_00644	1	Aspni3|55136
22	Asn_00831	1	Aspni3|54490
23	Asn_00835	1	Aspni3|119631
24	Asn_00846	1	Aspni3|36764
25	Asn_00850	1	#N/A
26	Asn_01446	1	Aspni3|56619
27	Asn_01561	1	Aspni3|205927
28	Asn_02480	1	Aspni3|209771
29	Asn_04007	1	Aspni3|47818
30	Asn_04493	1	Aspni3|52817
31	Asn_04713	1	#N/A
32	Asn_05745	1	Aspni3|203198
33	Asn_07943	1	Aspni3|143109
34	Asn_08182	1	Aspni3|56619
35	Asn_00715	2	Aspni3|47967
36	Asn_00830	2	Aspni3|47967
37	Asn_00078	3	Aspni3|52118
38	Asn_00358	3	Aspni3|56225
39	Asn_00544	3	Aspni3|208318
40	Asn_01316	3	Aspni3|196413
41	Asn_01637	3	Aspni3|190162
42	Asn_01999	3	#N/A
43	Asn_02459	3	Aspni3|53173
44	Asn_02640	3	Aspni3|56880
45	Asn_03291	3	Aspni3|197015
46	Asn_03316	3	Aspni3|208898
47	Asn_04086	3	Aspni3|42733
48	Asn_04110	3	Aspni3|205620
49	Asn_04193	3	Aspni3|196413
50	Asn_04368	3	Aspni3|52040
51	Asn_06249	3	Aspni3|188169
52	Asn_10820	3	Aspni3|208318
53	Asn_00309	4	Aspni3|52919
54	Asn_00404	4	Aspni3|54097
55	Asn_00726	4	Aspni3|205904
56	Asn_00766	4	Aspni3|55947
57	Asn_00815	4	Aspni3|209875
58	Asn_01126	4	Aspni3|201613
59	Asn_01508	4	Aspni3|199734
60	Asn_01557	4	Aspni3|52865
61	Asn_02278	4	Aspni3|51702
62	Asn_02428	4	Aspni3|212334

Continued on next page

Table 16: continued

No	EST	Module Number	JGI Hit
63	Asn_04380	4	#N/A
64	Asn_04553	4	Aspni3|225596
65	Asn_06497	4	Aspni3|38983
66	Asn_07370	4	Aspni3|52525
67	Asn_00396	5	Aspni3|172668
68	Asn_00428	5	#N/A
69	Asn_00508	5	Aspni3|214348
70	Asn_00898	5	Aspni3|209521
71	Asn_00988	5	#N/A
72	Asn_01438	5	Aspni3|55185
73	Asn_01574	5	Aspni3|211265
74	Asn_01646	5	Aspni3|209252
75	Asn_01657	5	Aspni3|49934
76	Asn_01744	5	Aspni3|51653
77	Asn_01941	5	Aspni3|52670
78	Asn_02386	5	#N/A
79	Asn_02743	5	Aspni3|56152
80	Asn_03366	5	Aspni3|45434
81	Asn_04141	5	Aspni3|214458
82	Asn_04771	5	Aspni3|53082
83	Asn_04798	5	Aspni3|54952
84	Asn_05148	5	Aspni3|52530
85	Asn_05786	5	#N/A
86	Asn_06020	5	Aspni3|205376
87	Asn_06737	5	Aspni3|205909
88	Asn_07192	5	#N/A
89	Asn_07224	5	Aspni3|53496
90	Asn_07531	5	Aspni3|225679
91	Asn_08332	5	Aspni3|212570
92	Asn_09151	5	Aspni3|200758
93	Asn_00331	6	Aspni3|57243
94	Asn_00470	6	#N/A
95	Asn_00506	6	Aspni3|54742
96	Asn_00507	6	Aspni3|53522
97	Asn_00563	6	Aspni3|196122
98	Asn_00593	6	Aspni3|55590
99	Asn_00613	6	Aspni3|54021
100	Asn_00952	6	Aspni3|54615
101	Asn_01045	6	Aspni3|204445
102	Asn_01451	6	Aspni3|127335
103	Asn_01602	6	Aspni3|200760
104	Asn_01943	6	#N/A
105	Asn_02074	6	Aspni3|214715
106	Asn_05354	6	Aspni3|175896
107	Asn_07606	6	Aspni3|56475
108	Asn_10322	6	Aspni3|209669
109	Asn_10482	6	Aspni3|56523
110	Asn_00075	7	Aspni3|41606
111	Asn_00765	7	Aspni3|56177
112	Asn_00974	7	Aspni3|196989
113	Asn_01607	7	Aspni3|198257
114	Asn_01664	7	Aspni3|55069
115	Asn_01919	7	Aspni3|200329
116	Asn_02012	7	Aspni3|54106
117	Asn_05808	7	Aspni3|225673
118	Asn_00524	8	Aspni3|124156
119	Asn_00686	8	Aspni3|52071
120	Asn_01360	8	Aspni3|56084
121	Asn_01537	8	Aspni3|208547
122	Asn_01770	8	Aspni3|203198
123	Asn_02169	8	Aspni3|183088

Continued on next page

Table 16: continued

No	EST	Module Number	JGI Hit
124	Asn_02189	8	Aspni3|54140
125	Asn_02307	8	Aspni3|51662
126	Asn_04445	8	Aspni3|56093
127	Asn_04446	8	Aspni3|205670
128	Asn_04727	8	Aspni3|211544
129	Asn_05204	8	Aspni3|198680
130	Asn_05299	8	Aspni3|207261
131	Asn_05346	8	Aspni3|200605
132	Asn_05634	8	#N/A
133	Asn_06231	8	Aspni3|54837
134	Asn_06795	8	Aspni3|51997
135	Asn_08044	8	Aspni3|208192
136	Asn_08784	8	Aspni3|53386
137	Asn_09354	8	Aspni3|38546
138	Asn_00743	9	Aspni3|214270
139	Asn_00927	9	Aspni3|54445
140	Asn_01026	9	Aspni3|55147
141	Asn_01382	9	Aspni3|206952
142	Asn_01560	9	#N/A
143	Asn_04275	9	Aspni3|52410
144	Asn_04287	9	Aspni3|54038
145	Asn_04299	9	Aspni3|122060
146	Asn_04301	9	Aspni3|54270
147	Asn_04311	9	Aspni3|211963
148	Asn_04313	9	Aspni3|55501
149	Asn_06610	9	Aspni3|57363
150	Asn_07607	9	Aspni3|180348
151	Asn_08711	9	Aspni3|208246
152	Asn_01022	10	Aspni3|197162
153	Asn_01049	10	Aspni3|124807
154	Asn_00107	11	Aspni3|56390
155	Asn_01789	11	#N/A
156	Asn_02113	11	Aspni3|56954
157	Asn_02441	11	Aspni3|207313
158	Asn_02492	11	Aspni3|52545
159	Asn_03280	11	Aspni3|52544
160	Asn_08808	11	Aspni3|189022
161	Asn_00065	12	Aspni3|47911
162	Asn_00260	12	Aspni3|213597
163	Asn_00547	12	Aspni3|214233
164	Asn_00584	12	Aspni3|207710
165	Asn_00696	12	Aspni3|200686
166	Asn_00697	12	Aspni3|54046
167	Asn_00757	12	Aspni3|46394
168	Asn_00917	12	Aspni3|214233
169	Asn_01759	12	Aspni3|209422
170	Asn_02060	12	Aspni3|51738
171	Asn_02218	12	Aspni3|207276
172	Asn_02240	12	Aspni3|52741
173	Asn_02254	12	#N/A
174	Asn_02264	12	Aspni3|56788
175	Asn_02265	12	Aspni3|213369
176	Asn_02350	12	Aspni3|208611
177	Asn_02384	12	Aspni3|214624
178	Asn_02704	12	Aspni3|41522
179	Asn_03271	12	Aspni3|52004
180	Asn_03282	12	Aspni3|53268
181	Asn_03617	12	Aspni3|202289
182	Asn_03864	12	Aspni3|136085
183	Asn_04128	12	Aspni3|202301
184	Asn_04133	12	Aspni3|173099

Continued on next page

Table 16: continued

No	EST	Module Number	JGI Hit
185	Asn_04627	12	Aspni3|45867
186	Asn_04908	12	Aspni3|57215
187	Asn_05574	12	Aspni3|54015
188	Asn_05797	12	#N/A
189	Asn_06532	12	Aspni3|179998
190	Asn_06724	12	Aspni3|51883
191	Asn_07659	12	#N/A
192	Asn_07673	12	Aspni3|207278
193	Asn_08867	12	Aspni3|130814
194	Asn_08878	12	Aspni3|55114
195	Asn_09005	12	Aspni3|197473
196	Asn_09186	12	Aspni3|37174
197	Asn_09187	12	Aspni3|49515
198	Asn_10425	12	Aspni3|205396
199	Asn_10480	12	#N/A
200	Asn_00186	13	Aspni3|53978
201	Asn_00290	13	Aspni3|56628
202	Asn_00460	13	Aspni3|187673
203	Asn_00910	13	Aspni3|56643
204	Asn_00981	13	Aspni3|208387
205	Asn_01031	13	Aspni3|203267
206	Asn_01102	13	Aspni3|172786
207	Asn_01374	13	#N/A
208	Asn_01375	13	Aspni3|202623
209	Asn_01791	13	Aspni3|53545
210	Asn_02245	13	Aspni3|206607
211	Asn_02400	13	#N/A
212	Asn_02714	13	Aspni3|207862
213	Asn_03498	13	Aspni3|52449
214	Asn_04082	13	Aspni3|55680
215	Asn_06949	13	Aspni3|183268
216	Asn_08517	13	#N/A
217	Asn_00105	14	Aspni3|206342
218	Asn_00277	14	Aspni3|52270
219	Asn_00534	14	Aspni3|53252
220	Asn_01142	14	Aspni3|200887
221	Asn_02026	14	Aspni3|184327
222	Asn_04815	14	Aspni3|56887
223	Asn_05200	14	Aspni3|130233
224	Asn_05699	14	Aspni3|208022
225	Asn_06220	14	Aspni3|52186
226	Asn_06676	14	Aspni3|213505
227	Asn_07438	14	Aspni3|126214
228	Asn_07461	14	Aspni3|214859
229	Asn_07474	14	Aspni3|54801

