Constructing Regulatory Networks from Gene Expression Data Using Association Rule Mining

by

Jiaye Zhou

B.S., Biochemistry and Molecular Biology, University of New Mexico, 1998

M.S., Computer Science, University of New Mexico, 2002

Abstract

Gene expression technology, such as high density DNA Microarray, allows us to monitor gene expression patterns at the genomic level. While this technology promises progress toward the understanding of transcriptional response and regulation, it also introduces the challenge of extracting relevant information from the large datasets generated. As a result, data mining of gene expression data has become an important area of research for biologists. Two classical data mining methods: data classification and clustering have been widely used to analyze gene expression data. These methods are valuable exploratory tools in data mining, but they are limited to placing genes into groups with others that share certain characteristics. While it is important to determine which genes are related, we also need to understand the mechanism of how genes relate and how they regulate one another. These two methods do not provide such insight. This

information, however, can be extracted from gene expression data with experiments that are well designed. In this thesis, I consider the use of association rule mining for discovering regulatory relationships among genes from gene expression data. Association rule mining, a relatively new technique in the area of data mining and knowledge discovery, has been the focus of much data mining research in computer science. Association rule mining is a process that identifies links between sets of correlated objects in large datasets. In this thesis, I describe the first application of association rule mining to gene expression data analysis. I develop a strategy for transforming gene expression data to make it suitable for association rule mining, and show how the FP-Growth algorithm for association rule mining can be adapted to apply to the transformed data. I implement, test and evaluate the association rule mining method using real gene expression data that is publicly available. I am able to validate our analysis method by showing that our results are consistent with published results generated by independent analysis methods. Furthermore, this method is able to generate previously unknown biological information, making it a valuable gene expression data analysis tool.

Table of Contents

List of Figures

List of Tables

Chapter 1: Background

1.1 Molecular Genetics

Cells are universal units of life. One of the first discoveries in biology was Robert Hooke's observation in 1665 that plant tissues were divided into tiny compartments, which he called *cellulae*, or **cells**. By 1840, Theodor Schwann proposed that all organisms exist as either a single cell or aggregates of cells. A century later, that hypothesis was confirmed. Inside the cells of organisms, chemical and biological processes take place as part of cell growth, differentiation, reproduction, and response to internal and external conditions. These chemical processes are tightly regulated to maintain an environment in which cells can function and thrive, called homeostasis. How this regulation is achieved is a fundamental question that needs to be answered in biological research and medicine. In this chapter, I will give a brief introduction to molecular biology so the readers have the necessary biology information for this thesis study.

1.1.1 Macromolecules

A living organism can be viewed as an information system. Outside of the organism, it communicates information with its environment. Inside of the organism, information is stored and communicated inter- and intra-cellularly. In a living cell, information is carried by two major classes of macromolecules: nucleic acids and proteins.

1.1.1.1Nucleic Acids

Nucleic acids are capable of storing information and directing the propagation of information by the synthesis of new copies of itself and proteins. Because they store the information that specifies the synthesis of cells' building blocks, they are often thought of as template of life. **DNA,** or deoxyribonucleic acid, is a type of nucleic acid. DNA is the central storage of information in a cell. All the information needed to direct a cell's activities are stored in DNA. DNA from all organisms is made up of the same chemical and physical components. DNA is in the form of double stranded helix, where the two strands are complementary of each other. Each strand is made up of repeating four bases: adenine (A), guanine (G), cytosine (C), and thymine (T). The complementarity of the two DNA strands comes from the specific pairing of A-T and G-C. This pairing property has been the basis of many technologies that are enhancing our understanding of genetics. The DNA sequence is the particular side-by-side non-random arrangement of bases along the DNA strand. (e.g., ATTCCGGA). The specific ordering of bases spells out the exact

instructions required to create a particular organism with its unique traits. There is a second type of nucleic acid, **RNA**, or ribonucleic acid. RNA shares great similarity with DNA in its chemical and physical properties. RNA has the same bases as DNA except that uracil (U) replaces thymine. RNA plays an essential role in the propagation of information inside of a cell.

The **genome** is an organism's complete set of DNA. Genomes vary widely in size, the smaller known genome for a free-living organism (a bacterium) contains about 600,000 DNA base pairs, while human and mouse genomes have some 3 billion. Except for mature red blood cells, all human cells contain a complete human genome. Almost all DNA in a cell is organized into large, physically separate molecules, called **chromosomes**. In the case of prokaryote[1] like E. *coli*, DNA is contained in a single, large circular DNA molecule, the prokaryotic chromosome. Genome in eukaryotes[2] is divided into several or many chromosomes, each of which contains a single, very large, linear DNA molecule. Although their sizes vary greatly among organisms and even among different chromosomes in a give species, these DNA molecules are commonly of the order of 10^7 to 10^9 base pairs in length. Different eukaryotic species contain widely varying number of *distinguishable* chromosomes, from 1 in an Australian ant to 190 in a butterfly species. The human genome consists of 23 distinct chromosomes that range in length from about 50 million to 250 million base pairs.

[1] Primitive single-celled organisms that are not compartmentalized by internal cellular membranes.
[2] Organisms whose cells are compartmentalized by internal cellular membranes to produce a nucleus and organelles.

Within the genome, the basic unit of hereditary information is a **gene**. A gene is made up of a specific sequence of DNA that encodes instructions on how to make proteins. Despite the size of the human genome, genes comprise only about 2% of the total DNA. The remainder consists of non-coding regions, whose functions may include providing structural integrity of chromosomes, and regulating where, when, and in what quantity proteins are made. The human genome is estimated to contain more than 30,000 genes.

1.1.1.2 Proteins

Although nucleic acids are the storage of information, it's the proteins that perform most life functions and even make up the majority of the cellular structures. They are the building blocks of organisms. Just as DNA and RNA are made up of sequences of repeating base monomers, proteins are large, complex molecules made up of sequence of subunits called amino acids. While there are 4 bases in DNA and RNA, chemical properties distinguish 20 different amino acids in protein. The uniqueness of DNA or RNA is primarily due to its sequence. Protein function is determined by the specific three dimensional structure caused by the folding of protein chains, in addition to the particular sequence of amino acids

Proteins play an enormous variety of roles. Some carry out the transport and storage of small molecules; others make up a large part of the structural framework of cells and tissues. Antibodies are proteins, and so are the blood clotting factors. Perhaps

the most important of all proteins are enzymes and regulatory factors that control reactions and essential pathways inside cells.

1.1.2 Information Metabolism

As mentioned previously, nucleic acids are capable of directing the propagation of information by the synthesis of new copies of themselves and the synthesis of proteins. In this section, I will discuss the preservation, retrieval, processing and transmission of biological information, which we call information metabolism.

Information stored in DNA can be inherited via a process called **DNA replication**, in which DNA serves as the template for its own synthesis. As mentioned before, DNA is stored as two complementary strands of molecules. While mechanically complex, conceptually the replication process is very simple. The replication process is based on the base pairing chemical property of DNA. The paired strands are unwound and separated, followed by the synthesis of two new complementary DNA strands simultaneously, giving rise to two daughter DNA molecules containing one strand each of parental and of newly synthesized material. Replication occurs at the genome level. Through this process, information is preserved.

The first step in information transfer is **transcription**, in which the information encoded in DNA specifies the structure of an RNA product. Mechanistically, transcription is similar to DNA replication. There are two major differences: only one

DNA template strand is transcribed, and only a small fraction of the entire genetic potential of an organism is realized in one cell. In a differentiated[3] eukaryotic cell, very little of the total DNA is transcribed. Even in a single-celled organism, in which virtually all of the DNA sequences can be transcribed, far fewer than half of all genes may be transcribed at any one time. Much of the concern with transcription in research involves the mechanisms by which particular genes and template strands for transcription are selected, because this selection in large part governs the metabolic capabilities of a cell. This process is highly specific, and often occurs as a response to particular chemical and physiological states in a cell. It is believed that this selection mechanism operates largely at the levels of initiation of transcription, through the actions of proteins that contact DNA in a highly site-specific manner. The product of transcription is RNA. There are several types of RNA. The RNA that is used as template for protein synthesis is called **messenger RNA** (mRNA). mRNA is very unstable, with a short life, and it constitutes a small proportion of total cellular RNA, ranging only 1% to 3% in bacteria. mRNA does not correspond to its DNA template exactly. It is actually a product of a post-transcriptional modification process in which initial RNA product is spliced and segments are joined to form the final mRNA.

Following transcription in information propagation is **translation**, in which mRNA serves as the template for synthesis of a particular protein. A sequence of mRNA (based on the complementary DNA template) is translated into another sequence of protein made up of the 20 amino acids. As mentioned before, protein function is not

[3] Differentiation is a process that, through rapid proliferation, embryonic cells become specialized type of cells that make up the tissues and organs of multi-cellular animals.

determined solely by its sequence, but also by its three dimensional structure. Therefore, translation is followed by post-translational modification of the protein and a folding process.

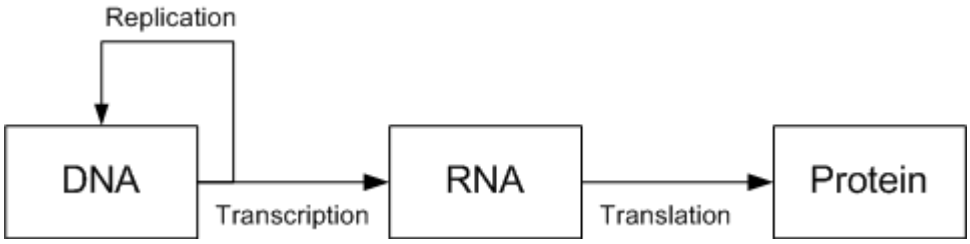

Figure 1.1 Flow of genetic information.

Figure 1.1 illustrates the flow of genetic information in a typical cell. It is the central dogma of molecular genetics. The transcription-translation process is also often called **gene expression**. Gene expression is a complex process that regulates other reactions and pathways inside a cell, which in turn is regulated itself.

1.1.3 Gene Expression and Microarray Technology

As discussed in the previous section, genomic data consists of DNA sequences made up of the same 4 bases (G, A, C, T) repeated non-randomly in strings of up to several million at a time. Despite the diversity observed within a species, the genome of a single species is nearly invariant. The human genome contains three billion bases in its

23 chromosomes, and is 99.99% invariant between individuals [14]. So what makes us different? We know that in a cell, not all genetic material is expressed at one time. While genetic variation exists, much of the diversity is accounted by the differential expression of the genetic information. So the selection of information to be processed has received great interest. Many technologies have been developed to study gene expression. Here, we will focus on a relatively new technology called DNA Microarray.

1.1.3.3 DNA Microarray

Traditional experimental methods in molecular biology are limited to studying only a few genes in one experiment. But genes and their products usually function in an orchestrated way. With the traditional methods, it is difficult to capture the entirety of what is going on inside a cell. In the last few years, a new technique called DNA Microarray promised the ability to monitor thousands of genes, even the entire genome, simultaneously.

