GENOMIC DATA MINING ENHANCED BY SYMBOLIC MANIPULATION OF BOOLEAN FUNCTIONS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Sungroh Yoon
October 2005

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Giovanni De Micheli Principal Advisor

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Russ B. Altman

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Luca Benini

Approved for the University Committee on Graduate Studies.

To Hyeyoung

Abstract

Today, more and more large-scale genomic data sets are being produced by various high-throughput technologies, and genomic data mining has never been more important. Clustering is an unsupervised learning technique that has been popular in data analysis. Although there is mature statistical literature on clustering, new types of genomic data such as gene expression data have sparked development of multiple new methods. Specifically, the technique of *biclustering* refers to a method that performs simultaneous clustering of rows and columns in a data matrix identifying patterns that appear in the form of (possibly overlapping) submatrices. Although this method has some clear advantages over conventional clustering techniques, it has been challenging to develop an efficient biclustering algorithm, since the problem of biclustering is inherently intractable and hard to approximate.

In the first part of this dissertation, a novel biclustering algorithm based upon the symbolic manipulation of Boolean functions is presented. This algorithm exploits the *zero-suppressed binary decision diagrams* (ZBDDs) to implicitly represent and manipulate massive intermediate data that occur in the biclustering process. Leveraged by the ZBDDs, the proposed algorithm can find all the biclusters that satisfy specific input parameters. The second part discusses the application of this algorithm to various genomic data mining tasks such as analyzing gene expression data, linking clinical traits with related genes, and predicting microRNA regulatory modules. The experimental results demonstrate that the proposed method outperforms the alternative techniques tested – in terms of response time, the number of biclusters that can be found, and more importantly, how accurately the discovered biclusters conform to the known biological knowledge.

Acknowledgments

First and foremost, I would like to thank my advisor Professor Giovanni De Micheli. From the very first moment when I knocked his door as a fresh PhD student, to the present day when I am planning my future career, he has never denied me his guidance, support and encouragement. I am greatly privileged to have him as my advisor.

I would also like to thank Professor Russ Biagio Altman for serving as my co-advisor and Professor Luca Benini for serving on my dissertation committee. Without the interaction with these two great mentors, my PhD research would have been severely compromised.

In addition, I gratefully acknowledge Professor Edward J. McCluskey for supervising my research for the Master's degree and Professor Yoshio Nishi for serving as the chair of my oral defense committee. Special thanks also go to Professor and Mrs. Creger for their continuous encouragement.

Additional thanks go to Stanford CAD group members, EPFL LSI people, academic collaborators, and friends. In particular, I would like to thank Eui-Young, Byung-Gon, and Nahmsuk for their invaluable help. I am also greatly indebted to Jerry Yang and Akiko Yamazaki for their vision and generous grant that supported my PhD research.

Last but not least, I would like to thank my wife Hyeyoung and my family (especially Hongseop, Young, Byungsoh, Keumgyou, Hanyoung, Hyejin and Yeonsoo) for their never-ending love and support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivations

High-throughput biology technologies such as DNA sequencing and gene expression measurement by DNA microarrays are producing a vast amount of biological information every day, and many researchers agree that biology is becoming an information science. In traditional biology, researchers usually pose a precise hypothesis and perform well-defined experiments to test the hypothesis. In contrast, in high-throughput biology, discoveries are data-driven, and data lead a hypothesis rather than the reverse.

Breakthroughs in high-throughput biotechnologies have already led to a rapid growth of biological data, both in size and complexity. For example, in recent years the rate at which the GenBank database[1] has grown exceeds the pace set by Moore's Law[2] [73], as seen in Figure 1.1. As more and more biological data emerge, the emphasis progressively switches from the accumulation of data to its interpretation.

The science of extracting useful information from large data sets or databases is known as *data mining*, which is one component in the area of machine learning and adaptive computation [37]. Defined more specifically, data mining is the analysis of

[1] `http://www.ncbi.nlm.nih.gov/Genbank`

[2] The empirical observation that at our rate of technological development, the complexity of an integrated circuit, with respect to minimum component cost, will double in about 18 months.