Table 16: *A. niger* EST module assignment

Gene	Module Number	Annotation
Spoth1|105405	1	unknown
Spoth1|105650	1	unknown
Spoth1|109530	1	unknown
Spoth1|109941	1	unknown
Spoth1|110006	1	unknown
Spoth1|110210	1	unknown
Spoth1|110274	1	unknown
Spoth1|110466	1	unknown
Spoth1|110569	1	unknown
Spoth1|110597	1	unknown
Spoth1|110627	1	unknown
Spoth1|110699	1	unknown
Spoth1|110707	1	unknown
Spoth1|110817	1	unknown
Spoth1|110927	1	unknown
Spoth1|110948	1	unknown
Spoth1|111005	1	unknown
Spoth1|111118	1	unknown
Spoth1|111196	1	unknown
Spoth1|111425	1	unknown
Spoth1|111493	1	unknown
Spoth1|111512	1	unknown
Spoth1|111593	1	unknown
Spoth1|111636	1	unknown
Spoth1|111694	1	unknown
Spoth1|111695	1	unknown
Spoth1|111773	1	unknown
Spoth1|111817	1	unknown
Spoth1|111881	1	unknown
Spoth1|112017	1	unknown
Spoth1|112150	1	unknown
Spoth1|112534	1	unknown
Spoth1|112551	1	unknown
Spoth1|112561	1	unknown
Spoth1|112583	1	unknown
Spoth1|112682	1	unknown
Spoth1|113725	1	unknown
Spoth1|116020	1	unknown
Spoth1|17667	1	unknown
Spoth1|36429	1	unknown
Spoth1|39517	1	unknown
Spoth1|47786	1	unknown
Spoth1|55762	1	unknown
Spoth1|59591	1	unknown
Spoth1|62593	1	unknown
Spoth1|63764	1	unknown
Spoth1|67358	1	unknown
Spoth1|68104	1	unknown
Spoth1|71369	1	unknown
Spoth1|71631	1	unknown
Spoth1|72204	1	unknown
Spoth1|72333	1	unknown
Spoth1|72822	1	unknown
Spoth1|75107	1	unknown
Spoth1|76559	1	unknown
Spoth1|77277	1	unknown
Spoth1|78218	1	unknown
Spoth1|79440	1	unknown
Spoth1|81425	1	unknown
Spoth1|83980	1	unknown
Spoth1|86357	1	unknown
Spoth1|86979	1	unknown

Continued on next page

Table 17: continued

Gene	Module Number	Annotation
Spoth1|87141	1	unknown
Spoth1|87236	1	unknown
Spoth1|87328	1	unknown
Spoth1|87600	1	unknown
Spoth1|87961	1	unknown
Spoth1|88229	1	unknown
Spoth1|88290	1	unknown
Spoth1|88813	1	unknown
Spoth1|88876	1	unknown
Spoth1|89090	1	unknown
Spoth1|91450	1	unknown
Spoth1|102781	2	unknown
Spoth1|104936	2	Histidine triad (HIT) protein
Spoth1|106343	2	D(P)-binding
Spoth1|109542	2	Alternative oxidase
Spoth1|109875	2	unknown
Spoth1|110028	2	Basic helix-loop-helix dimerisation region bHLH
Spoth1|110187	2	unknown
Spoth1|110640	2	Globin-like
Spoth1|111025	2	No domain
Spoth1|111232	2	unknown
Spoth1|111928	2	unknown
Spoth1|112100	2	No domain
Spoth1|112121	2	Thiamine pyrophosphate enzyme, central region
Spoth1|112227	2	D-xylulose 5-phosphate/D-fructose 6-phosphate phosphoketolase
Spoth1|112340	2	unknown
Spoth1|112491	2	Sugar transporter
Spoth1|112607	2	unknown
Spoth1|114633	2	unknown
Spoth1|116450	2	unknown
Spoth1|117297	2	No domain
Spoth1|12282	2	Beta-Ig-H3/fasciclin
Spoth1|35073	2	unknown
Spoth1|44608	2	unknown
Spoth1|45669	2	No domain
Spoth1|51371	2	unknown
Spoth1|63965	2	Glutamine synthetase, catalytic region
Spoth1|72700	2	unknown
Spoth1|80825	2	unknown
Spoth1|80919	2	unknown
Spoth1|81031	2	unknown
Spoth1|82154	2	unknown
Spoth1|82223	2	unknown
Spoth1|85131	2	No domain
Spoth1|86378	2	unknown
Spoth1|86783	2	unknown
Spoth1|88036	2	unknown
Spoth1|104913	3	unknown
Spoth1|105536	3	unknown
Spoth1|107442	3	unknown
Spoth1|107663	3	unknown
Spoth1|108137	3	unknown
Spoth1|109565	3	unknown
Spoth1|110231	3	unknown
Spoth1|110263	3	unknown
Spoth1|110451	3	unknown
Spoth1|110709	3	unknown
Spoth1|110717	3	unknown
Spoth1|110811	3	unknown
Spoth1|111107	3	unknown
Spoth1|111110	3	unknown

Continued on next page

Table 17: continued

Gene	Module Number	Annotation
Spoth1|111142	3	unknown
Spoth1|111191	3	unknown
Spoth1|111426	3	unknown
Spoth1|111482	3	unknown
Spoth1|111519	3	unknown
Spoth1|111987	3	unknown
Spoth1|112029	3	unknown
Spoth1|112136	3	unknown
Spoth1|112137	3	unknown
Spoth1|112165	3	unknown
Spoth1|112492	3	unknown
Spoth1|112499	3	unknown
Spoth1|112605	3	unknown
Spoth1|112708	3	unknown
Spoth1|113613	3	unknown
Spoth1|113873	3	unknown
Spoth1|116034	3	unknown
Spoth1|18022	3	unknown
Spoth1|36979	3	unknown
Spoth1|42718	3	unknown
Spoth1|59772	3	unknown
Spoth1|62195	3	unknown
Spoth1|63138	3	unknown
Spoth1|63276	3	unknown
Spoth1|64541	3	unknown
Spoth1|67310	3	unknown
Spoth1|67711	3	unknown
Spoth1|70702	3	unknown
Spoth1|72741	3	unknown
Spoth1|74496	3	unknown
Spoth1|74526	3	unknown
Spoth1|75870	3	unknown
Spoth1|79084	3	unknown
Spoth1|79643	3	unknown
Spoth1|80725	3	unknown
Spoth1|80889	3	unknown
Spoth1|86721	3	unknown
Spoth1|86936	3	unknown
Spoth1|87186	3	unknown
Spoth1|87550	3	unknown
Spoth1|87711	3	unknown
Spoth1|88076	3	unknown
Spoth1|103702	4	Aldose 1-epimerase
Spoth1|109566	4	Cellulose-binding region, fungal
Spoth1|109678	4	NULL
Spoth1|110651	4	Cellulose-binding region, fungal
Spoth1|111372	4	Cellulose-binding region, fungal
Spoth1|111388	4	Cellulose-binding region, fungal
Spoth1|112050	4	Cellulose-binding region, fungal
Spoth1|112089	4	Glycoside hydrolase, family 61
Spoth1|112264	4	No domain
Spoth1|112306	4	Methionine synthase, vitamin-B12 independent
Spoth1|112399	4	Cellulose-binding region, fungal
Spoth1|112471	4	Oxidoreductase, N-terminal
Spoth1|114107	4	General substrate transporter
Spoth1|114673	4	unknown
Spoth1|115968	4	Glycoside hydrolase, family 1
Spoth1|116553	4	Glycoside hydrolase, family 10
Spoth1|33936	4	Cellulose-binding region, fungal
Spoth1|39555	4	Glycoside hydrolase, family 43
Spoth1|46583	4	Cellulose-binding region, fungal

Continued on next page

Table 17: continued

Gene	Module Number	Annotation
Spoth1|66729	4	Cellulose-binding region, fungal
Spoth1|68753	4	No domain
Spoth1|76901	4	Glycoside hydrolase, family 45
Spoth1|80104	4	Glycoside hydrolase, family 43
Spoth1|80312	4	Cellulose-binding region, fungal
Spoth1|81925	4	FAD dependent oxidoreductase
Spoth1|84133	4	Lipase, GDSL
Spoth1|84164	4	General substrate transporter
Spoth1|86753	4	Cellulose-binding region, fungal
Spoth1|89872	4	Ribose/galactose isomerase
Spoth1|98003	4	Cellulose-binding region, fungal
Spoth1|102322	5	Pectate lyase, catalytic
Spoth1|103536	5	No domain
Spoth1|103539	5	unknown
Spoth1|108890	5	Sugar transporter
Spoth1|112014	5	Alcohol dehydrogenase, zinc-binding
Spoth1|46981	5	Dehydroquinase class I
Spoth1|52160	5	Glycosyl hydrolase, family 88
Spoth1|52463	5	Pectate lyase/Amb allergen
Spoth1|52525	5	NULL
Spoth1|52713	5	Pectin lyase fold/virulence factor
Spoth1|71204	5	Pectate lyase/Amb allergen
Spoth1|71406	5	No domain
Spoth1|79295	5	Oxidoreductase, N-terminal
Spoth1|81869	5	Mandelate racemase/muconate lactonizing enzyme, C-terminal
Spoth1|82505	5	Mandelate racemase/muconate lactonizing enzyme, C-terminal
Spoth1|87557	5	Dihydrodipicolinate synthetase
Spoth1|90594	5	Pectate lyase/Amb allergen
Spoth1|95168	5	No domain
Spoth1|97342	5	RTA1 like protein
Spoth1|98480	5	Lipase, GDSL
Spoth1|100518	7	Glycoside hydrolase, family 61
Spoth1|103537	7	Glycoside hydrolase, family 61
Spoth1|109508	7	No domain
Spoth1|110117	7	unknown
Spoth1|113698	7	unknown
Spoth1|43353	7	NULL
Spoth1|52068	7	Glycoside hydrolase, family 5
Spoth1|71222	7	Phosphate transporter
Spoth1|73312	7	unknown
Spoth1|74713	7	unknown
Spoth1|74716	7	unknown
Spoth1|76393	7	No domain
Spoth1|84297	7	Glycoside hydrolase, family 5
Spoth1|90655	7	Glycoside hydrolase family 2, immunoglobulin-like beta-sandwich
Spoth1|95378	7	No domain
Spoth1|97137	7	Glycoside hydrolase, family 7
Spoth1|97899	7	Glycosyltransferase sugar-binding region containing DXD motif
Spoth1|100838	8	unknown
Spoth1|102522	8	Pectin lyase fold/virulence factor
Spoth1|102694	8	Glycoside hydrolase, family 81
Spoth1|106218	8	unknown
Spoth1|108784	8	unknown
Spoth1|110315	8	unknown
Spoth1|111165	8	NULL
Spoth1|111313	8	No domain
Spoth1|111908	8	unknown
Spoth1|111909	8	unknown
Spoth1|112308	8	unknown
Spoth1|116694	8	unknown
Spoth1|12427	8	unknown