Because gene expression starts with transcription, DNA Microarray technology focuses on the regulation at this step. The technology is based on existing, well-used molecular biology methods, Southern blotting[4] and Northern blotting[5]. Conceptually, DNA Microarray is very simple. It takes advantage of the specific pairing property of

[4] A technique for detecting the presence of a specific DNA sequence in a genome: The DNA is extracted, cleaved into fragments, separated by gel electrophoresis, denatured, and blotted onto a nitrocellulose filter. There it is incubated under annealing (pairing) conditions with a radio-labeled probe for the sequence in question, and hetero-duplexes of the probe with genomic DNA are detected by radioautography.
[5] A technique similar to Southern Blotting for detecting the presence and the size of specific RNA sequences in a cell.

nucleic acids (A-T, G-C). Arrays of thousands of discrete DNA fragments which are complementary of RNA of interest are printed onto a surface, for example, glass microscope slides. The sample spot sizes in Microarray are typically less than 200 microns in diameter and these arrays usually contain thousands of spots. RNA collected from cells that has undergone different experimental conditions are collected and incubated with DNA fragments on the glass slides to allow complementary pairs to bind. The incoming RNA from the experiments is usually labeled with easily detectable substance, such as radioactive material or fluorescent dyes. The quantity of RNA corresponds to the quantity of the labels that are measured at each spot where the complementary DNA sequence is printed.

There are two variants of DNA Microarray technology. With the first method, often referred to as the "traditional" DNA Microarray developed at Stanford University, probe cDNA (500~5,000 bases long) is immobilized to a solid surface such as glass using robot spotting and exposed to a set of targets either separately or in a mixture. In an experiment, a sample is taken from a normal condition to be used as control and the experimental result is usually in the form of relative abundance (in the form of a ratio) of each of these gene sequences in the two RNA samples (experimental vs. control). The experimental sample and the control samples are first labeled using different fluorescent dyes, usually a red dye and a green dye. They are then mixed and hybridized with the arrayed DNA spots. After hybridization, fluorescence measurements of each color are made. The measured intensities are used to calculate a ratio, representing the relative abundance of sequence of each specific gene in the two mRNA sample, with respect to

the “normal” expression quantity. This is also often called the two color array system. A second method, “historically” called DNA Chips, is developed by Affymetrix, Inc. With this method, an array of oligonucleotide (20~80-mer oligos) or peptide nucleic acid (PNA) probes is synthesized either *in situ* (on-chip) or by conventional synthesis followed by on-chip immobilization. The array is exposed to labeled sample DNA, hybridized, and the identity and the abundance of complementary sequences are determined. The measurement and calculation of abundance of each DNA is usually proprietary.

New and better techniques of DNA Microarray are allowing simultaneous measurements of more RNA by allowing more DNA sequences to be printed onto the same surface. The completion of the genome sequence of model organisms, such as Saccharomyces cerevisiae and Caenorhabditis elegans, and dozens of bacterial species provides us with more complete genomes [16] to conduct Microarray experiments with. Currently we are able to monitor gene expression profiles of organisms such as S. *cerevisiae*, with approximately 6300 genes. Soon we will be able to monitor the approximately 30,000 genes in the human genome all at the same time.

While the technology is promising, we are now faced with a new challenge of extracting relevant information from the data DNA Microarray technology generates. With tens of thousands of genes monitored simultaneously and hundreds of experimental conditions and measurements, a single experiment can yield tens of millions of data points. As a result of this large amount of data, data mining and data analysis have received great interest in recent years in the area of gene expression. DNA Microarray

experiments are exploratory in nature. The technology is used for observing, describing and mapping undiscovered territory, rather than testing theories or models [15]. Therefore, the data analysis tools are developed and used for the purpose of exploration. The two popular types of data mining techniques in the area of gene expression are clustering and classification. In the next section, these techniques will be described in more detail.

1.2 Data Mining

Data mining has been an area of research in Computer Science for decades now. The emergence of advanced technology and data gathering devices such as point-of-sale or remote sensing devices has lead to an explosion of data stored in electronic format. This accumulation of data occurs in every area from marketing to molecular biology at an exponential rate. With computing power and storage media becoming available at low cost, we can accommodate the storage of such large amount of data. The challenge we are facing now is to be able to derive meaning from the data we now have. As a result, data mining has attracted increasing amount of attention in the past years. In this section, I will give an overview of general concepts and steps of data mining, discuss several data mining techniques, and eventually focus on association rule mining as a means of extracting information.

1.2.1 Data Mining Overview

Data mining, also known as Knowledge Discovery in Databases – KDD, has been defined as "the nontrivial extraction of implicit, previously unknown, and potentially useful information from data. This encompasses a number of different technical approaches, such as clustering, data summarization, learning classification rules, finding dependency networks, analyzing changes, and detecting anomalies"[6] It uses machine learning, statistical and visualization techniques to discover and present knowledge in a form which is easily comprehensible to humans. For example, data from business transactions gathered from point-of-sales systems can be used to extract the spending patterns of customers in a certain region. This data can be used to determine what goods should be promoted to these customers to benefit businesses. Data mining has application in areas from business to medical diagnostics. Several techniques have been developed for data mining.

1.2.2 Data Mining Techniques

1.2.2.1 Steps Involved

The general strategy for data mining involves several steps: data selection, data preprocessing, data transformation, data mining and interpretation and evaluation [30].

1. Data selection is a process for selecting the data according to some criteria. The criteria are usually determined based on the specific domain of application. In addition, the data should be relevant to the question that is asked. This step is usually carried out under the guidance and the knowledge of a domain expert. For example, data collected from point-of-sale systems may contain different information. Information such as pricing is not relevant to identifying the associations between sales of individual merchandise. Such information must be first removed.

2. Data preprocessing is the data cleansing step in which unnecessary information is removed from the selected data, often referred to as "scrubbing the data." This step is also needed when data from different sources are integrated to ensure that all data will have the same format. One example would be normalizing data so that data comparison is more relevant.

3. Data transformation is the preparation step to make data of one representation into another optimal representation or structure usable by the target technique and algorithm. This step is needed because data collected often is not usable in its original format. For example, in market basket data analysis, the data collected from point-of-sales system may contain quantities of items for each sale. A particular analysis technique

may only be concerned with whether an item is present in the transaction. So the data represented by quantity may need to be transformed into a Boolean representation of the presence or absence of each item in the transaction so that the data mining technique can be applied.

4. Data mining is the actual analysis step using specific techniques and algorithms. In this step, prepared data is further compressed or transformed so that one can easily identify any latent valuable nuggets of information.

5. Data interpretation and evaluation is the final step. After analysis results are generated by the data mining algorithms, they are used to see if additional domain knowledge is discovered and to determine the relative importance of the facts generated. Sometimes, they are also interpreted as knowledge which can then be used to support human decision-making. The results from this step often provide feedback for additional iterations of the data mining process.

1.2.2.2 Data Mining Techniques

There are several widely used data mining techniques. Traditionally, these techniques are carried out independently. These techniques include: *classification*, *clustering*, and *association rule mining*.

Classification is a process in which common characteristics among objects are identified. Objects in turn are classified into different classes. Classification usually requires a training set with labels. The training set is used to develop an accurate model to characterize each class. Once the model has been developed, incoming data is fitted into this model and classified with other objects sharing similar characteristics. The classification technique is a supervised learning technique. One example of classification would be Support Vector Machines. Support Vector Machines are a method for creating functions from a set of labeled training data. The function can be a classification function (the output is binary: is the input in a category) or the function can be a general regression function. For classification, SVMs operate by finding a hyper-surface in the space of possible inputs. This hyper-surface will attempt to split the positive examples from the negative examples. The split will be chosen to have the largest distance from the hyper-surface to the nearest of the positive and negative examples, such that the margins are maximized. Intuitively, this makes the classification correct for testing data that is near, but not identical to the training data. In order to successfully construct a classification model, there must exist a set of class identities or labels. This method of

analysis is useful in applications such as customer profiling, credit rating, medical diagnostics, and chemical classification. This method is not particularly useful when no pre-existing identity or labels are available.

Clustering is a process of partitioning a collection of objects into groups or clusters whose members share a common set of characteristics. It is similar to classification except that no labels are used, and it is an unsupervised method. Clustering is typically used to explain the characteristics of a data distribution. One example of clustering is: with a list of customers and various characteristics regarding these customers, a company can identify several market segments (types of customers the company has) and descriptions of these segments, including similarities among customers within a segment as well as distinguishing characteristics between segments. There are three common approaches to clustering: k-means (or partitioning) clustering, self-organizing maps, and hierarchical clustering [3, 4, 5]. **K-means** clustering is a very simple form of analysis that assigns each object from a collection to exactly one group. The technique begins with k cluster centers, where k is specified when the technique is applied. For each object, the cluster center that's nearest to it is found. The object then is placed in the cluster represented by this cluster center. As objects are added to and removed from each cluster, the cluster centers are recomputed. The second step is repeated to reassign objects to its nearest cluster center. The cluster centers are recomputed. This iteration is repeated. The algorithm terminates when no more clusters are altered. **Hierarchical** clustering also generates groups of objects. But each resulting group of size greater than one is in turn composed of smaller groups. There are two

approaches: bottom-up and top-down. The bottom-up approach involves several steps. We first start with n clusters given n objects, each containing one object. Then the two most similar clusters are merged into one cluster and the previous step is repeated until there is only one cluster. The top-down approach is the reverse of the bottom-up approach. The technique of **Self-Organizing Maps** is somewhat related to k-means clustering. It is based on neural networks. This method produces ordered low-dimensional representations of an input data space. Typically, such input data is complex and high-dimensional with data elements being related to each other in a nonlinear fashion. The basis of the method derives from biophysical studies of the brain and brain maps which form internal representations of external stimuli in a spatial manner. The mechanism of the technique is somewhat complex and will not be discussed here. Clustering is useful as a stand-alone tool for data analysis. It can provide useful insight in the data exploratory process. Clustering can also be used as a preprocessing step for data mining as seen in our study.

Association rule mining from large datasets has received much attention in recent data mining research. Association rule mining is a process that identifies links between sets of correlated objects in transactional databases where each transaction contains a list of items. If we are given a set of items $I = \{i_1, i_2, \ldots, i_n\}$, a transaction T where T is a subset of I, $T \subseteq I$ (T needs not to be in the same sequence as I), an association rule is of the form X=>Y, where $X \subset I$, $Y \subset I$, and $X \cap Y = \emptyset$. A typical application of association rules, identified by IBM, is Market Basket Analysis. For example, a grocery retailer runs an association operator over the point of sales transaction

log, which contains transaction identifiers, which in turn contain product identifiers. The output of the analysis is a list of product affinities such as "20% of the times when customers buy Wonder wheat bread, they also buy 2% milk." This produces information at a level higher than simply assigning objects into groups. The association relationship or implication among objects can be explored. The objective for association rule mining is to find rules with high *support* and *confidence*, which are measures of the likeliness of the rules. The details of association rule mining mechanisms will be discussed in the next section.