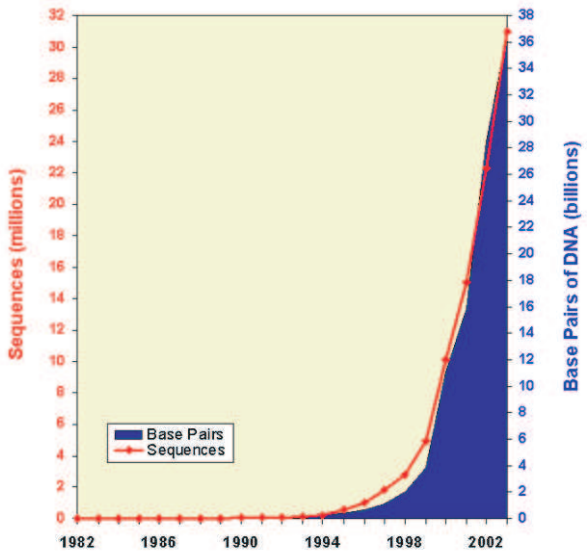


Figure 1.1: Growth of GenBank database. The growth rate exceeds the pace set by Moore's Law [73].

(often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner [37].

Large databases of biological information create both challenging data mining problems and opportunities for researchers in the field. In this regard, conventional computer science algorithms have been useful, but are increasingly unable to address many of the most interesting analysis problems. This is due to the inherent complexity of biological systems as well as our lack of a comprehensive theory of life's organization at the molecular level [7]. Machine learning approaches, on the other hand, are ideally suited for domains characterized by the presence of large amounts of data, "noisy" patterns, and the absence of general theories [7]. The fundamental idea behind these approaches is to learn the theory automatically from the data, through a process of inference, model fitting, or learning from examples.

The field of machine learning typically distinguishes three cases: supervised, unsupervised, and reinforcement learning [85]. Cluster analysis, or *clustering*, is an unsupervised learning technique that has been one of the most popular in a variety

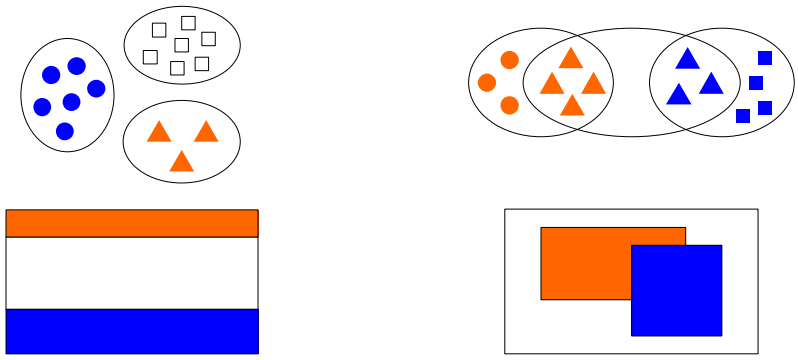
of disciplines including statistics, computer science, electrical engineering, biology, social science, and many others. Clustering is to group a set of objects into subsets, or *clusters*, such that those within each cluster are more closely related to one another than objects assigned to different clusters [41].

Although there is mature statistical literature on clustering, high-throughput genomic data sets have sparked the development of multiple new methods [78]. First of all, genomic data sets can have very high dimensions. When we analyze high-dimensional data, it is more difficult for a cluster to form, because we lose the meaning of clusters due to many irrelevant attributes. Also, it is harder to find a cluster due to *the curse of dimensionality* or lack of data separation in high-dimensional space. In addition, typical genomic data sets often exhibit different characteristics from traditional clinical data sets. For instance, the number of variables involved in a typical genomic study is far more than the number of the observations, in contrast to a typical clinical study where there are normally more observations than variables [52]. Thus, in typical genomic studies we often encounter the curse of dimensionality as well as the problem of identifying a highly underdetermined system.