Continued on next page

Table 17: continued

Gene	Module Number	Annotation
Spoth1|48666	8	unknown
Spoth1|50608	8	Glycoside hydrolase, family 18, N-terminal
Spoth1|50987	8	FAD linked oxidase, N-terminal
Spoth1|54328	8	unknown
Spoth1|56454	8	No domain
Spoth1|77995	8	unknown
Spoth1|79865	8	unknown
Spoth1|82368	8	unknown
Spoth1|84416	8	unknown
Spoth1|90182	8	Glycoside hydrolase, family 16
Spoth1|96790	8	No domain
Spoth1|100103	9	No domain
Spoth1|100909	9	unknown
Spoth1|102138	9	No domain
Spoth1|104825	9	unknown
Spoth1|106768	9	Nitroreductase
Spoth1|108831	9	unknown
Spoth1|109821	9	unknown
Spoth1|109823	9	unknown
Spoth1|109909	9	unknown
Spoth1|110063	9	No domain
Spoth1|110461	9	unknown
Spoth1|110491	9	unknown
Spoth1|110916	9	unknown
Spoth1|111289	9	unknown
Spoth1|111331	9	unknown
Spoth1|111567	9	unknown
Spoth1|111620	9	unknown
Spoth1|111708	9	No domain
Spoth1|111914	9	unknown
Spoth1|111934	9	unknown
Spoth1|112305	9	unknown
Spoth1|112338	9	unknown
Spoth1|112430	9	unknown
Spoth1|112431	9	unknown
Spoth1|112433	9	unknown
Spoth1|112477	9	unknown
Spoth1|112908	9	unknown
Spoth1|113582	9	unknown
Spoth1|114529	9	unknown
Spoth1|115479	9	No domain
Spoth1|115802	9	unknown
Spoth1|117357	9	unknown
Spoth1|23680	9	unknown
Spoth1|38458	9	Phospholipase A2, prokaryotic/fungal
Spoth1|40689	9	unknown
Spoth1|43088	9	unknown
Spoth1|44749	9	unknown
Spoth1|53851	9	unknown
Spoth1|56170	9	unknown
Spoth1|60177	9	unknown
Spoth1|63495	9	unknown
Spoth1|64356	9	unknown
Spoth1|64893	9	unknown
Spoth1|66848	9	unknown
Spoth1|68674	9	unknown
Spoth1|69394	9	unknown
Spoth1|71748	9	unknown
Spoth1|73461	9	unknown
Spoth1|78858	9	unknown
Spoth1|81466	9	No domain

Continued on next page

Table 17: continued

Gene	Module Number	Annotation
Spoth1|82399	9	unknown
Spoth1|83451	9	unknown
Spoth1|84480	9	unknown
Spoth1|86807	9	unknown
Spoth1|87390	9	unknown
Spoth1|87485	9	unknown
Spoth1|88885	9	unknown
Spoth1|95475	9	No domain
Spoth1|106502	10	Calycin-like
Spoth1|110778	10	No domain
Spoth1|110783	10	NULL
Spoth1|112714	10	FAD-dependent pyridine nucleotide-disulphide oxidoreductase
Spoth1|114666	10	Alcohol dehydrogenase, zinc-binding
Spoth1|114670	10	No domain
Spoth1|75151	10	Methyltransferase type 12
Spoth1|78013	10	AMP-dependent synthetase and ligase
Spoth1|78019	10	Tetracycline resistance protein, TetB
Spoth1|87202	10	No domain
Spoth1|98002	10	Methyltransferase type 11
Spoth1|102354	12	unknown
Spoth1|109981	12	unknown
Spoth1|110193	12	unknown
Spoth1|110306	12	unknown
Spoth1|110422	12	unknown
Spoth1|110889	12	unknown
Spoth1|110946	12	unknown
Spoth1|111041	12	unknown
Spoth1|111449	12	unknown
Spoth1|111487	12	unknown
Spoth1|111963	12	unknown
Spoth1|112204	12	ATPase, F0 complex, subunit C
Spoth1|112294	12	unknown
Spoth1|112644	12	unknown
Spoth1|112673	12	unknown
Spoth1|114719	12	unknown
Spoth1|115855	12	unknown
Spoth1|27308	12	unknown
Spoth1|43545	12	No domain
Spoth1|45292	12	No domain
Spoth1|45938	12	unknown
Spoth1|57398	12	No domain
Spoth1|57901	12	No domain
Spoth1|75824	12	unknown
Spoth1|75993	12	unknown
Spoth1|79085	12	unknown
Spoth1|82782	12	Major facilitator superfamily MFS-1
Spoth1|83475	12	D-dependent epimerase/dehydratase
Spoth1|84481	12	unknown
Spoth1|86076	12	unknown
Spoth1|86826	12	unknown
Spoth1|87434	12	unknown
Spoth1|105197	13	Zinc finger, AN1-type
Spoth1|105969	13	No domain
Spoth1|106527	13	Signal transduction response regulator, receiver region
Spoth1|109138	13	Major facilitator superfamily MFS-1
Spoth1|109157	13	unknown
Spoth1|109510	13	Heat shock protein Hsp20
Spoth1|109691	13	unknown
Spoth1|110220	13	unknown
Spoth1|110317	13	L-lactate/malate dehydrogenase
Spoth1|110342	13	unknown

Continued on next page

Table 17: continued

Gene	Module Number	Annotation
Spoth1|110532	13	Heat shock protein Hsp20
Spoth1|111479	13	unknown
Spoth1|111656	13	No domain
Spoth1|112322	13	Globin, subset
Spoth1|112686	13	unknown
Spoth1|115756	13	unknown
Spoth1|116156	13	unknown
Spoth1|16943	13	unknown
Spoth1|20534	13	unknown
Spoth1|28069	13	Zinc finger, RING-type
Spoth1|61543	13	No domain
Spoth1|76301	13	unknown
Spoth1|76670	13	DH:flavin oxidoreductase/NADH oxidase, N-terminal
Spoth1|76752	13	unknown
Spoth1|80427	13	Chaperonin clpA/B
Spoth1|84198	13	unknown
Spoth1|84852	13	unknown
Spoth1|85648	13	unknown
Spoth1|88496	13	unknown
Spoth1|89097	13	No domain
Spoth1|97046	13	unknown
Spoth1|103580	14	No domain
Spoth1|109532	14	unknown
Spoth1|109720	14	unknown
Spoth1|109978	14	Pyruvate carboxyltransferase
Spoth1|110427	14	unknown
Spoth1|110586	14	unknown
Spoth1|111339	14	unknown
Spoth1|111693	14	unknown
Spoth1|111780	14	unknown
Spoth1|112284	14	unknown
Spoth1|112362	14	Regulator of G protein signalling superfamily
Spoth1|112637	14	unknown
Spoth1|113666	14	unknown
Spoth1|115646	14	unknown
Spoth1|115983	14	No domain
Spoth1|43157	14	unknown
Spoth1|46070	14	No domain
Spoth1|57926	14	unknown
Spoth1|60294	14	unknown
Spoth1|60509	14	Amino acid transporter, transmembrane
Spoth1|70089	14	Bacterial transferase hexapeptide repeat
Spoth1|78828	14	No domain
Spoth1|80072	14	unknown
Spoth1|80133	14	No domain
Spoth1|86146	14	No domain
Spoth1|89037	14	No domain
Spoth1|90641	14	Tetracycline resistance protein, TetA
Spoth1|94208	14	unknown
Spoth1|100068	15	Cellulose-binding region, fungal
Spoth1|103032	15	Glycoside hydrolase, family 43
Spoth1|103054	15	Neuraminidase
Spoth1|108917	15	No domain
Spoth1|109444	15	Glycoside hydrolase, family 12
Spoth1|109943	15	Glycoside hydrolase, family 18, catalytic domain
Spoth1|111088	15	Cellulose-binding region, fungal
Spoth1|39279	15	Glycoside hydrolase, family 62
Spoth1|49824	15	Glycoside hydrolase, family 11
Spoth1|51596	15	unknown
Spoth1|55869	15	Cellulose-binding region, fungal
Spoth1|55982	15	Glycoside hydrolase, family 62