1.2.3 Mining Association Rules

1.2.3.1 Definition

Association rule mining was first introduced in 1993 [7]. The formal statement of the problem is as follows. Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of literals, called items. Let D be a set of transactions, where each transaction T is a set of items such that a transaction $T \subseteq I$ and $T \in D$. We say that a transaction T contains X, a set of items in I, if $X \subseteq T$. An *association rule*, as mentioned previously, is an implication of the form X=>Y, where $X \subset I$, $Y \subset I$, and $X \cap Y = \emptyset$. The rule X=> Y holds in the transaction set D with *confidence* c if c% of the transactions in D that contain X also contain Y. The rule X=>Y has *support* s in the transaction set D if s% of transactions in D contain $X \cup Y$.

The *support* of X is the fraction of transactions T supporting the itemset X with respect to database D, $supp(X) = |\{T \in D | X \subseteq T\}| / |D|$. We often see in literatures that the count of an item in a given database instead of the fraction is used as the support. Here in this thesis, we also use count and support interchangeably. The support of a rule X=>Y is defined as $supp(X=>Y) = supp(X \cup Y)$. An itemset X is called *large* or *frequent* if its support exceeds a given threshold, ξ. All others are called *small* or *infrequent* itemsets. The *confidence* of the rule X=>Y is defined as the percentage of transactions containing Y in addition to X with regard to the overall number of transactions containing X, $conf(X=>Y) = supp(X \cup Y)/supp(X)$. For the association rule X=>Y to be significant, $X \cup Y$ must be large and the confidence of the rule must exceed a given confidence threshold, γ. This can be thought of in probability terms as $P_{(\{X \cup Y\})} > \sigma$ and $P_{(Y|X)} > \gamma$, where $P_{(Y|X)} = P_{(XY)} / P_{(X)}$ [8, 9, 10].

1.2.3.2 Search Space Traversal

As described in the definition section, in order to generate association rules, we must find all itemsets that satisfy the minimum support. Examining all subsets of I is impractical due to the large search space. A linearly growing number of items imply an exponential growth of itemsets to be considered. Fortunately there is a general strategy we can employ while considering itemsets based on the downward closure property of itemsets. For example, we consider the case I = {A, B, C, D}. We can represent the search space or all possible itemsets with the following conceptual structure.

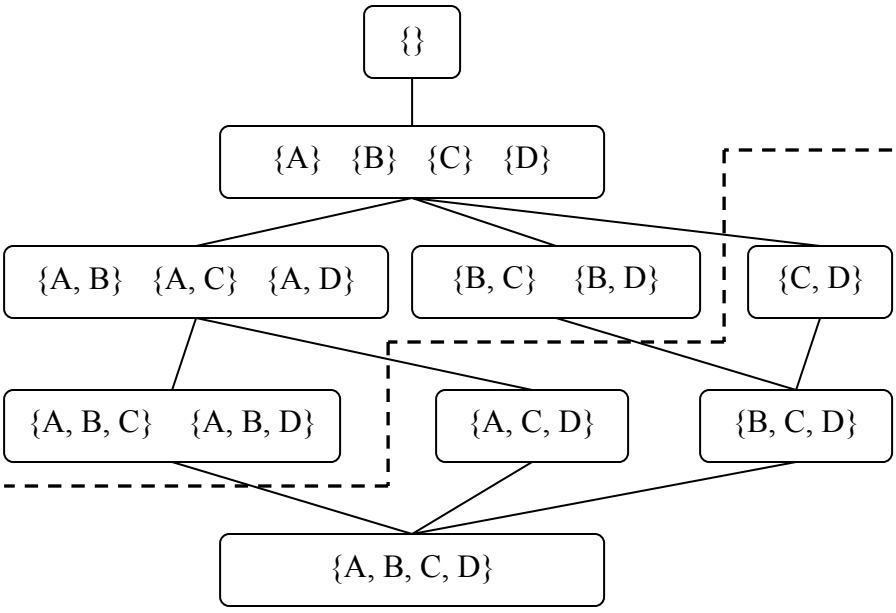

Figure 1.2 The construction of all itemsets for I = {A, B, C, D}. The rectangle grouping indicates the itemsets in each class (level) that share the same prefix.

If the support of itemset {C, D} does not satisfy the threshold (infrequent itemset), then we have a dashed line that separates the frequent itemsets (upper) and the infrequent itemsets (lower). The existence of such a border is guaranteed by the downward closure property of itemset support, that if a parent itemset does not satisfy the minimum support, neither will its child itemset. In this case, if {C, D} does not satisfy the minimum support threshold, we do not need to consider itemsets that contain {B, C, D}. The basic principle of the common algorithm is to employ this border to efficiently prune the search space. When we encounter an itemset that is not frequent, then the subsequent itemsets need not be considered. It is also implied that if the parent class E' of E does not contain at least two frequent itemsets, then, E must not contain any frequent itemset either. Once we

encounter such a class E' on our way down the tree, we have reached the border separating the infrequent and the frequent itemsets, and we do not need to go beyond this border. An itemset that is potentially frequent and for which we decide to determine its support during lattice traversal is called a candidate.

1.2.3.3 Existing Algorithms

One of the problems with mining association rules is the algorithm complexity. The number of rules generally grows exponentially with the number of items. Naturally, optimization has focused on efficiently pruning the search space, and a number of algorithms have been developed. An algorithm is characterized by a) its search space traversal strategy (BFS or DFS), and b) by its strategy to determine the support values of the itemsets (counting and intersecting). I will discuss several popular algorithms here.

Before discussing the algorithms, we need to first discuss the strategies to determine the support value of an itemset. There are two common approaches: *counting occurrences* and *TID-list intersections* [8, 10]. When determine the support values by *counting occurrences*, the counter for each itemset investigated is initialized and set to 0. All transactions are scanned and whenever a candidate is recognized as a subset of a transaction, its counter is incremented. Typically subset generation and candidate lookup are integrated and implemented with a hashtree or a similar data structure. The second common approach to determine the support values is by *TID-list intersections*. A TID is a unique transaction identifier. For a single item the TID-list is the set of identifiers that

correspond to the transactions containing the item. For every itemset X there is also a TID-list, denoted by X.TID-list. The TID-list of $X = A \cup B$ is obtained by X.TID-list = A.TID-list $\cap$ B.TID-list. TID-lists are usually sorted in an ascending order to allow efficient intersections. The actual support value of a candidate X is |X.TID-list|.

Apriori [8] is a popular breadth first search and occurrence counting algorithm. The heart of this algorithm is making use of the downward closure property of itemsets by pruning the candidates that have infrequent subsets before counting their supports. This optimization becomes possible because BFS ensures that the support values of all subsets of a candidate are known in advance. For example, if supp({A,B}) < min. support, {A,B,C,D} will not be considered. Apriori counts all candidates of a cardinality k together in one scan over the database. A hashtree structure is used for the purpose of looking up the candidates in each of the transaction. The items in each transaction are used to descend in the hashtree. Whenever we reach one of its leaves, we find a set of candidates having a common prefix that is contained in the transaction.

Partition [11] is an Apriori-like algorithm that uses BFS and TID-list intersection. As described above, Apriori determines the support values of all candidates of cardinality k-1 before counting the candidates of cardinality k. Instead of counting, Partition uses the TID-list of the frequent (k-1)-itemsets to generate the TID-lists of the k-candidates by appending single additional item to the frequent (k-1)-itemsets. One of the problems with Partition is that when generating TID-lists of k-candidates, the size of intermediate results easily grows beyond the physical memory limitations of common machines. Partition

overcomes this by splitting the database into several chunks that are treated independently. In the end, an extra scan is required to ensure that locally frequent itemsets are also globally frequent.

Eclat [11] is introduced that combines depth first search with TID-list intersection to address the memory problem. When using DFS, it suffices to keep the TID-list on the path from the root down to the node itemset currently investigates (see the previous subsection). Splitting the database done by Partition is no longer needed. Eclat employs an optimization called "fast intersection," in that whenever two TID-lists are intersected, we only consider the resulting TID-list if its cardinality reaches min support. In other words, each intersection is eliminated as soon as it does not meet the minimum support.

FP-growth [12] is a recently introduced approach using DFS and counting occurrences. Generally, using a DFS approach to scan every node in the search space results in tremendous overhead. The simple combination of DFS with counting occurrences is of no practical relevance [13]. But FP-Growth is considered a fundamentally new approach. The heart of this algorithm is a pre-processing step in which FP-growth derives a highly condensed representation of the transaction data, a FP-tree. In a second step, FP-growth uses the FP-tree to derive the support values of all frequent itemsets. This algorithm will be discussed in detail in the next chapter.

1.3 Motivation and Research Goal

With the promise of DNA Microarray technology also comes the challenge of disseminating the vast amount of data we have collected. Numerous data mining tools have been developed and adapted for gene expression data analysis. Most of the methods used are based on the classification or the clustering techniques of data mining [21, 31, 32, 33]. Little work has been done using association rule mining. These methods (supervised and unsupervised) are useful for grouping genes, and identify genes of unknown functions by their resembling characteristics to other genes of known functions. While the existing tools are useful for determining membership of genes by homology, they do not identify the regulatory relationships among genes that are found in the same class of molecular actions. For example, using clustering tools, we can say that gene A, B, and C are closely related in their expression pattern. But we cannot say anything about the relationship among A, B, and C. With association rule mining, we can take a step further and may be able to discover relationships such as A=>{B, C}, that when gene A is expressed a certain way, B and C are also expressed a certain way with a confidence c. The reverse isn't necessarily true. Intuitively, this is very similar to what we consider as regulatory network for gene expression. By discovering relationships such as A-up=> {B-up, C-up}, we might be able to also construct a regulatory network by combining the association rules to describe a regulatory process. Because of the exploratory nature of DNA Microarray technology, we believe that association rule mining may be a useful tool for discovery regulatory relationships among genes from gene expression data.

Chapter 2: Association Rule Mining Using FP-growth

2.1 Introduction

Association rule mining is a two step process. The first step is to find frequent patterns to serve as candidates for association rules. This step requires finding the support for itemsets that satisfy a specified minimum value. The second step is to generate association rules using the itemsets found during the first step. This step involves calculating the confidence of potential rules and selecting those with confidence that, again, meet a minimum threshold. As we have discussed in the previous chapter, the first step is computationally intensive. The number of itemsets to be considered grows exponentially with linear growth of the number of items. Therefore, most of the research and algorithms developed have been addressing the first step of the process. In 2000, Han et al. [12] introduced the FP-Growth algorithm for discovering frequent patterns. We will discuss this algorithm in detail in this chapter.