Among the methods that have been proposed to handle this challenge, one of the most natural and effective approaches is to focus only on subsets of the entire data set [6]. Furthermore, by performing simultaneous clustering of rows and columns in a data matrix, some clustering techniques can discover important local structures appearing in the form of (possibly overlapping) submatrices of the matrix. In the literature [10,18,21,35,51,56,74,90,98,107,113,115,120], this method has been referred to as many different names such as *biclustering*, *co-clustering*, *conjugate clustering*, *module finding*, *block clustering*, or the *information bottleneck method*, to name just a few[3]. In this dissertation, only the terms biclustering and co-clustering will be used.

As informally compared in Figure 1.2, an object involved in biclustering can belong to multiple biclusters, unlike in conventional clustering. Thus, the biclustering technique may provide additional biological insight that has been overlooked by traditional clustering approaches. For example, biclustering is more compatible with

[3]The interested reader is directed to [65] for a comprehensive review of the biclustering technique for biological applications.



(a) Clustering (b) Biclustering

Figure 1.2: Informal comparison between conventional clustering and biclustering. (a) Objects are partitioned into mutually exclusive groups. (b) Objects are allowed to belong to multiple groups.

our understanding of cellular processes: we expect subsets of genes to be co-regulated and co-expressed under certain experimental conditions, but to behave almost independently under other conditions [10]. The biclustering method may be useful in recognizing reusable genetic "modules" that are mixed and matched in order to create more complex genetic responses [6]. In gene expression analysis, the biclustering technique is therefore more suitable for cases where genes have multiple functions and experimental conditions are diverse.

Despite its potential effectiveness, the problem of biclustering is often computationally challenging. Let A be a matrix with row set R and column set C[4]. The matrix A can be converted to a weighted bipartite graph $G = (V, E)$, where the vertex set $V = R \cup C$ and the edge set E consists of edge $\{i, j\}$ connecting row $i \in R$ and column $j \in C$ with weight a_{ij}. A submatrix of A then corresponds to a biclique in the graph G. To find not just any submatrix but a useful one, we need to consider individual elements of a submatrix, or equivalently the edge weights of a biclique. Moreover, in order to avoid redundancy, we usually focus on finding maximal submatrices. Therefore, the problem of discovering patterns with certain semantics is at least as difficult as that of finding the maximum edge biclique in a bipartite graph, a

[4]For instance, in a gene expression data set, R and C typically correspond to a set of genes and a set of experimental conditions, respectively.

problem known to be NP-complete [65, 76]. Additionally, there is evidence that the maximum edge biclique problem is difficult to approximate [33].

To cope with this computational challenge, this dissertation proposes a novel biclustering method that exploits the techniques for the symbolic manipulation of Boolean functions. These techniques have been intensively studied in the field of design and verification of *very large-scale integration* (VLSI) digital circuits and reported to be useful for solving many practical instances of intractable problems [15, 16, 26, 67, 88]. In particular, the proposed method employs the *zero-suppressed binary decision diagrams* (ZBDDs) [69, 70] to implicitly represent and manipulate massive intermediate data occurring in the data mining process. This dissertation will describe this ZBDD-based biclustering method as well as its application to several computational genomics problems.

1.2 Contributions

The objective of this study was to develop an efficient and flexible biclustering method that can be applied to various types of large-scale data mining problems in genomics. Compared with previous biclustering approaches (see Section 2.3.3 for related work), the proposed method is unique in that it is an exact *and* scalable approach. More specifically, the contributions of this dissertation include the following:

- A unified problem formulation that can encompass a large spectrum of biclusters [115]: It was first discovered by this study that many definitions of biclusters in the literature share a common property. A bicluster is *nested* if any sub-bicluster of this bicluster is yet another bicluster under the same definition and condition. On top of this formulation, this dissertation describes a biclustering method that can find any type of nested biclusters. Many types of biclusters (see [10, 18, 61, 74, 112, 113] for some examples of nested biclusters) are in fact nested biclusters and can thus be found by the proposed method with minor modifications. Furthermore, this unifying method can find nested biclusters more efficiently than alternative techniques.