Continued on next page

Table 17: continued

Gene	Module Number	Annotation
Spoth1|56237	15	Glycoside hydrolase, family 11
Spoth1|79765	15	Glycoside hydrolase, family 61
Spoth1|85556	15	Glycoside hydrolase, family 61
Spoth1|89603	15	Glycoside hydrolase, family 11
Spoth1|92668	15	Glycoside hydrolase, family 61
Spoth1|96478	15	Esterase, PHB depolymerase
Spoth1|98122	15	Glycoside hydrolase, family 61
Spoth1|99678	15	Lipase, GDSL

Table 17: *S. thermophile* gene module assignment

Gene	Module Number	Annotation
Phchr1|00431	1	40S ribosomal protein S21
Phchr1|00436	1	Heat shock protein 90 homolog
Phchr1|00439	1	Nascent polypeptide-associated complex subunit alpha
Phchr1|00639	1	Arginase
Phchr1|01404	1	Tubulin beta chain
Phchr1|01553	1	60S ribosomal protein L38
Phchr1|01653	1	60S ribosomal protein L8
Phchr1|02127	1	unknown
Phchr1|02159	1	40S ribosomal protein S16
Phchr1|02166	1	60S ribosomal protein L13
Phchr1|02226	1	40S ribosomal protein S11
Phchr1|02233	1	Isopentenyl-diphosphate Delta-isomerase
Phchr1|02292	1	60S ribosomal protein L32-A
Phchr1|02329	1	Polyubiquitin
Phchr1|02508	1	Farnesyl pyrophosphate synthase
Phchr1|02616	1	40S ribosomal protein S14
Phchr1|02643	1	60S ribosomal protein L24
Phchr1|02655	1	Hydrophobin-1
Phchr1|02674	1	Glucosamine 6-phosphate N-acetyltransferase 1
Phchr1|02744	1	60S ribosomal protein L23
Phchr1|03099	1	Protein priA
Phchr1|03367	1	40S ribosomal protein S28
Phchr1|03374	1	60S ribosomal protein L22
Phchr1|03563	1	unknown
Phchr1|04170	1	Protein SnodProt1
Phchr1|04571	1	V-type proton ATPase 16 kDa proteolipid subunit
Phchr1|04674	1	60S ribosomal protein L25
Phchr1|04812	1	40S ribosomal protein S9-B
Phchr1|04813	1	60S ribosomal protein L21-A
Phchr1|05242	1	40S ribosomal protein S7
Phchr1|05252	1	Ribonucleoside-diphosphate reductase small chain
Phchr1|05456	1	60S ribosomal protein L2
Phchr1|05544	1	40S ribosomal protein S1
Phchr1|05715	1	unknown
Phchr1|05908	1	60S ribosomal protein L6-2
Phchr1|05913	1	60S ribosomal protein L36
Phchr1|05923	1	60S ribosomal protein L9-A
Phchr1|06594	1	40S ribosomal protein S0
Phchr1|06651	1	60S ribosomal protein L5-B
Phchr1|06871	1	40S ribosomal protein S2
Phchr1|07055	1	60S ribosomal protein L1-B
Phchr1|07088	1	40S ribosomal protein S12
Phchr1|07275	1	Probable 60S ribosomal protein L37-B
Phchr1|07311	1	60S ribosomal protein L35a-4
Phchr1|07335	1	Histone H4
Phchr1|07336	1	Histone H3.2

Continued on next page

Table 18: continued

Gene	Module Number	Annotation
Phchr1|07358	1	Histone H3.2
Phchr1|07359	1	Histone H4
Phchr1|07452	1	17.7 kDa class I heat shock protein
Phchr1|07508	1	40S ribosomal protein S15
Phchr1|07509	1	60S acidic ribosomal protein P2
Phchr1|07678	1	60S ribosomal protein L37a
Phchr1|07717	1	Delta(24(24(1)))-sterol reductase
Phchr1|07942	1	Ubiquitin-40S ribosomal protein S27a-2
Phchr1|07987	1	60S ribosomal protein L11
Phchr1|08256	1	40S ribosomal protein S10-B
Phchr1|08895	1	UPF0368 protein YPL225W
Phchr1|09016	1	ATP synthase subunit gamma, mitochondrial
Phchr1|09047	1	60S ribosomal protein L27-A
Phchr1|09253	1	40S ribosomal protein S22
Phchr1|09254	1	40S ribosomal protein S17-B
Phchr1|09257	1	unknown
Phchr1|09342	1	unknown
Phchr1|09531	1	60S ribosomal protein L31
Phchr1|09535	1	60S ribosomal protein L34-A
Phchr1|09608	1	60S ribosomal protein L20
Phchr1|09768	1	60S ribosomal protein L14-A
Phchr1|09993	1	Guanine nucleotide-binding protein subunit alpha
Phchr1|10122	1	40S ribosomal protein S23
Phchr1|10245	1	40S ribosomal protein S8
Phchr1|10264	1	40S ribosomal protein S5
Phchr1|10392	1	60S ribosomal protein L30-1
Phchr1|10648	1	NA
Phchr1|10865	1	Chaperone protein dnaJ
Phchr1|10890	1	10 kDa heat shock protein, mitochondrial
Phchr1|11273	1	40S ribosomal protein S6-B
Phchr1|11376	1	Histone H2B
Phchr1|11377	1	Histone H2A
Phchr1|11379	1	C-8 sterol isomerase
Phchr1|11388	1	60S ribosomal protein L28
Phchr1|11463	1	unknown
Phchr1|11514	1	Histone H2A
Phchr1|11515	1	Histone H2B
Phchr1|11766	1	60S ribosomal protein L17
Phchr1|12086	1	Meiotically up-regulated gene 158 protein
Phchr1|00243	2	unknown
Phchr1|00766	2	unknown
Phchr1|00827	2	Alpha-amylase 1
Phchr1|01202	2	unknown
Phchr1|01871	2	unknown
Phchr1|02423	2	unknown
Phchr1|02619	2	unknown
Phchr1|02748	2	Chitinase 1
Phchr1|02807	2	Carboxyvinyl-carboxyphosphonate phosphorylmutase
Phchr1|02849	2	unknown
Phchr1|02950	2	unknown
Phchr1|03012	2	Glucan 1,3-beta-glucosidase
Phchr1|03558	2	unknown
Phchr1|03574	2	unknown
Phchr1|03897	2	unknown
Phchr1|04168	2	unknown
Phchr1|04588	2	unknown
Phchr1|04738	2	unknown
Phchr1|04749	2	unknown
Phchr1|04847	2	unknown
Phchr1|04864	2	unknown
Phchr1|05130	2	Endothiapepsin

Continued on next page

Table 18: continued

Gene	Module Number	Annotation
Phchr1|05316	2	Trihydroxybenzophenone synthase
Phchr1|05318	2	Chalcone synthase G
Phchr1|05549	2	unknown
Phchr1|05816	2	Probable beta-glucosidase I
Phchr1|05879	2	Spherulin-1B
Phchr1|06351	2	unknown
Phchr1|06413	2	unknown
Phchr1|07450	2	unknown
Phchr1|07820	2	Lipase
Phchr1|07823	2	unknown
Phchr1|08035	2	unknown
Phchr1|08417	2	Acyl-CoA-binding protein
Phchr1|08509	2	Chitinase 2
Phchr1|08824	2	unknown
Phchr1|09281	2	unknown
Phchr1|09818	2	Aspergillopepsin-2
Phchr1|09978	2	unknown
Phchr1|10224	2	Probable glycosidase C21B10.07
Phchr1|10253	2	unknown
Phchr1|10594	2	unknown
Phchr1|11843	2	Thaumatin-like protein 2
Phchr1|12104	2	Polyporopepsin
Phchr1|12132	2	Polyporopepsin
Phchr1|12220	2	unknown
Phchr1|12733	2	unknown
Phchr1|12771	2	unknown
Phchr1|00240	3	unknown
Phchr1|00539	3	Pentachlorophenol 4-monooxygenase
Phchr1|00812	3	unknown
Phchr1|01054	3	18.1 kDa class I heat shock protein
Phchr1|01068	3	17.7 kDa class I heat shock protein
Phchr1|01470	3	Tripeptidyl aminopeptidase
Phchr1|02232	3	unknown
Phchr1|02429	3	unknown
Phchr1|02450	3	unknown
Phchr1|02601	3	unknown
Phchr1|02677	3	unknown
Phchr1|03318	3	unknown
Phchr1|04613	3	unknown
Phchr1|05120	3	unknown
Phchr1|05315	3	unknown
Phchr1|05860	3	unknown
Phchr1|07014	3	unknown
Phchr1|07017	3	unknown
Phchr1|07115	3	Calcium/calmodulin-dependent protein kinase type I
Phchr1|07118	3	unknown
Phchr1|07154	3	unknown
Phchr1|07420	3	Uncharacterized protein C17G6.02c
Phchr1|07432	3	Uncharacterized protein C17G6.02c
Phchr1|07888	3	unknown
Phchr1|07902	3	Hydroxymethylglutaryl-CoA synthase, cytoplasmic
Phchr1|08200	3	unknown
Phchr1|08202	3	unknown
Phchr1|08225	3	Ferric reductase transmembrane component 6
Phchr1|08422	3	unknown
Phchr1|08986	3	unknown
Phchr1|09449	3	unknown
Phchr1|10116	3	Diphosphomevalonate decarboxylase
Phchr1|10189	3	unknown
Phchr1|10254	3	unknown
Phchr1|10302	3	unknown