2.2 FP-Tree for Pattern Generation

Conceptually, the FP-Growth algorithm uses a Depth First Search, a recursive process to "grow" frequent patterns from pattern fragments. The algorithm achieves efficient data mining with three techniques: 1. the large database is compressed into a highly condensed data structure which avoids costly, repeated database scans, 2. a pattern fragment growth method is used to avoid the costly generation of a large number of candidate sets, and 3. partitioning based on a divide-and-conquer method is used to decompose the problem into smaller tasks, which dramatically reduces the search space.

The data structure used by this algorithm is a Frequent Pattern tree or FP-Tree. The algorithm generates the FP-Tree from the transaction database based on the support of items in the transactions in the database. Using each item as the *suffix* pattern (of length 1), a *conditional* database is generated (a sub-database that consists of all transactions containing the *suffix* pattern). From the *conditional* database, a *conditional* FP-Tree is generated. The two steps are repeated recursively. The *conditional* databases and the *conditional* FP-trees from the recursive process will decrease in size with each iteration. Pattern growth is achieved by concatenating the suffix patterns resulting from each recursive step.

2.2.1 Frequent Pattern Tree

Frequent Pattern Tree, or FP-Tree, is an extended prefix-tree structure for storing quantitative information about frequent patterns. The use of the prefix-tree is to compact the representation and storage of most frequent, or commonly shared patterns.

Consider the following example transaction database represented by the following table. We can make the following observations:

Transaction	Items
1	f, a, c, d, g, i, m, p
2	a, b, c, f, l, m, o
3	b, f, h, j, o
4	b, c, k, s, p
5	a, f, c, e, l, p, m, n

Table 2.1 An example transaction database

1. Since we need to consider only the frequent items, it is sufficient to scan the database to identify the set of frequent items by counting frequency.

2. If the set of frequent items of each transaction is stored in some compact structure, we can avoid repeated scans of the database.

3. If multiple transactions share identical frequent items, they can be merged into one with the number of occurrences registered as a count. This check can be done easily if the items in all of the transactions are sorted according to a fixed order.

4. If two transactions share a common prefix according the sorted order of items, the shared parts can be merged using one prefix structure as long as the count is registered properly. If a descending order is used to sort the frequency of items, there is a better chance that more prefixes can be shared.

With these observations in mind, we can define our compact data structure, FP-Tree as follows:

1. FP-Tree consists of one root labeled as *null*, a set of item prefix sub-trees, and a sorted frequent item header table.

2. Each node in the item prefix sub-tree consists of three fields: *item-name*, *count*, *node-link,* and *parent*. *Item-name* registers the item this node represents. *Count* registers the number of transactions represented by the portion of the path in the prefix tree from the root reaching this node. *Node-link* links to the next node in the FP-Tree carrying the same *item-name*. When there is none, *node-link* links to null. This results in a linked-list structure for nodes with the same *item-name*. *Parent* points to the parent node in the prefix tree.

3. Each entry in the frequent item header table consists of two fields, *item-name,* which registers the item represented, and *head of node-link* which points to the first node in the FP-Tree carrying the same *item-name*.

Based on the definition, the **FP-Tree Construction Algorithm** is:

Input: A transaction database *D* and a minimum support threshold ξ.

Output: Its frequent pattern tree, FP-Tree.

Method: The FP-Tree is constructed in the following steps.

1. Scan the transaction database *D* once. Collect the set of frequent items *F* and their respective support values. Sort *F* in descending order as *L*, the list of ordered frequent items that satisfy the minimum support threshold ξ.
2. Create the root of an FP-Tree, *T*, and label it as null. For each transaction, *Trans*, in *D* do the following.
    - Select and sort the frequent items in Trans according to the order of *L*. Let the sorted frequent item list in Trans be [*p*|P], where *p* is the first element and *P* is the remaining list. Call *insert_tree*([*p*|P], *T*).
    - The function *insert_tree*([*p*|P], *T*) is performed as follows. If *T* has a child *N* such that *N.item-name* = *p.item-name*, then increment *N.count*; else create a new node *N* and set *N.item-name* = *p.item-name*, and set *N.count* = 1, its parent link be linked to *T* and its *node-link* be linked to the nodes with the same item-name via the *node-link* structure. If *P* is nonempty, call insert_tree(*P*, *N*) recursively.

Given the definition and the construction algorithm, we construct a FP-Tree in the following example. Given a minimum support $\xi = 3$ and a transaction database shown in Table 2.2, we have a list of frequent items from a database scan {(f:4), (c:4), (a:3), (b:3),

Transaction	Ordered Frequent Items
1	f, c, a, m, p
2	f, c, a, b, m
3	f, b
4	c, b, p
5	f, c, a, m, p

Table 2.2 Frequent items in each transaction sorted in descending order of frequency

(m:3), (p:3)} with descending order. We sort the frequent items in each transaction as shown in Table 2.2. From the frequent items, we now construct the FP-Tree. We create the root of the tree, labeled with *null*. Transaction 1 (f, c, a, m, p) leads to the construction of the first branch, with *count* of each node equals to 1. Transaction 2 (f, c, a, b, m) shares a common prefix (f, c, a) with the existing path (f, c, a, m, p), so the count of each node along the prefix is incremented by 1. A new node (b:1) is created and its *parent* points to (a:2) and another new node (m:1) is created and is linked as the child of (b:1). Transaction 3 (f, b) only shares prefix (f) with the existing paths. Node f's count is incremented to 3 and a new node (b:1) is created and linked as a child of (f:3). Since we already have a node (b:1), the node link of this node will point to the newly created (b:1) to create a linked list. Transaction 4 (c, b, p) does not share a prefix with the existing paths in the tree, therefore leading to the creation of the second branch of the tree, <(c:1),

(b:1), (p:1)>. Node (c:2) will have its node link point to the newly created node (c:1). The last node, (b:1), in the linked list for item b will have its node link point to the newly created (b:1). Transaction 5 (f, c, a, m, p) is identical to transaction 1, the path is shared and the count of each node along the path is incremented by 1. We then have the following FP-Tree:

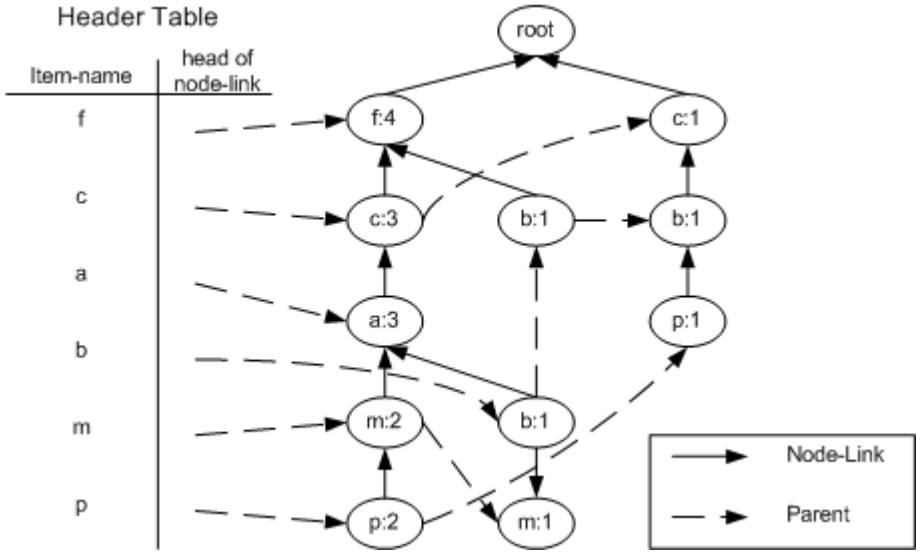

Figure 2.1 The FP-Tree constructed from the example transaction database

Based on the construction algorithm, we see that two scans of the transaction database are required: first to collect the set of frequent items, and the second to construct the FP-Tree. Inserting a transaction into the FP-Tree is $\Theta(|T|)$, where $|T|$ is the number of frequent items in a transaction.

2.2.3 Pattern Generation Using FP-tree.

Mining with a compact data structure such as FP-Tree does not automatically guarantee that it will be highly efficient. One may still encounter the combinatorial problem of candidate generation if we simply use this FP-tree to generate and check all the candidate patterns. [12] In this section, we will discuss in detail the algorithm used to mine frequent patterns using the FP-Tree, described by Han et al.

In order to facilitate the frequent pattern mining process, there are several properties of the FP-Tree structure we should consider.

Property 2.2.3-1 (Node-link property) *For any frequent item a_i, all the possible frequent patterns that contain a_i can be obtained by following a_i's node-links, starting from a_i's head in the FP-Tree header.*

Property 2.2.3-2 (Prefix path property) *To calculate the frequent patterns for a node a_i on a path P, only the prefix sub path of node a_i in P needs to be accumulated, and the frequency count of every node in the prefix path should carry the same count as node a_i.*

Rationale The count of nodes in this prefix path can be greater than the count of a_i. But the instances of these nodes in the frequent patterns containing a_i will have the count set to the count of a_i.

Lemma 2.2.3-1 (Fragment growth) *Let α be an itemset in D, B be α's conditional pattern base (a sub-database that consists of all transactions containing the suffix pattern α), and β be an itemset in B. Then the support of α∪β in D is equivalent to the support of β in B if support is the count of the item.*

Rationale According to the definition of conditional pattern base, each (sub) transaction in B occurs under the condition of the occurrence of α in the original transaction database D. If an itemset β appears in B ψ times, it appears with α in D ψ times as well. Since all such items are collected in the conditional pattern base of α, α∪β occurs exactly ψ times in D as well.

Corollary 2.2.3-1 (Pattern growth) *Let α be a frequent itemset in D, B be α's conditional pattern base, and β be an itemset in B. Then α∪β is frequent in D if and only if β is frequent in B.*

This corollary is the case when α is a frequent itemset in D, and when the support of β in α's conditional pattern base B is no less than ξ, the minimum support threshold.

Lemma 2.2.3-2 (Single FP-Tree path pattern generation) *Suppose an FP-Tree T has a single path P. The complete set of the frequent patterns of T can be generated by the enumeration of all the combinations of the subpaths of P, with the support being the minimum support of the items contained in the subpath.*

Based on the above properties and lemmas, we have the following algorithm for mining frequent patterns using FP-Tree.

Input: FP-Tree, minimum support threshold ξ

Output: The complete set of frequent patterns.