- A ZBDD-based biclustering algorithm that is flexible, scalable, and exact [115, 119, 120]: This dissertation describes a novel biclustering method that exploits the ZBDD, a compact data structure to represent massive sets. As stated above, this algorithm is flexible in that it can be applied to a variety of biclustering problems. In addition, this method is scalable to real genomic data sets of non-trivial size. Moreover, the proposed technique is exact in that it can find all the biclusters that satisfy specific input conditions.

- Applications in computational genomics: The proposed biclustering method was successfully applied to several genomic data mining tasks, including analyzing gene expression data [116, 119, 120], linking gene expression with clinical traits [114], and predicting *microRNA* (miRNA) regulatory modules [117, 118]. The experimental results demonstrate that the proposed method outperforms the alternative techniques tested – in terms of response time, the number of biclusters that can be found, and more importantly, how accurately the discovered biclusters conform to the known biological knowledge.

1.3 Assumptions and limitations

This work assumes that the input data set is represented by a two-dimensional matrix of real numbers. Although not every biological data set can be represented by a matrix, this assumption can often be justified. After readout and preliminary data processing, biological data produced by high-throughput technologies are typically arranged in a matrix. For example, DNA microarray data sets are one of the most well-known and widely available data sets in a matrix format. Chapter 2 covers more examples of genomic data represented by a matrix.

As shown by the experimental studies in Chapters 5, 6, and 7, the response time of the proposed algorithm run on typical benchmarks is practical. However, the problem investigated by this study is inherently intractable, meaning that it is unlikely that an efficient algorithm to find an optimal solution in polynomial time exists.

Finally, the experiments presented in this dissertation were performed and verified

in silico. Further investigation through "wet lab" experiments may be needed. To this end, some experiment design strategies are included in Section 7.4.1.

1.4 Perspectives

Essentially, the proposed method performs clustering on two distinct sets of objects simultaneously and finds hidden local patterns. Each pattern is represented by two sets, each of which corresponds to a subset of one of the two input sets. The two sets in a pattern are linked to each other with respect to a given input condition.

From a computer science perspective, the proposed technique is unique in the sense that it aims at finding all the possible patterns without sacrificing its scalability to practical problems. From a biomedical research perspective, this method can also be a versatile tool for quantitative analysis.

For instance, in gene expression data analysis, the two input sets consist of a set of genes and a set of experimental conditions. The computational method proposed in this dissertation allows us to find out which genes are related to the experimental conditions of specific interest. Given that DNA microarray technology [27, 62] allows us to monitor transcription levels of thousands of genes simultaneously, our method can help researchers perform a variety of analysis tasks such as annotating gene functions, diagnosing disease conditions, and characterizing effects of medical treatments.

Depending upon what the two input sets are, the proposed technique can provide various kinds of insight. For example, when the two input sets consist of a set of genes and a set of clinical traits, the approach explained here can help us understand how the specific clinical traits and the potentially responsible genes are linked. This can contribute to medical diagnosis and prognosis. As another example, when the two input sets consist of a set of *messenger RNA* (mRNA) and a set of its regulator, the proposed method can allow us to find a group of mRNAs and their regulators that are believed to participate in biological processes cooperatively. This may provide critical information for reconstructing gene regulatory networks. The application of the proposed computational method is not limited to these examples, and many other

interesting problems in computational genomics can be approached by the various techniques explained in this dissertation.

1.5 Organization

This dissertation consists of three major parts. First, Chapter 2 covers the background materials. This chapter is intended to help the reader better understand the subsequent development of algorithms and applications as well as to put the work described in this thesis in a proper context.

The second part contains Chapters 3 and 4, and these chapters describe the proposed computational method at length. Chapter 3 covers the ZBDD-based biclustering algorithm for gene expression data analysis. Chapter 4 presents a generalized version of this algorithm. This extended algorithm can be applied to a wide variety of data sets.