Continued on next page

Table 18: continued

Gene	Module Number	Annotation
Phchr1|10549	3	unknown
Phchr1|11154	3	Lathosterol oxidase
Phchr1|11270	3	unknown
Phchr1|12119	3	unknown
Phchr1|12218	3	unknown
Phchr1|12229	3	unknown
Phchr1|12244	3	unknown
Phchr1|12488	3	unknown
Phchr1|12842	3	unknown
Phchr1|12932	3	unknown
Phchr1|00028	4	unknown
Phchr1|00560	4	Dicarboxylic amino acid permease
Phchr1|00789	4	Probable sulfate permease C869.05c
Phchr1|00858	4	Tetracycline resistance protein, class H
Phchr1|00948	4	Probable E3 ubiquitin ligase complex SCF subunit sconB
Phchr1|01308	4	unknown
Phchr1|01317	4	O-acetylhomoserine (thiol)-lyase
Phchr1|01461	4	Alpha-ketoglutarate-dependent taurine dioxygenase
Phchr1|01806	4	Alpha-ketoglutarate-dependent sulfonate dioxygenase
Phchr1|01819	4	Putative carbonate dehydratase-like protein Rv1284
Phchr1|02353	4	unknown
Phchr1|02838	4	N amino acid transport system protein
Phchr1|02839	4	Alpha-ketoglutarate-dependent taurine dioxygenase
Phchr1|02992	4	Polyporopepsin
Phchr1|03041	4	Probable quinone oxidoreductase
Phchr1|03043	4	Probable quinone oxidoreductase
Phchr1|03225	4	Uncharacterized transporter YIL166C
Phchr1|04988	4	Uncharacterized transporter C1002.16c
Phchr1|05400	4	unknown
Phchr1|05519	4	Peroxiredoxin-6
Phchr1|05946	4	unknown
Phchr1|06101	4	unknown
Phchr1|07147	4	Aorsin
Phchr1|07153	4	unknown
Phchr1|07936	4	unknown
Phchr1|08232	4	unknown
Phchr1|08233	4	1-aminocyclopropane-1-carboxylate oxidase
Phchr1|08234	4	Pantothenate transporter liz1
Phchr1|08283	4	Zinc-binding alcohol dehydrogenase domain-containing protein cipB
Phchr1|08350	4	Alpha-ketoglutarate-dependent sulfonate dioxygenase
Phchr1|08983	4	Flavonol synthase/flavanone 3-hydroxylase
Phchr1|09283	4	Dehydrogenase/reductase SDR family member 2
Phchr1|09299	4	Cystathionine gamma-lyase
Phchr1|09309	4	unknown
Phchr1|09672	4	unknown
Phchr1|09686	4	unknown
Phchr1|09789	4	Alpha-ketoglutarate-dependent taurine dioxygenase
Phchr1|10589	4	unknown
Phchr1|10776	4	unknown
Phchr1|11294	4	High-affinity methionine permease
Phchr1|11486	4	unknown
Phchr1|11487	4	Uncharacterized protein ycaC
Phchr1|11500	4	Uncharacterized protein ycaC
Phchr1|12163	4	unknown
Phchr1|12930	4	Uncharacterized transporter YIL166C
Phchr1|00047	5	ATP synthase subunit beta, mitochondrial
Phchr1|00321	5	unknown
Phchr1|00579	5	Heat shock protein HSS1
Phchr1|00653	5	Uncharacterized MFS-type transporter C409.08
Phchr1|01182	5	Uncharacterized protein C13G6.15c
Phchr1|01568	5	unknown

Continued on next page

Table 18: continued

Gene	Module Number	Annotation
Phchr1|01967	5	unknown
Phchr1|02030	5	unknown
Phchr1|02185	5	ADP,ATP carrier protein
Phchr1|02194	5	unknown
Phchr1|02243	5	Chaperone protein ClpB
Phchr1|02304	5	Aspartate aminotransferase, mitochondrial
Phchr1|02406	5	unknown
Phchr1|02679	5	unknown
Phchr1|02823	5	Heat shock 70 kDa protein
Phchr1|02952	5	60S ribosomal protein L28-1
Phchr1|03376	5	unknown
Phchr1|03861	5	60S ribosomal protein L12
Phchr1|04185	5	Lanosterol 14-alpha demethylase
Phchr1|04193	5	60S acidic ribosomal protein P0
Phchr1|04663	5	unknown
Phchr1|04747	5	unknown
Phchr1|04859	5	UDP-glucose 4-epimerase
Phchr1|04950	5	unknown
Phchr1|04962	5	Peroxiredoxin 1
Phchr1|05066	5	Putative fungistatic metabolite
Phchr1|05166	5	Chitin synthase 1
Phchr1|05238	5	60S ribosomal protein L3
Phchr1|05495	5	UDP-glucuronic acid decarboxylase 1
Phchr1|05853	5	Uncharacterized protein C553.10
Phchr1|05856	5	Uncharacterized protein C553.10
Phchr1|06235	5	Siderophore iron transporter 3
Phchr1|06450	5	Alternative oxidase, mitochondrial
Phchr1|07506	5	unknown
Phchr1|07558	5	ATP synthase subunit alpha, mitochondrial
Phchr1|08004	5	unknown
Phchr1|08141	5	unknown
Phchr1|08246	5	unknown
Phchr1|08265	5	60S ribosomal protein L4-B
Phchr1|09038	5	Peptidyl-prolyl cis-trans isomerase D
Phchr1|09609	5	Iron transport multicopper oxidase fio1
Phchr1|09610	5	Plasma membrane iron permease
Phchr1|09674	5	3-hydroxy-3-methylglutaryl-coenzyme A reductase 2
Phchr1|10548	5	unknown
Phchr1|11114	5	Probable glucosamine–fructose-6-phosphate aminotransferase [isomerizing]
Phchr1|11158	5	Probable phosphoketolase
Phchr1|11272	5	40S ribosomal protein S13
Phchr1|11951	5	unknown
Phchr1|12397	5	unknown
Phchr1|12406	5	unknown
Phchr1|12521	5	unknown
Phchr1|01973	6	unknown
Phchr1|02322	6	G1/S-specific cyclin CLN1
Phchr1|02323	6	unknown
Phchr1|02432	6	Sorbose reductase sou1
Phchr1|02701	6	unknown
Phchr1|04558	6	Proteasome subunit beta type-1
Phchr1|04610	6	Protein VTS1
Phchr1|05308	6	Iron sulfur cluster assembly protein 1, mitochondrial
Phchr1|05503	6	unknown
Phchr1|06685	6	unknown
Phchr1|06982	6	unknown
Phchr1|07069	6	unknown
Phchr1|07444	6	unknown
Phchr1|07758	6	3-dehydroquinate synthase
Phchr1|07927	6	unknown
Phchr1|08579	6	Aldehyde reductase 1

Continued on next page

Table 18: continued

Gene	Module Number	Annotation
Phchr1|09185	6	Sexual differentiation process protein isp4
Phchr1|09264	6	unknown
Phchr1|09559	6	ATP-dependent permease PDR12
Phchr1|09859	6	unknown
Phchr1|09904	6	unknown
Phchr1|10403	6	Cysteine proteinase 1, mitochondrial
Phchr1|10791	6	Alcohol oxidase
Phchr1|11069	6	Negative regulator of sexual conjugation and meiosis
Phchr1|11360	6	Sorbitol dehydrogenase
Phchr1|11523	6	unknown
Phchr1|12016	6	Elongation factor 3
Phchr1|12227	6	Glycerol dehydrogenase
Phchr1|12230	6	unknown
Phchr1|12455	6	unknown
Phchr1|12802	6	unknown
Phchr1|00058	7	unknown
Phchr1|00641	7	Oligopeptide transporter 3
Phchr1|00642	7	Oligopeptide transporter 7
Phchr1|01000	7	OV-16 antigen
Phchr1|01424	7	unknown
Phchr1|01425	7	unknown
Phchr1|01927	7	Chitin deacetylase
Phchr1|02053	7	unknown
Phchr1|02221	7	unknown
Phchr1|02334	7	unknown
Phchr1|02336	7	UPF0654 protein C11D3.01c
Phchr1|02463	7	Probable urea active transporter 1
Phchr1|02581	7	unknown
Phchr1|02719	7	unknown
Phchr1|02752	7	Succinate dehydrogenase [ubiquinone] iron-sulfur subunit, mitochondrial
Phchr1|02771	7	Zinc-type alcohol dehydrogenase-like protein PB24D3.08c
Phchr1|02792	7	unknown
Phchr1|04046	7	Tripeptidyl-peptidase SED2
Phchr1|04141	7	Choline dehydrogenase
Phchr1|04142	7	unknown
Phchr1|04806	7	Putative dioxygenase C576.01c
Phchr1|04824	7	Histone H3.2
Phchr1|05208	7	unknown
Phchr1|05897	7	unknown
Phchr1|06343	7	unknown
Phchr1|06371	7	unknown
Phchr1|06544	7	Lactoylglutathione lyase
Phchr1|06564	7	Lysine-specific permease
Phchr1|06760	7	Purine permease
Phchr1|06777	7	unknown
Phchr1|07124	7	unknown
Phchr1|07214	7	Conidiation-specific protein 6
Phchr1|07445	7	Succinate dehydrogenase [ubiquinone] cytochrome b small subunit, mitochondrial
Phchr1|07780	7	Pyranose 2-oxidase
Phchr1|07808	7	Protein AIM2
Phchr1|08454	7	Uncharacterized permease C1683.05
Phchr1|08520	7	Oleate-induced peroxisomal protein POX18
Phchr1|10299	7	unknown
Phchr1|10726	7	Putative fumarate reductase
Phchr1|10771	7	unknown
Phchr1|10798	7	unknown
Phchr1|10907	7	Protein FDD123
Phchr1|11143	7	unknown
Phchr1|11163	7	unknown
Phchr1|11648	7	Sexual differentiation process protein isp4
Phchr1|11839	7	Altered inheritance rate of mitochondria protein 38