Method: Call **FP-Growth**(FP-Tree, null)

```
Procedure FP-Growth(Tree, α) //α is the suffix of the current FP-Tree
{
   if Tree contains a single path P
   then for each combination (denoted as β) of the nodes in the path P do
      generate pattern β ∪ α with support = minimum support of nodes in β
   else for each a_i in the header of Tree do {
      generate pattern β = a_i ∪ α with support = a_i.support;
      construct β's conditional pattern base and then β's conditional FP-Tree Treeβ;
      if Treeβ ≠ Ø
      then call FP-Growth(Treeβ, β)
   }
}
```

Let's consider the example from the previous section and the FP-Tree in Figure 2.2.1-1, where the minimum support threshold ξ=3. For node p, we have a frequent pattern (p:3) and two paths in the FP-Tree: <(f:4), (c:3), (a:3), (m:2), (p:2)> and <(c:1), (b:1), (p:1)>. Although item (f) appears 4 times, and (c, a) 3 times, they only appear twice together with p. We only consider p's prefix path <(f:2), (c:2), (a:2), (m:2)>. Similarly, the second prefix path for p is <(c:1), (b:1)>. These two prefix paths form p's conditional pattern base. Construction of an FP-Tree on this conditional pattern base, p's conditional FP-Tree, leads to only one branch (c:3). Hence only one frequent pattern (cp:3) is derived.

Item	Conditional pattern base	Conditional FP-Tree
p	{ (f:2, c:2, a:2, m:2), (c:1, b:1)}	{(c:3)}|p
m	{(f:4, c:3, a:3, m:2), (f:4, c:3, a:3, b:1, m:1)}	{(f:3, c:3, a:3)} | m
b	{(f:4, c:3, a:3, b:1), (f:4, b:1), (c:1, b:1)}	Ø
a	{(f:3, c:3)}	{(f:3, c:3)}|a
c	{(f:3)}	{(f:3)}|c
f	Ø	Ø

Table 2.3 Conditional pattern bases and FP-Trees

The search for frequent patterns associated with p terminates. Similarly, conditional pattern bases and the conditional FP-Trees can be generated as shown in Table 2.3.

2.3 Discovering Rules

Association rule generation from frequent patterns is the second step of the mining process. The association rules that we consider are more general than described by Agrawal et al. [7] in that we allow a consequent to have more than one item. Because gene regulation is not necessarily a one-to-one relationship, we need to allow the ability to generate regulations that involve multiple genes that appear on either side of the rules.

A straightforward process for generating rules is as follows. Given a minimum confidence *c*, for every large itemset L, of size greater than 1, we find all non-empty subsets of L. For every such subset a, we output a rule of the form a=>(L-a) if the

confidence of this rule, that is, the ratio of supp(L) to supp(a), satisfies *c*. We consider all subsets of L to generate rules with multiple consequents. Large itemsets are stored in a hash table, the support counts for the subset itemsets can be found efficiently. This procedure can be improved by generating the subsets of a large itemset in a recursive depth first fashion. So if a subset S of a large itemset L does not generate a rule, the subsets of S do not need to be considered for generating rules using L. For example, given an itemset ABCD, we first consider the subset ABC, then AB, etc. If ABC=>D does not have enough confidence, we do not need to check if AB=>CD holds. This observation can be generalized as if a=>(L-a) does not hold, neither does c=>(L-c) for any $c \subset a$.

Chapter 3: Methods

3.1 Implementation of FP-Growth Algorithm

The FP-Growth algorithm is implemented in Java for our study. The following UML diagram shows the design of the program in the form of a class interaction diagram. Implementation of both FP-Tree and the association rule mining algorithm follows the details discusses in chapter 2.

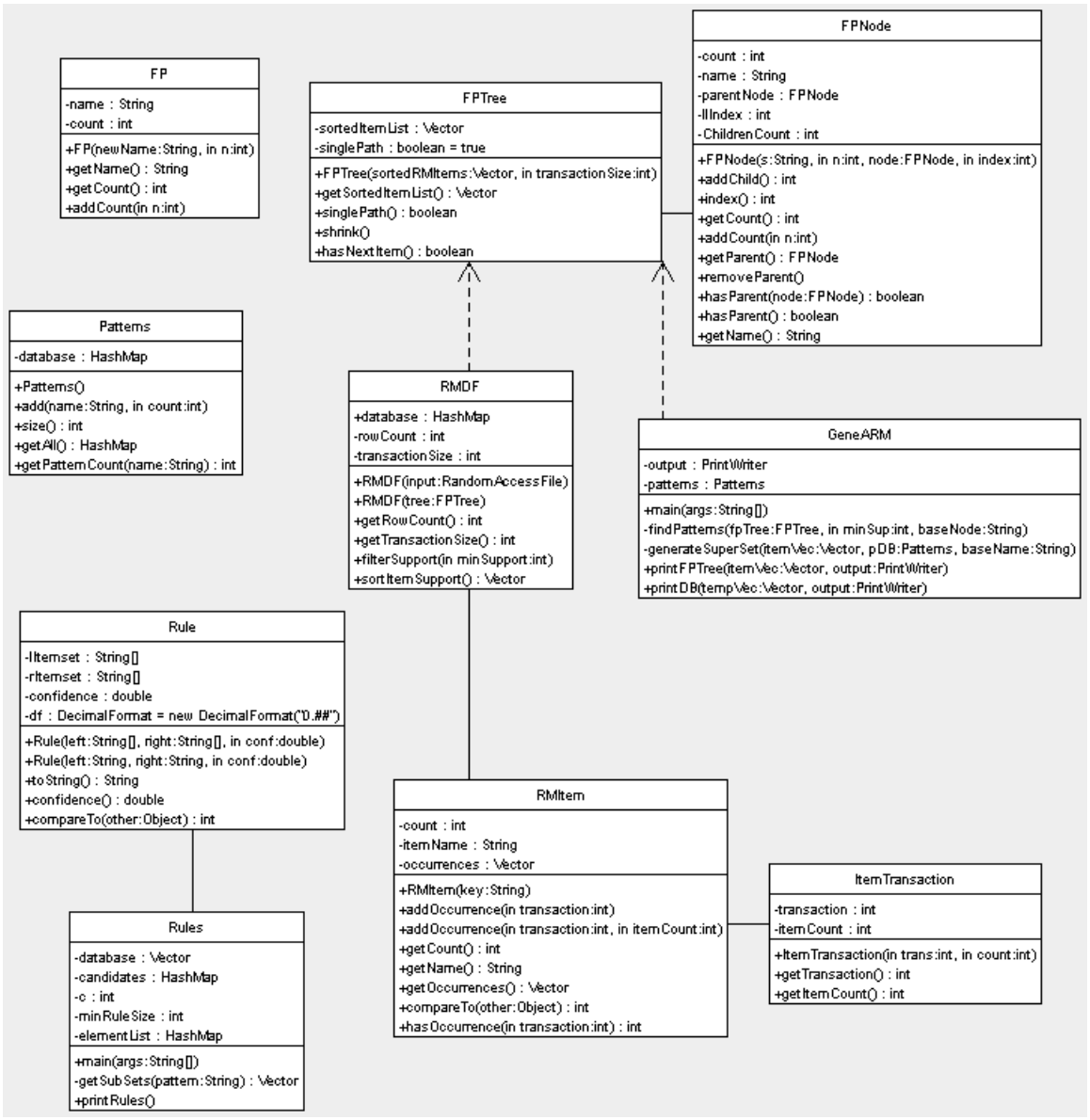

Figure 3.1 Class interaction diagram of the program that implements FP-Growth algorithm

3.2 Data Selection

The objective of this research is to use an association rule mining method to discover regulatory relationships among genes. The rules generated can further be combined to generate regulatory networks. Due to the fact that association rule mining is based on a probabilistic model, there are several features in the data we are looking for: large sample size, correlation in experimental design and annotation that allows us to go back and verify the results.

The target species for this study is *Saccharomyces cerevisiae*, or budding yeast. Yeast is considered the model organism for genetic research of eukaryotes. It shares great similar molecular functions with other higher eukaryotic organisms. At the same time, being a microorganism, yeast shares many of the technical advantages that permitted rapid progress in the molecular genetics of prokaryotes and their viruses. Some of the properties that make yeast particularly suitable for biological studies include rapid growth, dispersed cells, the ease of replica plating and mutant isolation, a well-defined genetic system, and most important, a highly versatile DNA transformation system. Being nonpathogenic, yeast can be handled with little precautions. Large quantities of normal bakers' yeast are commercially available and can provide a cheap source for biochemical studies. These properties of yeast allowed it to be used as the primary research system for many molecular biologists in the last several decades. A large percentage of the molecular functions in yeast have been well characterized. More significantly, the complete yeast genome has been sequenced since 1996 [18, 19] and many genes have

been identified. This makes it relatively easy to validate results from our association rule mining method using known biological functions of genes studied. With the availability of Microarray technology, there also have been a great number of systematic investigations of all genes in the yeast genome. There is an abundant source of yeast gene expression data available to us. Of the publicly available experimental data, we selected two relatively large datasets that were published in recent studies to be used in our research.

The first dataset was published by T.R. Hughes, et al. [20]. This study constructed a reference database or "compendium" of full genome expression profiles of yeast corresponding to 300 diverse mutations and chemical treatments in yeast. DNA mutations and chemical treatments were introduced to trigger DNA damage responses in yeast cells. The original experiments were designed to capture the gene expression profile as a response to these damages and to identify genes that play a role in DNA damage repair. This experiment was particularly good for our research, because DNA damage repair is a highly regulated process that involves several identified key players. The ability to identify genes that are involved in the regulation of the repair response can be used to validate our association rule mining method. The resulting dataset, including 300 compendium experiments and 63 control experiments, was obtained as supplemental data from the Rosetta Inpharmatics website: http://www.rii.com/publications/cell_hughes.htm.

The second dataset is a smaller dataset generated by Eisen et al. [21] from a genome wide expression study of yeast during its cell growth/reproduction cycle. This

dataset consists of combined 127 measurements. The experiments were designed to capture the yeast gene expression profiles during normal cell growth and proliferation, as well as under several artificial treatments. This is a good dataset for discovering gene regulations that occur during a cell's life cycle.

3.3 Data Preprocessing

The traditional input for association rule mining, such as market basket data, has some significant differences from gene expression data. The first difference is that market basket data contains a large number of transactions (instances of subsets of items): e.g. in the thousands, and the number of total items in the transaction database is relatively smaller: e.g. in the hundreds. An instance of an item in a specific transaction is usually represented with a Boolean value, present or not present. It is also the case that not all items are represented in all transactions. The absent items in a given transaction are ignored, therefore, the size of the resulting FP-tree is relatively smaller. On the other hand, gene expression data usually contains a large number of items (in our case, genes that represent the entire genome), in the range of thousands to tens of thousands. In the case of yeast experiments, there are a little over 6300 genes. The number of measurements, which is equivalent to transactions from market basket data, is much smaller, often in the range of hundreds per experiment (with improvements of DNA Microarray technology and decrease in cost, larger experiments with thousands of measurements are becoming common practice). Also, in most cases all genes (items) are

represented in every measurement (transaction). As stated in the following lemma, since the size of the FP-Tree is determined by the number of items in the transactions, and the ability to compact transactions into prefixes, it is expected that gene expression data will produce larger FP-Trees.