The last part covers the applications of the proposed method. Chapter 5 describes an application to gene expression data analysis. The proposed method and alternative techniques are compared in various criteria. Discovered biclusters are evaluated in terms of biological and statistical significance. Chapter 6 describes how to link clinical traits with the genes possibly responsible for these traits using the proposed method. An intermediate data set is constructed from a gene expression matrix and a matrix of clinical traits. Biclusters are found from this intermediate data set. A survey of the literature and validation with Gene Ontology suggest that these biclusters can provide meaningful biological insight. Chapter 7 covers the prediction of microRNA regulatory modules. MicroRNAs are a novel class of gene products that repress mRNA translation or mediate mRNA degradation in a sequence-specific manner [8]. Using the proposed method, we can predict groups of miRNAs and target mRNAs that are believed to participate cooperatively in post-transcriptional gene regulation.

Finally, Chapter 8 concludes this dissertation. This chapter presents a summary and directions for future research.

Chapter 2

Background

This chapter aims at providing some background knowledge helpful to understand the algorithms and their applications to be presented in subsequent chapters. In Section 2.1, fundamentals of basic molecular biology are reviewed. In particular, this section explains the building blocks of genetic information flow in cells and how the flow is controlled. Section 2.2 presents high-throughput biological data acquisition technologies that have caused a focus of research in biology to move from the realm of traditional experimental science to that of information science. Section 2.3 shows computational means to interpret a large volume of accumulated biological data and extract useful information. Especially, an overview of data mining and machine learning approaches is provided. Typically, mining large-scale data is computationally intensive. To cope with this challenge, a method that exploits the symbolic manipulation of Boolean functions will be developed in later sections. In order to facilitate the explanation of this method, Section 2.4 introduces fundamental concepts and some examples of the symbolic Boolean function representation and manipulation.

2.1 Biological foundations

This section begins with a brief overview of fundamental concepts in molecular biology with emphasis on the flow of genetic information and its regulation. More details on this subject can be found in [3, 12, 39, 63].

2.1.1 The flow of genetic information

The flow of genetic information in normal cells is from *deoxyribonucleic acid* (DNA) to *ribonucleic acid* (RNA) to protein (Figure 2.1). The synthesis of RNA from a DNA template is called *transcription*, whereas the synthesis of a protein from an RNA template is termed *translation*. This two-step process has been called the *central dogma* of molecular biology.

DNA DNA is a linear polymer made up of four different monomers. As shown in Figure 2.2, each unit of the polymeric structure is composed of a sugar (deoxyribose), a phosphate, and a variable base (adenine, cytosine, guanine or thymine) that is attached to the sugar-phosphate backbone. The bases can be arranged in any order along a strand of DNA. The sequence of bases constitutes the genetic information that specifies which proteins an organism will make as well as when and where protein synthesis will occur.