Continued on next page

Table 18: continued

Gene	Module Number	Annotation
Phchr1|11856	7	unknown
Phchr1|12085	7	unknown
Phchr1|12231	7	Aconitate hydratase, mitochondrial
Phchr1|12647	7	Sexual differentiation process protein isp4
Phchr1|12814	7	Zinc-type alcohol dehydrogenase-like protein C1773.06c
Phchr1|12843	7	unknown
Phchr1|00620	8	Uncharacterized GST-like protein yfcG
Phchr1|00743	8	unknown
Phchr1|01227	8	unknown
Phchr1|01617	8	Putative peroxiredoxin (Fragment)
Phchr1|02190	8	Probable glutamine amidotransferase DUG3
Phchr1|02935	8	unknown
Phchr1|03136	8	unknown
Phchr1|03146	8	Glutaredoxin-C1
Phchr1|04361	8	unknown
Phchr1|04384	8	Protein FDD123
Phchr1|05094	8	Beta-1,3-glucan-binding protein
Phchr1|05426	8	Uncharacterized amino-acid permease C794.03
Phchr1|06155	8	Probable glucan 1,3-beta-glucosidase D
Phchr1|06353	8	Epoxide hydrolase 1
Phchr1|07379	8	unknown
Phchr1|07393	8	unknown
Phchr1|07535	8	Uncharacterized 21.2 kDa protein
Phchr1|07730	8	Oligopeptide transporter 6
Phchr1|08138	8	unknown
Phchr1|09811	8	Aspergillopepsin-2
Phchr1|10456	8	unknown
Phchr1|10731	8	unknown
Phchr1|11427	8	NADP-dependent malic enzyme
Phchr1|11490	8	Uncharacterized protein ycaC
Phchr1|11768	8	Oligo-1,6-glucosidase
Phchr1|12126	8	Phosphatase yfbT
Phchr1|12458	8	unknown
Phchr1|00076	9	unknown
Phchr1|00195	9	Small COPII coat GTPase SAR1
Phchr1|00320	9	Syntaxin-like protein psy1
Phchr1|00333	9	unknown
Phchr1|00369	9	unknown
Phchr1|00502	9	GTP-binding nuclear protein spi1
Phchr1|00576	9	Plasma membrane proteolipid 3
Phchr1|00589	9	60S ribosomal protein L26-2
Phchr1|00593	9	Adenylate kinase 2
Phchr1|00780	9	Cytochrome b-c1 complex subunit 7
Phchr1|00814	9	Multiprotein-bridging factor 1
Phchr1|01174	9	CBM21 domain-containing protein CG9619
Phchr1|01217	9	Inositol polyphosphate multikinase
Phchr1|01824	9	Probable Ras-related protein Rab7
Phchr1|01888	9	40S ribosomal protein S27
Phchr1|02109	9	Putative acyl carrier protein, mitochondrial
Phchr1|02263	9	40S ribosomal protein S24-1
Phchr1|02280	9	40S ribosomal protein S3-A
Phchr1|02411	9	unknown
Phchr1|02584	9	unknown
Phchr1|03435	9	Protein wos2
Phchr1|03463	9	40S ribosomal protein S29
Phchr1|03469	9	Cytochrome P450 61
Phchr1|03631	9	60S ribosomal protein L19-3
Phchr1|03641	9	Protein YOP1
Phchr1|03652	9	unknown
Phchr1|04260	9	Cytochrome c oxidase subunit 4, mitochondrial
Phchr1|04269	9	unknown

Continued on next page

Table 18: continued

Gene	Module Number	Annotation
Phchr1|04437	9	unknown
Phchr1|04676	9	Protein vip1
Phchr1|04721	9	unknown
Phchr1|05209	9	Isocitrate dehydrogenase [NAD] subunit 2, mitochondrial
Phchr1|05403	9	Vacuolar aspartic protease
Phchr1|05417	9	unknown
Phchr1|05459	9	40S ribosomal protein S25-A
Phchr1|05469	9	60S ribosomal protein L18-B
Phchr1|05475	9	ATP synthase subunit d, mitochondrial
Phchr1|05678	9	14-3-3 protein homolog
Phchr1|06051	9	60S ribosomal protein L35-3
Phchr1|06144	9	Actin-1
Phchr1|06458	9	Delta(12) fatty acid desaturase
Phchr1|06734	9	Microsomal glutathione S-transferase 3
Phchr1|07028	9	Protein mago nashi homolog
Phchr1|07100	9	60S acidic ribosomal protein P1-alpha 1
Phchr1|07207	9	unknown
Phchr1|07495	9	Histone H2A
Phchr1|07655	9	unknown
Phchr1|07766	9	UPF0357 protein C1687.07
Phchr1|08597	9	Programmed cell death protein 6
Phchr1|08839	9	Serine hydroxymethyltransferase, cytosolic
Phchr1|09510	9	unknown
Phchr1|09585	9	unknown
Phchr1|09639	9	40S ribosomal protein S26
Phchr1|09833	9	unknown
Phchr1|09910	9	GTP-binding protein rhoA
Phchr1|10233	9	ADP-ribosylation factor 6
Phchr1|10326	9	Prenylated Rab acceptor 1
Phchr1|10520	9	unknown
Phchr1|10827	9	ATP synthase subunit g, mitochondrial
Phchr1|10880	9	40S ribosomal protein S20
Phchr1|10940	9	Cytochrome c oxidase subunit 6B
Phchr1|11016	9	60S ribosomal protein L10
Phchr1|11135	9	Nascent polypeptide-associated complex subunit beta
Phchr1|11409	9	Ubiquitin-conjugating enzyme E2-16 kDa
Phchr1|11545	9	NA
Phchr1|11546	9	Plasma membrane ATPase 3
Phchr1|11738	9	40S ribosomal protein S27
Phchr1|11871	9	Histone H3
Phchr1|11916	9	unknown
Phchr1|11925	9	Thioredoxin-1
Phchr1|12073	9	unknown
Phchr1|12076	9	Cofilin
Phchr1|12405	9	unknown
Phchr1|12494	9	Pre-mRNA-splicing factor srp1
Phchr1|12872	9	Probable elongation factor 1-beta
Phchr1|00555	10	Protein FDD123
Phchr1|01095	10	Glutathione reductase
Phchr1|02018	10	Bifunctional nitrilase/nitrile hydratase NIT4
Phchr1|02780	10	unknown
Phchr1|03587	10	Homoserine O-acetyltransferase
Phchr1|04992	10	Uncharacterized protein C4H3.07c
Phchr1|05138	10	Probable aspartate-semialdehyde dehydrogenase
Phchr1|06094	10	unknown
Phchr1|07839	10	unknown
Phchr1|07994	10	unknown
Phchr1|09678	10	Probable aspartokinase
Phchr1|10076	10	unknown
Phchr1|10120	10	unknown
Phchr1|11382	10	Probable homoserine dehydrogenase

Continued on next page

Table 18: continued

Gene	Module Number	Annotation
Phchr1|11423	10	unknown
Phchr1|12215	10	Acetyl-coenzyme A synthetase
Phchr1|00626	11	Exoglucanase 3
Phchr1|00718	11	unknown
Phchr1|01140	11	Sodium/potassium-transporting ATPase subunit alpha-4
Phchr1|02102	11	Probable mannan endo-1,4-beta-mannosidase C
Phchr1|02642	11	unknown
Phchr1|04401	11	High-affinity glucose transporter
Phchr1|07536	11	Exoglucanase 1
Phchr1|07690	11	High-affinity glucose transporter SNF3
Phchr1|08445	11	Diacetyl reductase [(S)-acetoin forming]
Phchr1|08641	11	unknown
Phchr1|08642	11	Probable 1,4-beta-D-glucan cellobiohydrolase C
Phchr1|08751	11	unknown
Phchr1|09040	11	unknown
Phchr1|09318	11	unknown
Phchr1|09454	11	unknown
Phchr1|11106	11	unknown
Phchr1|11751	11	unknown
Phchr1|12695	11	unknown
Phchr1|00595	12	unknown
Phchr1|00759	12	Autophagy-related protein 8
Phchr1|01152	12	Cell division control protein 42 homolog
Phchr1|01419	12	Extracellular metalloproteinase 4
Phchr1|01566	12	Myosin regulatory light chain cdc4
Phchr1|01816	12	unknown
Phchr1|01902	12	unknown
Phchr1|02094	12	Phosphatidylglycerol/phosphatidylinositol transfer protein
Phchr1|03152	12	unknown
Phchr1|03211	12	unknown
Phchr1|03436	12	unknown
Phchr1|04750	12	unknown
Phchr1|07181	12	3-ketoacyl-CoA thiolase, peroxisomal
Phchr1|07489	12	Psi-producing oxygenase C
Phchr1|10099	12	unknown
Phchr1|10932	12	Fructose-bisphosphate aldolase
Phchr1|00099	13	S-adenosylmethionine synthase
Phchr1|00447	13	unknown
Phchr1|00458	13	NA
Phchr1|02260	13	Protein priA
Phchr1|03694	13	O-methyltransferase mdmC
Phchr1|05918	13	5-methyltetrahydropteroyltriglutamate–homocysteine methyltransferase
Phchr1|06325	13	Putative sterigmatocystin biosynthesis peroxidase stcC
Phchr1|07320	13	Zinc-regulated transporter 1
Phchr1|09505	13	unknown
Phchr1|10706	13	unknown
Phchr1|12155	13	unknown
Phchr1|12421	13	Alpha-amylase mde5
Phchr1|12905	13	Uncharacterized bolA-like protein C8C9.11
Phchr1|00752	14	unknown
Phchr1|00768	14	unknown
Phchr1|01096	14	Manganese peroxidase H3
Phchr1|04126	14	Manganese peroxidase 1
Phchr1|04129	14	Manganese peroxidase 1
Phchr1|04494	14	Aspartic protease
Phchr1|05059	14	Ligninase LG5
Phchr1|08551	14	unknown
Phchr1|09298	14	Manganese peroxidase H4
Phchr1|11305	14	Manganese peroxidase H4
Phchr1|00166	15	E3 ubiquitin ligase complex SCF subunit sconC
Phchr1|00387	15	unknown