Lemma 4.1-1 *Without considering the (null) root, the size of an FP-tree is bounded by the overall occurrences of the frequent items in the database, and the height of the tree is bounded by the maximal number of frequent items in any transaction in the database.*

Rationale Based on the FP-tree construction process, for any transaction T in database D, there exists a path in the FP-tree starting from the corresponding item prefix sub-tree so that the set of nodes in the path is exactly the same set of frequent items in T. Since no frequent item in any transaction can create more than one node in the tree, the root is the only extra node not created by frequent item insertion, and each node contains one node-link and one count information, we have the bound of the size of the tree stated in the Lemma. The height of any p-prefix sub-tree is the maximum number of frequent items in any transaction with p appearing at the head of its frequent item list. Therefore, the height of the tree is bounded by the maximal number of frequent items in any transaction in the database, if we do not consider the additional level added by the root. □

Because the height of the FP-Tree determines the depth of the recursive step in the algorithm given in section 2.23, pattern generation from FP-Tree constructed from gene expression data will use more computation. In order to address this issue, we need

to find a way to reduce the number of items to be considered in our transaction database. Because we are looking for correlation between genes from gene expression data, we can select a subset of genes that seem related by preprocessing the raw data using a clustering method to organize correlated genes into clusters. Once the groups are obtained, we perform our association rule mining on genes that are in the same cluster as well as genes in several clusters. This step not only will reduce the depth of the recursion in the algorithm, but also serve as a pre-process step that help us select the genes that are related.

There are several types of clustering method as described in Chapter 1. Here we use K-means clustering methods to simply place genes into related groups. K-means clustering is a nonhierarchical method that initially takes the number of components of the population equal to the final required number of clusters. In this step the final clusters is chosen as an input for the algorithm and the initial points are chosen such that the points are mutually farthest apart. Next, each component in the dataset is examined and assigned to one of the clusters to produce a minimum distance to the cluster center. Each cluster center is recalculated every time a component is added to the cluster and this continues until all the components are grouped into the final required number of clusters. Because clustering is not our focus of study, we used Cluster software written by Eisen to perform the k-means clustering.

In this study, because we are evaluating association rule mining as a tool to analyze gene expression data by recovering known gene regulations, we selected the clusters that contained genes we are interested in.

3.4 Data Transformation

Another key difference between gene expression data and market basket data is that an instance of an item in a transaction is represented as a Boolean value, present or not present. A frequent item is an item with number of occurrences greater than the minimum support. But the instance of a gene in a measurement is represented as a real numerical value, usually the ratio of the measured value at that particular point and a reference value[6]. Since all genes are present in each measurement, there isn't a simple notion of a frequent gene. A sample of Microarray fluorescent ratio data is shown in the following table. One can consider an ORF (Open Reading Frame), representing a gene, as an item, and each time point as the transaction.

[6] In the case of data from Affymetric technology, the data is already normalized, and represented as a log ratio using undisclosed calculation.

ORF	Time 1 Ratio	Time 2 Ratio	Time 3 Ratio	Time 4 Ratio	Time5 Ratio	Time 6 Ratio	Time 7 Ratio
YHR007C	1.12	1.19	1.32	0.88	0.84	0.38	0.43
YBR218C	1.18	1.23	0.77	0.75	0.79	0.71	2.7
YAL051W	0.97	1.32	1.33	1.18	1.12	0.88	0.93
YAL053W	1.15	1.33	1.18	1	0.81	0.7	0.96
YAL054C	0.63	0.88	1	0.88	0.81	2.78	12.5
YAL055W	0.68	1	0.92	0.96	0.81	1.28	1.85
YAL056W	1.01	1.56	1.19	1	0.94	1.09	1.41

Table 3.1 A sample listing of two dye DNA Microarray data output

The question is how we can transform this kind of data into a traditional input for association rule mining. Our answer to this question is to represent each gene as three items. Because ratios already convey a relative expression level of a gene at a given point with respect to a reference point, we can generalize this ratio r into 3 discrete values representing an increase ($r > 1$), a decrease ($r<1$), and unchanged ($r=1$) expression level relative to the reference (control) expression level. Each gene g can then be represented as 3 items in the transaction database: g-up, g-down, g-unchanged. For a given gene, at a single measurement, the three new items are exclusive. Exactly one of the three items is present in each transaction. Therefore, even though we increase the total number of items by three fold, the number of items in any transaction remains unchanged. Based on

Lemma 4.1-1, the size of the FP-tree will not increase because the size of the FP-Tree depends on the number of items in a transaction.

We considered two approaches to transforming gene expression data into the above format we discussed.

The first approach was to use measurement-pair-wise comparison, which represents the transition from one experimental condition to another. We take series-based gene expression data, for each gene, compare the ratio value of two adjacent measurements. The adjacent measurements usually represent two consecutive experimental treatments such as increase in the concentration of a substrate or increase in an unit of incubation time. This method allows us to detect the change in the data during such transition, whether the expression level of a gene has increased or decreased or unchanged. After this transformation, each transaction now is not a measurement at an experimental instance, but a transition from measurement n-1 to measurement n. If we observe an increase in the expression level of g, we then represent g-up in this transaction as "present" or 1, and g-down and g-unchanged will have the value of "absent" or 0. This transformation worked well with series-based data where consecutive data measurements are related in the experimental design. For example, in time-series experiments, the transition from measurement n to measurement n+1 actually represents the increase in time from t to $t + \tau$, where τ is an increment in time. In the case of quantity-series data, the transition represents an increase in the concentration of a substrate or a chemical.

While the above method is a simple transformation, not all experiments are series-based. Adjacent measurements don't necessarily relate to each other. For example, an experiment can consist of measurements representing mutations that don't relate to one another. The Hughes, et al. data is one example. Each measurement represents distinct mutations and is not related. It is irrelevant to look at how data changes from one measurement to another.

The second approach is to transform the measurements in-place. Since at each point, the expression level is already represented as the level relative to the reference expression level. We already have a sense of change. If the ratio is greater than 1, we know that the expression level has increased. If the ratio is between 0 and 1, then we know that the expression level has decreased. A ratio of 1 means no change in expression. If a gene has a ratio greater than 1 in a transaction, we set the value of item g-up = 1 in the transaction, and set g-down and g-unchanged = 0. If a gene has a ratio less than 1, we set the value of item g-down = 1 in the transaction, and set g-up and g-unchanged = 0. If a gene has a ratio equal to 1, we set the value of item g-unchanged=1, and set g-up and g-down = 0 in the transaction.

Chapter 4: Results and Discussion

We applied our association rule mining algorithm to the DNA Microarray data we have selected, preprocessed, and transformed. The number of rules generated is still large due to the large number of genes that we are considering. In order to validate our method of analysis, we compare the association rules we have generated, as predictions, with known gene regulations that have been identified from experimental research. We used two methods to validate the results from our association rule mining analysis.

First, we summarized the association rules we generated. Genes that have high occurrence on the left-hand side of the rules are identified. Because we are relating association rules to gene regulation, an association rule A-up =>B-up predicts that when gene A's expression increases gene B's expression also increases, therefore gene A may play a role as a positive regulator of gene B. Similarly, an association rule A-up => B-down predicts that gene A is a negative regulator of gene B and it suppresses gene B's expression. Genes that appear on the left hand side of the association rules are therefore predicted as genes that may be regulators that determine the expression of genes that appear on the right hand side of the rules. The genes are regulatory components. These

genes that are predicted as regulators are compared with their respective biological function.

Second, we select genes that have been identified to have regulatory functions, and select association rules from our results from association rule mining that contain these genes. We then determine if the association rules containing these genes have corresponding gene regulations involving the same genes that have been identified by experimental research. We also extend this method to characterize more complete regulatory processes. We use our association rules generated that contain genes that have been characterized in literature, and combine the associations to form a network. This network will show a sequence of regulations among the genes we have selected. Such regulatory network can be used to describe more complex processes involving multiple genes that take place in a yeast cell. We verify that the resulting networks we predict are consistent with what have been characterized in yeast cells.

Although the yeast genome has been completely sequenced, not all genes have been identified. In fact, a large percentage of the open reading frames or ORFs, that potentially represent genes, have not yet been identified. Because the functions of these ORFs are unknown, they are not particularly useful in validating our predictions. But we can use our analysis to predict possible functions that the genes represented by the ORFs have. In order to simplify the validation process, we performed our analysis with only the genes that have been identified so that the association rules discovered contain only genes with known functions.

4.1 Identify Genes as Regulators

Because the goal for this analysis is to identify the genes that are most likely to play regulator roles, we included all the genes for our association rule mining. We predict that genes appearing on the left hand side of the rules frequently may have regulating roles. We summarized the association rules generated from our analysis and obtained a list of top ranking genes with the highest frequency of appearance on the left hand side of the rules. For validation, we identified the ones with known biological functions from the list of top ranking genes and compared them with the biological processes they are involved in, and respective molecular functions. In the following table, the top forty genes that are found most frequently on the left hand side of the rules generated from the cell cycle data are listed. Most of these genes have catalytic functions. From the list, a large number of genes are involved in protein synthesis, which is a highly regulated process. Protein synthesis is probably one of the most active processes taking place in a cell. Proteins are needed for virtually all types of cellular functions, from energy metabolism, to transport, to DNA replication. It can be expected that genes that are involved in protein biosynthesis occur frequently as regulators. Other types of genes are also found in the list, such as HPR5, a helicase. Helicases are enzymes that catalyze the unwinding of double stranded DNA to make replication and transcription possible. They regulate gene expression by making transcription physically possible. Among the high frequency genes, we have also found transcription factors (not listed). Transcription factors are the elements that regulate the initiation of the gene expression process

(transcription). Because transcription factors are found in virtually all transcription processes, we can expect to see a high frequency of them being on the left side of the rules. Among the high frequency genes, we also found genes that have regulation functions in cell wall biogenesis, and electron transport. These are also highly regulated biological processes and are essential to cell growth and proliferation. These findings are consistent with what we would expect from a cell cycle experiment.