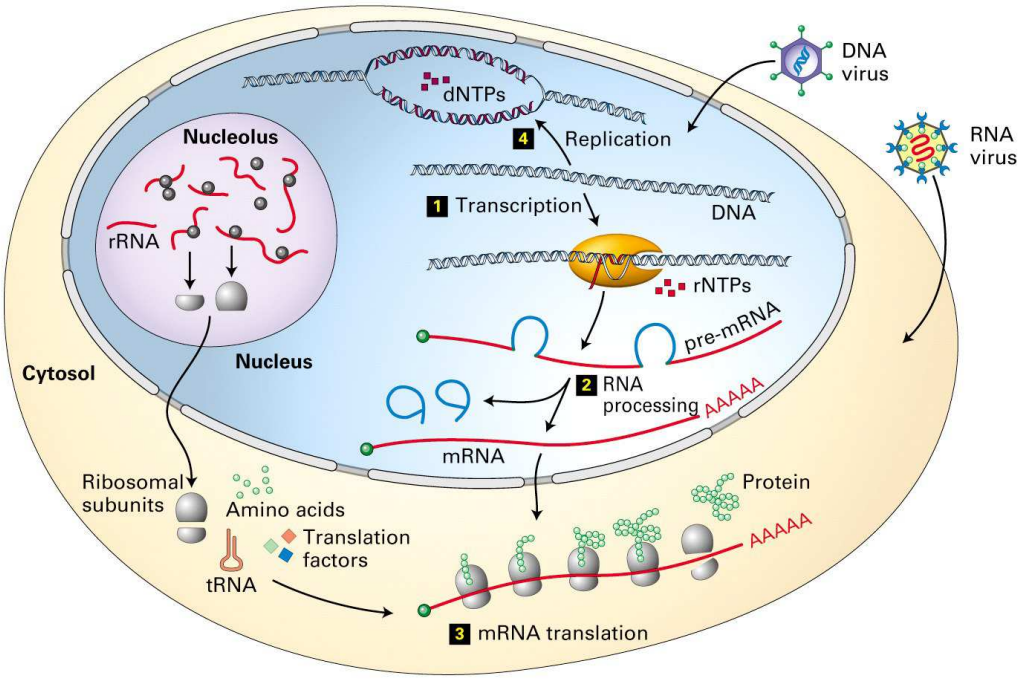
Two single strands of DNA typically combine to form a *double helix*, in which the sugar-phosphate backbones of the two chains are intertwined and the bases form *specific base pairs* (bp). By hydrogen bonds, adenine pairs with thymine (A-T) and guanine pairs with cytosine (G-C).

The DNA regions that encode proteins are called *genes*, and *chromosomes* are organelles that package and manage the storage, duplication, expression, and evolution of DNA. The entire collection of chromosomes in each cell of an organism is its *genome*.

RNA RNA is an intermediate[1] in the flow of genetic information from DNA, the hereditary material, to protein, the primary functional molecules of the cell. The DNA is first transcribed into *messenger RNA* (mRNA), which is then translated into protein. In addition to the role as the intermediate in the flow of information, RNA has many critical roles within a cell. For example, *ribosomal RNA* (rRNA) is one of the structural components of the molecular machinery called *ribosomes*, in which

[1]Some viruses use RNA as the genetic material.

(a)



(b)

Figure 2.1: (a) The central dogma of molecular biology states the flow of genetic information in its simplest form, which is from DNA via RNA to protein. (b) A more realistic illustration of the flow of genetic information in a cell [63].