Continued on next page

Table 18: continued

Gene	Module Number	Annotation
Phchr1|00434	15	Protein translation factor sui1
Phchr1|00446	15	Protein FDD123
Phchr1|00547	15	unknown
Phchr1|00700	15	Pyruvate dehydrogenase E1 component subunit alpha, mitochondrial
Phchr1|00705	15	unknown
Phchr1|00778	15	Probable transketolase
Phchr1|00781	15	Nucleoside diphosphate kinase
Phchr1|00859	15	unknown
Phchr1|00883	15	Phosphoglycerate kinase
Phchr1|01101	15	Nuclear transport factor 2
Phchr1|01498	15	unknown
Phchr1|01832	15	6-phosphogluconate dehydrogenase, decarboxylating
Phchr1|02096	15	unknown
Phchr1|02182	15	unknown
Phchr1|02248	15	unknown
Phchr1|02249	15	Elongation of fatty acids protein 2
Phchr1|02290	15	Elongation factor 1-alpha
Phchr1|02615	15	unknown
Phchr1|02639	15	unknown
Phchr1|02687	15	Calmodulin
Phchr1|02726	15	Profilin-1B
Phchr1|02934	15	unknown
Phchr1|02944	15	Enolase (Fragment)
Phchr1|02969	15	Thioredoxin
Phchr1|03115	15	unknown
Phchr1|03261	15	Inorganic pyrophosphatase
Phchr1|03289	15	Elongation factor 2
Phchr1|03730	15	Accumulation of dyads protein 2
Phchr1|04171	15	unknown
Phchr1|04259	15	unknown
Phchr1|04387	15	Protein FDD123
Phchr1|04438	15	unknown
Phchr1|04630	15	NAD-specific glutamate dehydrogenase
Phchr1|05054	15	unknown
Phchr1|05488	15	unknown
Phchr1|05502	15	unknown
Phchr1|05580	15	unknown
Phchr1|05582	15	unknown
Phchr1|05824	15	unknown
Phchr1|05876	15	Glutamine synthetase
Phchr1|05928	15	Altered inheritance rate of mitochondria protein 38
Phchr1|06150	15	Probable malate dehydrogenase, mitochondrial
Phchr1|06208	15	unknown
Phchr1|06648	15	Translationally-controlled tumor protein homolog
Phchr1|06745	15	Protein disulfide-isomerase
Phchr1|07143	15	Small ubiquitin-related modifier 1
Phchr1|07369	15	Phosphate carrier protein, mitochondrial
Phchr1|07555	15	Tropomyosin-2
Phchr1|07706	15	unknown
Phchr1|07716	15	unknown
Phchr1|07836	15	Transitional endoplasmic reticulum ATPase
Phchr1|08263	15	unknown
Phchr1|08488	15	Eukaryotic translation initiation factor 1A, X-chromosomal
Phchr1|08647	15	Inositol-3-phosphate synthase
Phchr1|08756	15	unknown
Phchr1|08829	15	L-threonine 3-dehydrogenase
Phchr1|09280	15	ATP-citrate synthase
Phchr1|09317	15	unknown
Phchr1|09433	15	Peptidyl-prolyl cis-trans isomerase
Phchr1|09528	15	unknown
Phchr1|09532	15	unknown

Continued on next page

Table 18: continued

Gene	Module Number	Annotation
Phchr1|09599	15	Adenosylhomocysteinase
Phchr1|09602	15	unknown
Phchr1|09896	15	Transcription factor prr1
Phchr1|09991	15	Heat shock protein homolog SSE1
Phchr1|10011	15	Chitinase A1
Phchr1|10241	15	Probable UTP–glucose-1-phosphate uridylyltransferase
Phchr1|10262	15	unknown
Phchr1|10284	15	ATP synthase subunit delta, mitochondrial
Phchr1|10593	15	Serine proteinase inhibitor IA-1
Phchr1|10612	15	unknown
Phchr1|10814	15	Probable aspartic-type endopeptidase CTSD
Phchr1|10822	15	Cystathionine beta-lyase
Phchr1|10944	15	Citrate synthase, mitochondrial
Phchr1|11330	15	unknown
Phchr1|11343	15	mRNA export protein mlo3
Phchr1|11417	15	C2 domain-containing protein C31G5.15
Phchr1|11694	15	unknown
Phchr1|11711	15	unknown
Phchr1|11828	15	unknown
Phchr1|11829	15	unknown
Phchr1|11905	15	Heat shock protein 83
Phchr1|12017	15	Mitochondrial outer membrane protein porin
Phchr1|12064	15	unknown
Phchr1|12094	15	Pyruvate decarboxylase
Phchr1|12438	15	unknown
Phchr1|12440	15	unknown
Phchr1|12520	15	Glyceraldehyde-3-phosphate dehydrogenase
Phchr1|12573	15	78 kDa glucose-regulated protein homolog
Phchr1|12633	15	unknown
Phchr1|12651	15	Eukaryotic translation initiation factor 5A
Phchr1|12775	15	Elongation factor 1-gamma 3
Phchr1|12832	15	unknown
Phchr1|12865	15	unknown
Phchr1|12871	15	unknown
Phchr1|00364	17	GTP-binding protein ypt1
Phchr1|01372	17	Histone H4
Phchr1|03258	17	Histone H4
Phchr1|03972	17	unknown
Phchr1|04598	17	unknown
Phchr1|05235	17	unknown
Phchr1|05721	17	Uncharacterized protein C11D3.13
Phchr1|06504	17	unknown
Phchr1|07429	17	Heat shock protein 16
Phchr1|07986	17	unknown
Phchr1|08301	17	Cytochrome c
Phchr1|08353	17	unknown
Phchr1|09086	17	NA
Phchr1|09087	17	NA
Phchr1|09312	17	Ammonium transporter 1
Phchr1|11209	17	unknown
Phchr1|11663	17	unknown
Phchr1|11730	17	unknown
Phchr1|11790	17	unknown
Phchr1|12898	17	unknown
Phchr1|00086	18	unknown
Phchr1|00646	18	unknown
Phchr1|01605	18	unknown
Phchr1|01829	18	Nucleolysin TIAR
Phchr1|01873	18	Protein yippee-like At3g08990
Phchr1|02035	18	unknown
Phchr1|02121	18	Uncharacterized protein YKR043C

Continued on next page

Table 18: continued

Gene	Module Number	Annotation
Phchr1|02742	18	Uncharacterized protein C4H3.03c
Phchr1|02743	18	Uncharacterized protein C4H3.03c
Phchr1|02811	18	unknown
Phchr1|03090	18	Putative voltage-gated potassium channel subunit beta
Phchr1|03170	18	unknown
Phchr1|03914	18	29 kDa ribonucleoprotein, chloroplastic
Phchr1|03985	18	Glucan 1,3-beta-glucosidase
Phchr1|04048	18	Polyubiquitin
Phchr1|04308	18	Peroxiredoxin HYR1
Phchr1|04379	18	Glutathione S-transferase 2
Phchr1|05127	18	NEDD8
Phchr1|06095	18	O-methylsterigmatocystin oxidoreductase
Phchr1|06363	18	unknown
Phchr1|06503	18	unknown
Phchr1|07860	18	unknown
Phchr1|07954	18	unknown
Phchr1|08660	18	Subtilisin-like serine protease pepC
Phchr1|08870	18	Uncharacterized protein C32A11.02c
Phchr1|08887	18	Putative sterigmatocystin biosynthesis peroxidase stcC
Phchr1|09021	18	Protein rds1
Phchr1|09158	18	L-threonine 3-dehydrogenase
Phchr1|09178	18	Uncharacterized protein C32A11.02c
Phchr1|09754	18	Ubiquitin-conjugating enzyme E2 2
Phchr1|10026	18	unknown
Phchr1|10439	18	unknown
Phchr1|10564	18	unknown
Phchr1|10702	18	unknown
Phchr1|11228	18	unknown
Phchr1|11319	18	Trehalose-phosphatase
Phchr1|11410	18	unknown
Phchr1|12032	18	O-acetylhomoserine (thiol)-lyase
Phchr1|12123	18	unknown
Phchr1|12258	18	unknown
Phchr1|12437	18	unknown
Phchr1|12538	18	unknown
Phchr1|00024	19	unknown
Phchr1|00732	19	Heat shock protein sks2
Phchr1|01320	19	C-4 methylsterol oxidase
Phchr1|01565	19	unknown
Phchr1|02110	19	60S ribosomal protein L16
Phchr1|02407	19	Probable serine hydrolase C5E4.05c
Phchr1|04296	19	unknown
Phchr1|04633	19	Mitochondrial-processing peptidase subunit beta
Phchr1|04765	19	Cytochrome c oxidase subunit 6, mitochondrial
Phchr1|05546	19	Acetyl-CoA acetyltransferase, mitochondrial
Phchr1|05871	19	ADP-ribosylation factor
Phchr1|06390	19	unknown
Phchr1|07818	19	unknown
Phchr1|08119	19	unknown
Phchr1|09645	19	60S ribosomal protein L44
Phchr1|10311	19	Guanine nucleotide-binding protein subunit beta-2-like 1
Phchr1|10475	19	ATP synthase subunit 5, mitochondrial
Phchr1|10601	19	unknown
Phchr1|10656	19	Uncharacterized membrane protein C576.04
Phchr1|11138	19	Sterol 24-C-methyltransferase
Phchr1|11742	19	Multidrug resistance protein 3
Phchr1|11846	19	Mitochondrial protein import protein mas5
Phchr1|12970	19	GTP-binding protein rho2