ORF	Gene Name	Biological Function	Molecular Function
YJL001W	PRE3	ubiquitin-dependent protein degradation	multicatalytic endopeptidase
YIL123W	SIM1	cell cycle	molecular_function unknown
YLR075W	RPL10	protein biosynthesis	structural protein of ribosome
YGL031C	RPL24A	protein biosynthesis	structural protein of ribosome
YLR216C	CPR6	protein folding	peptidylprolyl isomerase
YLR325C	RPL38	protein biosynthesis	structural protein of ribosome
YGR148C	RPL24B	protein biosynthesis	structural protein of ribosome
YER102W	RPS8B	protein biosynthesis	structural protein of ribosome
YJR099W	YUH1	Deubiquitylation	ubiquitin-specific protease
YLR167W	RPS31	protein biosynthesis	structural protein of ribosome
YGR244C	LSC2	Tricarboxylic acid cycle	succinate--CoA ligase (ADP-forming)
YFR052W	RPN12	ubiquitin-dependent protein degradation	molecular_function unknown
YHL015W	RPS20	protein biosynthesis	structural protein of ribosome
YKL145W	RPT1	ubiquitin-dependent protein degradation	adenosinetriphosphatase
YOR157C	PUP1	ubiquitin-dependent protein degradation	multicatalytic endopeptidase
YCR034W	FEN1	fatty acid biosynthesis	molecular_function unknown
YBR243C	ALG7	N-linked glycosylation	UDP-N-acetylglucosamine--dolichyl-phosphate N-acetylglucosamine-1-phosphate transferase
YER074W	RPS24A	protein biosynthesis	structural protein of ribosome
YHR006W	STP2	tRNA splicing	molecular_function unknown
YKL003C	MRP17	protein biosynthesis	structural protein of ribosome
YGL103W	RPL28	protein biosynthesis	structural protein of ribosome
YGL048C	RPT6	ubiquitin-dependent protein degradation	adenosinetriphosphatase
YOL139C	CDC33	protein synthesis initiation	translation initiation factor
YML063W	RPS1B	protein biosynthesis	structural protein of ribosome
YNL315C	ATP11	protein complex assembly	chaperone
YJL092W	HPR5	DNA repair	A helicase
YIL062C	ARC15	cell growth and/or maintenance	structural protein
YKR094C	RPL40B	protein biosynthesis	structural protein of ribosome
YLL009C	COX17	Cytochrome c oxidase biogenesis	intracellular copper delivery
YGR085C	RPL11B	protein biosynthesis	structural protein of ribosome
YGL048C	RPT6	ubiquitin-dependent protein degradation	adenosinetriphosphatase
YLR084C	RAX2	maintenance of cell polarity (sensu Saccharomyces)	not yet annotated
YJL009W		biological_process unknown	molecular_function unknown
YHR010W	RPL27A	protein biosynthesis	structural protein of ribosome
YML057W	CMP2	ion homeostasis	calcium-dependent protein serine/threonine phosphatase
YLR249W	YEF3	protein synthesis elongation	translation elongation factor
YKL117W	SBA1	protein folding	not yet annotated
YOR157C	PUP1	ubiquitin-dependent protein degradation	multicatalytic endopeptidase
YLR333C	RPS25B	protein biosynthesis	structural protein of ribosome
YER103W	SSA4	stress response	chaperone

Table 4.1 Genes predicted as regulators and their known biological functions

Among the high frequency genes, we also found genes that are not yet annotated and ORFs that are not yet identified as genes. Because the genes with known functions in this list do show regulatory roles, we predict that these unidentified ORFs may have regulatory functions. These genes may be worth further investigation by experimental research. In order to predict the biological processes that these ORFs and genes are likely to participate in, clustering analysis can be performed to see which other genes these are closely associated with. It is possible that these unidentified ORFs and genes can be found in the same biological processes. Here we list the top 20 genes and ORFs with molecular function unknown.

ORF	Gene Name
YKL195W	
YDR041W	RSM10
YLR217W	
YIL131C	FKH1
YGL097W	SRM1
YLR083C	EMP70
YOR246C	
YMR215W	
YJL183W	MNN11
YGL139W	
YKL169C	
YLL032C	
YMR311C	GLC8
YPR100W	
YBR086C	IST2
YLL044W	
YKL151C	
YJL009W	
YNL087W	
YHR097C	

Table 4.2 Gene with unknown biological functions that are predicted as regulators

4.2 Identify Regulations Using Association Rules

We believe that association rules obtained from gene expression data can be used to predict gene regulations. A rule A-up=>B-up, with the absence of rule B-up=>A-up, may indicate that gene A is a positive regulator of gene B where increase in A's expression level leads to increase in B's expression level. Similarly, a rule A-up=>B-down may indicate that gene A is a repressor of B such that increase in A's expression level actually causes B's expression level to decrease. To verify that our predictions have corresponding real gene regulations, we examined the association rules containing a small set of genes. As mentioned before, genes with unknown functions cannot be used for validation purpose. We only used genes that have been characterized. Association rules from the cell cycle data are used. Because the frequent patterns only contain known genes, the rules also contain only genes with known functions. For this analysis, we decided to look at gene regulations that are found in the yeast respiration process.

We selected association rules that contain the following genes that are known to participate in the yeast respiration process: HAP1, CYB2, CYC7, and CYC1 [23, 24]. Based on the association rules we generated, we find that lowered expression level of HAP1 gene implies the increase in expression level of CYC7 and CYC1 (HAP1-down => CYC7-up, CYC1-up). Therefore, we make a prediction that HAP1 is a negative regulator of CYC7 and CYC1 genes under the experimental conditions for our dataset. When HAP1 is suppressed, CYC7 and CYC1 increase in their expression level. It is known that transcription factor HAP1 has a role in the repression of the nuclear encoding cytochrome

genes CYC7, and CYC1 under anaerobic growth [23, 24], which is found in the original experiment. The prediction by this association rule is consistent with experimental results. Another association rule (CYB2 up => CYC7 up, CYC1 up) indicates that increase in gene CYB2's expression leads to the increase in expression level of CYC7 and CYC1 genes. We predict that CYB2 gene is a positive regulator of CYC7 and CYC1 genes. CYB2, also known as L-(+)-lactate cytochrome C oxidoreductase, is a soluble protein form the inter-membrane of the mitochondrial. The protein product of this gene transfers electrons form L-(+)-lactate to cytochrome C and is upstream of cytochrome C in the electron transport chain. Experimental findings, characterized by Ouspenski et al., show that CYB2 exhibits preferential interaction with CYC7 during the electron transfer process. [25] Again, the prediction of a positive regulation is consistent with experimental results.

Because of the large number of genes, there is a large number of resulting association rules. The results are available for download upon request.

4.3 Regulatory Networks

In most cases, a single regulation only describes a step of a process. The analysis becomes more useful when we can describe a biological process as a whole. We extend the analysis performed in the previous section and combine the association rules to form a regulatory network that may be used to describe a sequence of regulations in a

biological pathway. In this analysis, we also looked at the association rules generated from the "compendium" data which deals with DNA damage response. In this experiment, DNA damage is simulated by introducing mutations in the DNA of yeast cells and chemical treatments in the cell growth medium. The experiment was intended to trigger the DNA damage repair mechanisms in yeast cells in order to identify the genes that are involved in this process. As a result, we can expect to find gene regulations that are found in the DNA damage repair pathways.

First we repeat the analysis described in the previous section. Genes that are known to participate in DNA damage repair response are identified. Here in particular, we focus on the following genes: RAD9, RAD17, RAD23, RAD53, POL2, DUN1, CRT1, TUP1, RNR2, RNR3, and RNR4. From all of the association rules using the "compendium" gene expression data, we selected the ones containing the genes listed above. Gene RAD53 is one of the key regulators that triggers the DNA repair process as described by Allen et al. [28, 29]. We use this gene as our starting point and look for association rules where this gene is found. We found the following association rules that contain RAD53 where this gene is both regulated as well as being a regulator: RAD9up=>RAD53up, RAD24up=>RAD53up, RAD17up=>RAD53up, RAD53up=>DUN1up, and RAD53up=>RNR2up. We then followed these associations and selected the rules that, in turn, contain RAD9, RAD24, RAD17, DUN1, and RNR2. By following the associations and linking the association rules, we are able to form a regulatory network involving these genes.

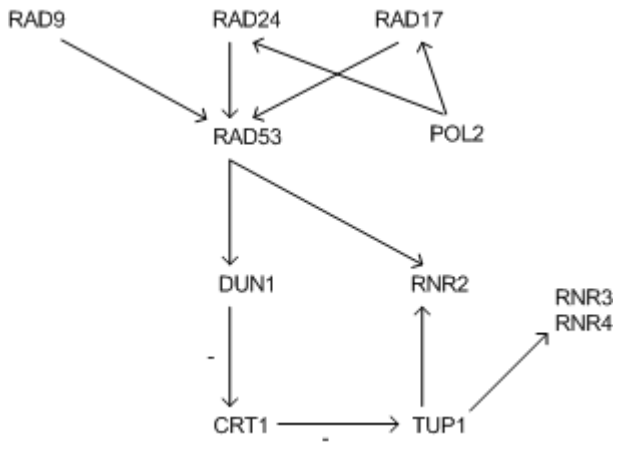

Figure 4.1 Predicted regulatory network that describes DNA damage response

The triggering of DNA repair process is highly regulated, and a result of a cascade of events. Results from laboratory experiments have shown that RNR2, RNR3, and RNR4 are genes that provide the precursor necessary for DNA synthesis and repair [25, 26, 27]. They are the trigger to initiate the DNA repair process. The expression of these genes is repressed under normal conditions. As a response to DNA damage, these genes are activated. It has also been shown that RAD53, and DUN1 have regulatory roles in inducing the expression of these three genes as a response to DNA damage [28, 29]. These two genes provide the signal for the activation of RNR2, RNR3, and RNR4. The mechanism by which this regulation is achieved is still unclear. Our prediction shown by the regulatory network indicates that RAD53 and DUN1 up-regulates the three genes by first suppressing the expression of CRT1, which, in turn, negatively regulates and causes an increase in expression of TUP1. TUP1 is a positive regulator of RNR2, RNR3, RNR4. As a result, these three genes are up-regulated and increase in expression level. The overall outcome of our prediction pathway is consistent with results observed from

experimental research: that RAD53 and DUN1 are positive regulators of the three RNR genes. In addition, our predicted network also describes the mechanism by which this regulation is achieved. How the expression of RAD53 is triggered is unclear. It has been proposed that RAD9, RAD17, RAD24, and POL2 are involved in the regulation of RAD53 based on experimental observations [29]. Our association rules also show that RAD53 is regulated by RAD9, RAD17, RAD24, and POL2 in response to DNA damage. From the regulatory network, POL2 seems to regulate RAD53 by increasing the expression level of RAD17 and RAD24, which are positive regulators of RAD53 themselves.

By combining our association rules, we are able to form a regulatory network that describes the mechanism of regulations of genes as a response to DNA damage. Not only are the predicted regulations consistent with known pathways, we are also able to predict the sequence of events by which the regulations are achieved.

Chapter 5: Summary and Conclusion

In our study, we considered the problem of discovering regulatory relationship among genes from DNA Microarray data using association rule mining. We adapted the FP-Growth algorithm for our data mining analysis. Microarray data was preprocessed and transformed so that it is suitable to be used with our algorithm.