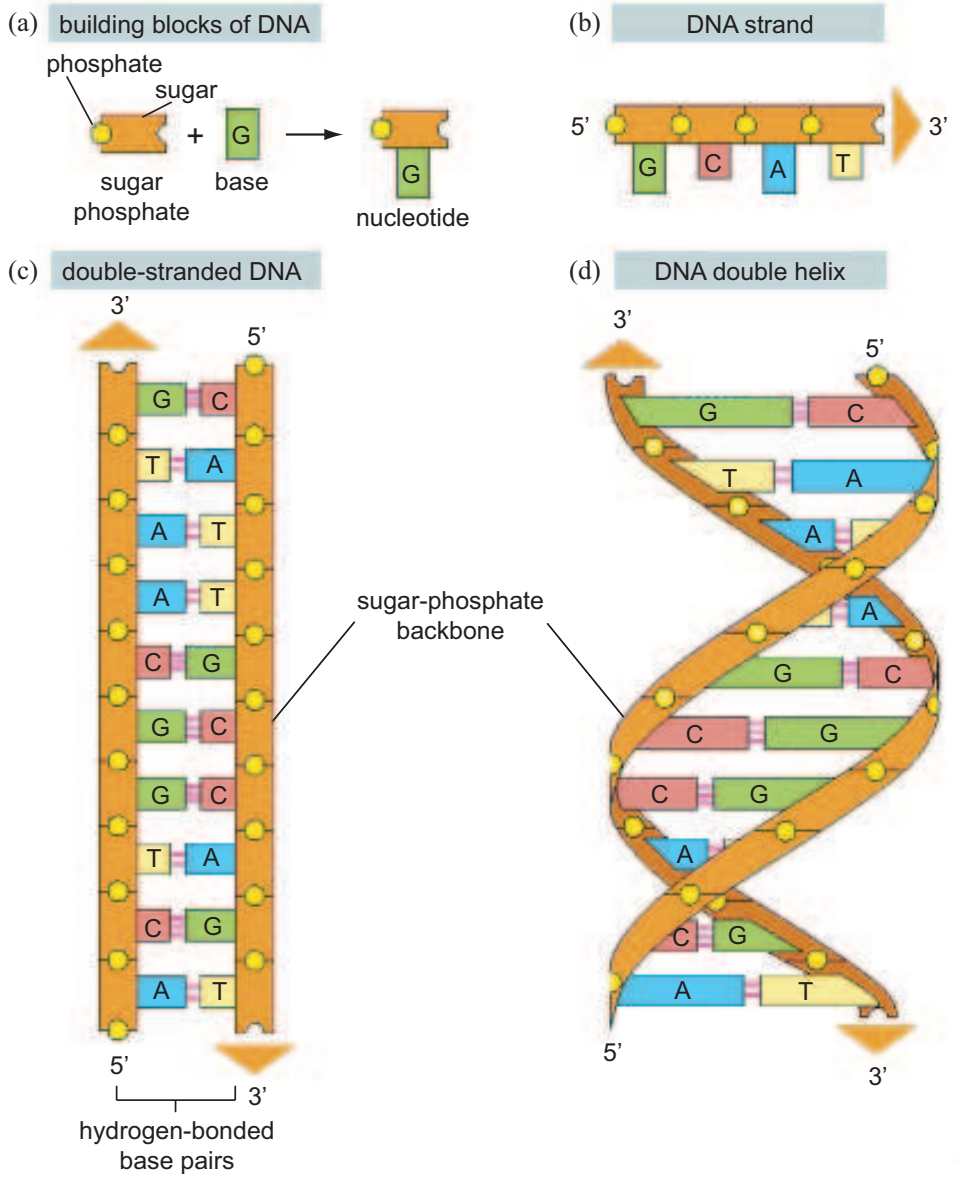


Figure 2.2: DNA and its building blocks [3]. DNA is made of four types of nucleotides, which are linked covalently into a polynucleotide chain (a DNA strand) with a sugar-phosphate backbone from which the bases (A, C, G, and T) extend. A DNA molecule is composed of two DNA strands held together by hydrogen bonds between the paired bases. The *arrowheads* at the ends of the DNA strands indicate the polarities of the two strands, which run antiparallel to each other in the DNA molecule. In the diagram at the bottom left of the figure, the DNA molecule is shown straightened out; in reality, it is twisted into a double helix, as shown on the right.

the translation process takes place. Recently, a new class of RNA called small *non-coding RNA* (ncRNA) has been identified and reported to play a crucial role in gene regulation. Section 2.1.3 provides more details.

The structure of RNA is similar to that of DNA. RNA is a linear polymer composed of a sugar-phosphate backbone and four types of bases are attached to it. The sugar in RNA is ribose, and one of the bases is uracil instead of thymine. In contrast to DNA, an RNA molecule typically exists as a single strand, although double-stranded segments can appear with the intrastrand base-pairing of A-U and G-C.

Protein Proteins are the building blocks of cells and participate in virtually all processes within cells. Proteins are linear polymers formed from a selection of 20 building blocks, called *amino acids*. Three bases along a DNA strand encode a single amino acid according to the genetic code. The complete genetic code and 20 amino acids are shown in Figure 2.3. The genetic code is redundant but not ambiguous.

The amino acid sequence of a protein entirely dictates its three-dimensional structure, which determines the functional properties of the protein. The self-folding nature of proteins enables the transition from the one-dimensional world of sequence information to the three-dimensional world of biological function.

2.1.2 Gene regulation

Gene activity is controlled first and foremost at the level of transcription. Much of this control is achieved through the interplay between proteins that bind to specific DNA sequences and their DNA-binding sites.