Table 18: *Phanerochaete chrysosporium* gene module assignment

No	Gene	Annotation
1	Phchr1|00028	Unknown
2	Phchr1|00560	Dicarboxylic amino acid permease
3	Phchr1|00789	Probable sulfate permease C869.05c
4	Phchr1|00858	Tetracycline resistance protein, class H
5	Phchr1|00948	Probable E3 ubiquitin ligase complex SCF subunit sconB
6	Phchr1|01308	Unknown
7	Phchr1|01317	O-acetylhomoserine (thiol)-lyase
8	Phchr1|01461	Alpha-ketoglutarate-dependent taurine dioxygenase
9	Phchr1|01806	Alpha-ketoglutarate-dependent sulfonate dioxygenase
10	Phchr1|01819	Putative carbonate dehydratase-like protein Rv1284
11	Phchr1|02353	Unknown
12	Phchr1|02838	N amino acid transport system protein
13	Phchr1|02839	Alpha-ketoglutarate-dependent taurine dioxygenase
14	Phchr1|02992	Polyporopepsin
15	Phchr1|03041	Probable quinone oxidoreductase
16	Phchr1|03043	Probable quinone oxidoreductase
17	Phchr1|03225	Uncharacterized transporter YIL166C
18	Phchr1|04988	Uncharacterized transporter C1002.16c
19	Phchr1|05400	Unknown
20	Phchr1|05519	Peroxiredoxin-6
21	Phchr1|05946	Unknown
22	Phchr1|06101	Unknown
23	Phchr1|07147	Aorsin
24	Phchr1|07153	Unknown
25	Phchr1|07936	Unknown
26	Phchr1|08232	Unknown
27	Phchr1|08233	1-aminocyclopropane-1-carboxylate oxidase
28	Phchr1|08234	Pantothenate transporter liz1
29	Phchr1|08283	Zinc-binding alcohol dehydrogenase domain-containing protein cipB
30	Phchr1|08350	Alpha-ketoglutarate-dependent sulfonate dioxygenase
31	Phchr1|08983	Flavonol synthase/flavanone 3-hydroxylase
32	Phchr1|09283	Dehydrogenase/reductase SDR family member 2
33	Phchr1|09299	Cystathionine gamma-lyase
34	Phchr1|09309	Unknown
35	Phchr1|09672	Unknown
36	Phchr1|09686	Unknown
37	Phchr1|09789	Alpha-ketoglutarate-dependent taurine dioxygenase
38	Phchr1|10589	Unknown
39	Phchr1|10776	Unknown
40	Phchr1|11294	High-affinity methionine permease
41	Phchr1|11486	Unknown
42	Phchr1|11487	Uncharacterized protein ycaC
43	Phchr1|11500	Uncharacterized protein ycaC
44	Phchr1|12163	Unknown
45	Phchr1|12930	Uncharacterized transporter YIL166C

Table 19: Annotation of genes in module 4 in the *Phanerochaete chrysosporium* dataset

No	Gene	Annotation
1	Phchr1|00058	Unknown
2	Phchr1|00641	Oligopeptide transporter 3
3	Phchr1|00642	Oligopeptide transporter 7
4	Phchr1|01000	OV-16 antigen
5	Phchr1|01424	Unknown
6	Phchr1|01425	Unknown
7	Phchr1|01927	Chitin deacetylase
8	Phchr1|02053	Unknown
9	Phchr1|02221	Unknown
10	Phchr1|02334	Unknown

Continued on next page

Table 20: continued

No	Gene	Annotation
11	Phchr1|02336	UPF0654 protein C11D3.01c
12	Phchr1|02463	Probable urea active transporter 1
13	Phchr1|02581	Unknown
14	Phchr1|02719	Unknown
15	Phchr1|02752	Succinate dehydrogenase [ubiquinone] iron-sulfur subunit, mitochondrial
16	Phchr1|02771	Zinc-type alcohol dehydrogenase-like protein PB24D3.08c
17	Phchr1|02792	Unknown
18	Phchr1|04046	Tripeptidyl-peptidase SED2
19	Phchr1|04141	Choline dehydrogenase
20	Phchr1|04142	Unknown
21	Phchr1|04806	Putative dioxygenase C576.01c
22	Phchr1|04824	Histone H3.2
23	Phchr1|05208	Unknown
24	Phchr1|05897	Unknown
25	Phchr1|06343	Unknown
26	Phchr1|06371	Unknown
27	Phchr1|06544	Lactoylglutathione lyase
28	Phchr1|06564	Lysine-specific permease
29	Phchr1|06760	Purine permease
30	Phchr1|06777	Unknown
31	Phchr1|07124	Unknown
32	Phchr1|07214	Conidiation-specific protein 6
33	Phchr1|07445	Succinate dehydrogenase [ubiquinone] cytochrome b small subunit, mitochondrial
34	Phchr1|07780	Pyranose 2-oxidase
35	Phchr1|07808	Protein AIM2
36	Phchr1|08454	Uncharacterized permease C1683.05
37	Phchr1|08520	Oleate-induced peroxisomal protein POX18
38	Phchr1|10299	Unknown
39	Phchr1|10726	Putative fumarate reductase
40	Phchr1|10771	Unknown
41	Phchr1|10798	Unknown
42	Phchr1|10907	Protein FDD123
43	Phchr1|11143	Unknown
44	Phchr1|11163	Unknown
45	Phchr1|11648	Sexual differentiation process protein isp4
46	Phchr1|11839	Altered inheritance rate of mitochondria protein 38
47	Phchr1|11856	Unknown
48	Phchr1|12085	Unknown
49	Phchr1|12231	Aconitate hydratase, mitochondrial
50	Phchr1|12647	Sexual differentiation process protein isp4
51	Phchr1|12814	Zinc-type alcohol dehydrogenase-like protein C1773.06c
52	Phchr1|12843	Unknown

Table 20: Annotation of genes in module 7 in the *Phanerochaete chrysosporium* dataset

No	Gene	Annotation
1	Phchr1|00076	Unknown
2	Phchr1|00195	Small COPII coat GTPase SAR1
3	Phchr1|00320	Syntaxin-like protein psy1
4	Phchr1|00333	Unknown
5	Phchr1|00369	Unknown
6	Phchr1|00502	GTP-binding nuclear protein spi1
7	Phchr1|00576	Plasma membrane proteolipid 3
8	Phchr1|00589	60S ribosomal protein L26-2
9	Phchr1|00593	Adenylate kinase 2
10	Phchr1|00780	Cytochrome b-c1 complex subunit 7
11	Phchr1|00814	Multiprotein-bridging factor 1
12	Phchr1|01174	CBM21 domain-containing protein CG9619

Continued on next page

Table 21: continued

No	Gene	Annotation
13	Phchr1|01217	Inositol polyphosphate multikinase
14	Phchr1|01824	Probable Ras-related protein Rab7
15	Phchr1|01888	40S ribosomal protein S27
16	Phchr1|02109	Putative acyl carrier protein, mitochondrial
17	Phchr1|02263	40S ribosomal protein S24-1
18	Phchr1|02280	40S ribosomal protein S3-A
19	Phchr1|02411	Unknown
20	Phchr1|02584	Unknown
21	Phchr1|03435	Protein wos2
22	Phchr1|03463	40S ribosomal protein S29
23	Phchr1|03469	Cytochrome P450 61
24	Phchr1|03631	60S ribosomal protein L19-3
25	Phchr1|03641	Protein YOP1
26	Phchr1|03652	Unknown
27	Phchr1|04260	Cytochrome c oxidase subunit 4, mitochondrial
28	Phchr1|04269	Unknown
29	Phchr1|04437	Unknown
30	Phchr1|04676	Protein vip1
31	Phchr1|04721	Unknown
32	Phchr1|05209	Isocitrate dehydrogenase [NAD] subunit 2, mitochondrial
33	Phchr1|05403	Vacuolar aspartic protease
34	Phchr1|05417	Unknown
35	Phchr1|05459	40S ribosomal protein S25-A
36	Phchr1|05469	60S ribosomal protein L18-B
37	Phchr1|05475	ATP synthase subunit d, mitochondrial
38	Phchr1|05678	14-3-3 protein homolog
39	Phchr1|06051	60S ribosomal protein L35-3
40	Phchr1|06144	Actin-1
41	Phchr1|06458	Delta(12) fatty acid desaturase
42	Phchr1|06734	Microsomal glutathione S-transferase 3
43	Phchr1|07028	Protein mago nashi homolog
44	Phchr1|07100	60S acidic ribosomal protein P1-alpha 1
45	Phchr1|07207	Unknown
46	Phchr1|07495	Histone H2A
47	Phchr1|07655	Unknown
48	Phchr1|07766	UPF0357 protein C1687.07
49	Phchr1|08597	Programmed cell death protein 6
50	Phchr1|08839	Serine hydroxymethyltransferase, cytosolic
51	Phchr1|09510	Unknown
52	Phchr1|09585	Unknown
53	Phchr1|09639	40S ribosomal protein S26
54	Phchr1|09833	Unknown
55	Phchr1|09910	GTP-binding protein rhoA
56	Phchr1|10233	ADP-ribosylation factor 6
57	Phchr1|10326	Prenylated Rab acceptor 1
58	Phchr1|10520	Unknown
59	Phchr1|10827	ATP synthase subunit g, mitochondrial
60	Phchr1|10880	40S ribosomal protein S20
61	Phchr1|10940	Cytochrome c oxidase subunit 6B
62	Phchr1|11016	60S ribosomal protein L10
63	Phchr1|11135	Nascent polypeptide-associated complex subunit beta
64	Phchr1|11409	Ubiquitin-conjugating enzyme E2-16 kDa
65	Phchr1|11545	Unknown
66	Phchr1|11546	Plasma membrane ATPase 3
67	Phchr1|11738	40S ribosomal protein S27
68	Phchr1|11871	Histone H3
69	Phchr1|11916	Unknown
70	Phchr1|11925	Thioredoxin-1
71	Phchr1|12073	Unknown
72	Phchr1|12076	Cofilin
73	Phchr1|12405	Unknown

Continued on next page

Table 21: continued

No	Gene	Annotation
74	Phchr1|12494	Pre-mRNA-splicing factor srp1
75	Phchr1|12872	Probable elongation factor 1-beta

Table 21: Annotation of genes in module 9 in the *Phanerochaete chrysosporium* dataset