Yeast is the organism under investigation. We chose two distinct types of experiments. The first one is a dataset from a cell cycle study done by Spellman, et al. The experiment was designed to find genes that participate in various processes during the cell cycle. The second dataset is obtained from an experiment done by Hughes, et al to discover genes that are involved in DNA damage response and repair. From the two datasets we were able to identify genes that are likely to have regulation roles and regulatory relationships among individual genes. We further combined these associations and constructed regulatory networks to describe the mechanisms by which genes are regulated as a response to an experimental condition. The findings are consistent with what can be found from experimental research.

We believe that association rule mining using DNA Microarray data is a valid analysis method for discovering gene regulation. This method can also be used to construct regulatory networks to describe the mechanism of how genes interact with and regulate one another. As DNA Microarray experiments are designed as exploratory types of experiments, we believe that association rule mining can be used as an exploratory tool for identifying interesting possible gene functions, relationships, and biological pathways that can be further studied by designing experiments targeting these genes.

Chapter 6: Future Work

We have shown some promising results using association rule mining. There are still several issues that need to be addressed.

1. The analysis is yielding a large number of rules. It has been shown that for yeast, there are several hundred genes that exhibit high variation in expression profile even under normal conditions. The expression profile of these genes should be considered as background and subtracted from the original dataset for analysis that is trying to discover gene regulations that are specific to the particular experiment. A possible improvement for the association rule mining method is to remove these genes from the original data during the data preprocessing step.

2. We consider the transformation method we used to represent each gene with three items based on ratio values at each measurement. The change in expression level captured by ratio value is not represented symmetrically. Consider an experiment where we are interested in the gene expression over time. The results (ratios) are relative to expression level at time 0, which is the reference expression level. For example, at time point 1, the expression level is unchanged; at time point 2, the expression level is increased 2-fold; and at time point 3, the expression level is

decreased by 2-fold. So we have the ratio representation of three time points as 1.0, 2.0, and 0.5. When we consider the ratios and their magnitude, the 2-fold up change is twice as significant as the 2-fold down change. In our example, a 2-fold increase (ratio = 2.0) has a change in magnitude of 1.0 (2.0-1.0). A 2-fold decrease (ratio = 0.5) has a change in magnitude of -0.5 (0.5 – 1.0). Even though both ratios represent the change of 2-fold, 1.0 is twice the magnitude as -0.5. In our application, we would like to treat the two changes with the same magnitude, but in opposite directions. This can be accomplished using a log-transformation. In log space, base 2 for example, the data points 1, 2.0, 0.5 become 0, 1.0, and -1.0. The up and down changes are symmetric about 0 (unchanged) with the same magnitude. So the ratio values of greater than 1 will be positive, values between 0 and 1 will be negative, and value of 1 will be 0. In our study, the transformation of each gene into three items is a special case of this approach where all log values are symmetrically, that is, all positive transformed values are represented as 1 or g-up; all negative transformed values are represented as 1 for g-down; and 0 is represented as g-unchanged. We may consider the possibility of representing each gene as more than 3 items. For example, we can use 7 items to represent each gene with different magnitude of change. This transformation will provide a stronger sense of level of regulations of genes rather than a simple up or down.

3. In order to discover generic gene regulations that are not experiment specific, data from multiple DNA Microarray experiments should be considered. Due to the compact nature of FP-Tree, and the fact that the size of an FP-Tree is determined

by the number of items in the transactions, the algorithm should be able to handle a large number of transactions (measurements) without significant increase in resources required. By combining multiple experiment data, we should be able to discover regulations that are independent of experimental condition more easily, thereby yielding higher level biological knowledge.

4. In our study, we applied association rule mining to data collected at the same measurement points, in that a measurement at a time point is considered as a transaction. This may only capture gene regulation at particular time point but not regulations that might occur over time. Because regulatory response often requires time to develop, the gene expression level will change as the response to an experimental condition develops. We may want to generalize our analysis method to discover how the change of expression level of a particular gene over time may regulate another gene. To do this, one possible approach is to first generate an expression motif of a particular gene over all the time or quantity points. Such motifs can be collected from different experiments. The motifs together with data characterizing the experimental conditions can be used in association rule mining to determine the effects of particular motif on other genes.

References

1. K.C. Venter, M.D. Adams and E.W. Myers, *the Sequence of the Human Genome*, Science 2001 Feb;291:1304-51
2. P. Hieter and M. Boguski, *Functional Genomics: It's All How You Read It*, Science, 1997 Oct;278:601-602
3. B.S. Everitt, *Cluster Analysis*. Halsted Press, 3rd edition, 1993.
4. A.K. Jain and R.C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, 1988.
5. D. Fasulo, *An Analysis of Recent Work on Clustering Algorithms*, http:// www.cs.washington.edu/homes/dfasulo/clustering.ps 1999.
6. W. Frawley, G. Piatetsky-Shapiro and C. Matheus, *Knowledge Discovery in Databases: An Overview*, AI Magazine, Fall 1992:213-228.
7. R. Agrawal, T. Imielinski, and A. Swami, *Mining Association Rules between Sets of Items in Large Databases*, Proceedings of the ACM SIGMOD International Conference on Management of Data (ACM SIGMOD 1993), May 1993
8. R. Agrawal and R Srikant, *Fast Algorithms for Mining Association Rules*, Proceedings of the 20th International Conference on Very Large Databases (VLDB 1994), June 1994.
9. S. Brin, R. Motiwani, J.D. Ullman, and S. Tsur, *Dynamic Itemset Counting and Implication Rules for Market Basket Data*, SIGMOD Record (ACM Special Interest Group on Management of Data), 26(2):255, 1997.

10. J. Hipp, U. Güntzer, and G. Nakhaeizadeh, *Algorithms for Association Rule Mining – A General Survey and Comparison*. SIGKDD Explorations. 2(1):58, July 2000.
11. M.J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, *New Algorithms for Fast Discovery of Association Rules*. Proceedings of the 3rd International Conference on KDD and Data Mining (KDD 1997), August 1997.
12. J. Han, J. Pei, and Y. Yin, *Mining Frequent Patterns without Candidate Generation*. Proceedings of the 2000 ACM-SIGMOD International Conference on Management of Data, May 2000.
13. J. Hipp, U. Güntzer, and G. Nakhaeizadeh, *Mining Association Rules: Deriving a Superior Algorithm by Analysing Today's Approaches*. Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery, September 2000.
14. H. Mangalam, J. Steward, J. Zhou, M. Waugh, K. Schlauch, G. Chen, A. Farmer, G. Collelo, J. Weller, *GeneX: an Open Source Gene Expression Database and Integrated Tool Set*. IBM Systems Journal. 2001.
15. D.E. Bassett, M.B. Eisen, and M.S. Boguski, *Gene Expression Informatics – It's All in Your Mine*. Nature Genetics. 21, 51-55 1999.
16. http://www.ncbi.nlm.nih.gov/PMGifs/Genomes/allorg.html
17. http://www.genome.ad.jp/kegg-bin/search_pathway_www?org_name=sce&gene_name=YLR420W
18. B. Dujon. *The Yeast Project: What Did We Learn*? Trends in Genetics 7, 263-270 (1996).

19. Goffeau, B.G. Barrell, H. Bussey, R.W. Davis, B. Dujon, H. Feldmann, F. Galibert, J.D. Hoheisel, C. Jacq, M. Johnston, E.J. Louis, H.W. Mewes, Y. Murakami, P. Philippsen, H. Tettelin and S.G. Oliver, *Life with 6000 genes.* Science 1996;274:546-552.
20. T.R. Hughes, M.J. Marton, A.R. Jones, C.J. Roberts, R. Stoughton, C.D. Armour, H.A. Bennett, E. Coffey, H. Dai, Y.D. He, M.J. Kidd, A.M. King, M.R. Meyer, D. Slade, P.Y. Lum, S.B. Stepaniants, D.D. Shoemaker, D. Gachotte, K. Chakraburtty, J. Simon, M. Bard, and S.H. Friend, *Functional Discovery via a Compendium of Expression Profiles*. Cell 200 Jul;102:109-126.
21. M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein, *Cluster Analysis and Display of Genome-Wide Expression Patterns*. Proceedings of National Academy of Science. 95:14863-14868, (1998)
22. M.B. Eisen, *Cluster and TreeView Manual*. 1998
23. S. Fytlovich, M. Gervais, C. Agrimonti, and B Guiard, *Evidence for an interaction between the CYP1(HAP1) activator and a cellular factor during heme-dependent transcriptional regulation in the yeast Saccharomyces cerevisiae*. EMBO J 12:1209-1218 (1993)
24. T. Prezant, K. Pfeifer, and L. Guarente, *Organization of the regulatory region of the yeast cyc7 gene: multiple factors are involved in regulation*. Mol Cell Biol 7:3253-3259, 1987.

25. I.I. Ouspenski, S.J. Elledge, and B.R. Brinkley, *New yeast genes important for chromosome integrity and segregation identified by dosage effects on genome stability*. Nucleic Acids Research 1999 Aug 1;27(15):3001-8

26. P.J. Wang, A. Chabes, R. Casagrande, X.C. Tian, L. Thelander, and T.C. Huffaker, Rnr4p*, a novel ribonucleotide reductase small-subunit protein*. Molecular and Cell Biology, 1997 Oct;17(10):6114-21

27. S.J. Elledge, and R.W. Davis, *Identification of the DNA damage-responsive element of RNR2 and evidence that four distinct cellular factors bind it*. Molecular and Cell Biology, 1989 Dec;9(12):5373-86.

28. J.B. Allen, Z. Zhou, W. Siede, E.C. Friedberg, and S.J. Elledge, *The SAD1/RAD53 protein kinase controls multiple checkpoints and DNA damage-induced transcription in yeast*. Genes Development, 1994 Oct 15;8(20):2401-15

29. Z. Zhou, and S.J. Elledge, *DUN1 encodes a protein kinase that controls the DNA damage response in yeast*, Cell, 1993 Dec 17;75(6):1119-27.

30. J. P. Bigus, *Data Mining With Neural Networks*, *McGraw Hill*, 1996.

31. P.T. Spellman, G. Sherlock, M.Q. Zhang, V.R. Iyer, K, Anders, M.B. Eisen, P.O. Brown, D. Botstein, and B. Futcher, *Comprehensive identification of cell cycle-regulated genes of the yeast* Saccharomyces cerevisiae *by microarray hybridization*, Molecular Biology of the Cell*,* 9:3273-3297. 1998

32. P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. Lander, and T. Golub, *Interpreting patterns of gene expression with self-organizing maps*. *PNAS*, 96:2907-2912. 1999

33. M.P. Brown, W.N. Grundy, D. Lin, N. Cristianini, C.W. Sugnet, T.S. Furey, M. Ares Jr., and D. Haussler, *Knowledge-based analysis of microarray gene expression data by using support vector machines*. *PNAS*, 4;97(1):262-7. 1/2000