In prokaryotes[2], many genes are clustered into *operons*, which are units of coordinated genetic expression. An operon consists of regulatory elements (an operator and a promoter[3]) and structural genes. In addition, regulator genes encode proteins that interact with the operator and promoter sites to stimulate or inhibit transcription. Some proteins activate transcription by directly contacting RNA polymerase.

[2]Unicellular organisms such as bacteria, which lack a nucleus.

[3]DNA sequences near the beginning of genes that indicates where to begin transcription

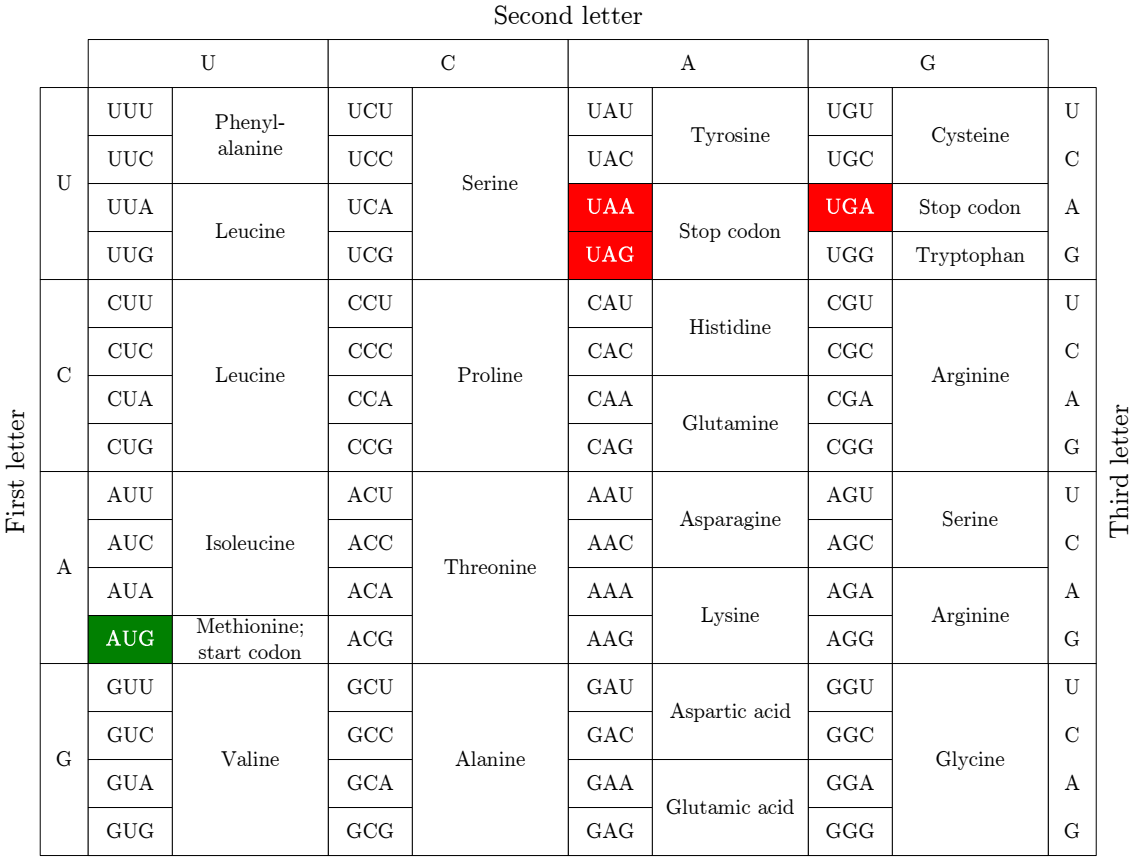


Figure 2.3: The universal genetic code. Genetic information is encoded in mRNA in three-letter units (codons) made up of the bases uracil (U), cytosine (C), adenine (A) and guanine (G).

The greater complexity of genomes in eukaryotes[4] requires elaborate mechanisms for gene regulation. For example, cis-acting regulatory elements are regions of DNA sequence that lie nearby to the gene they control. Because these sequences are always on the *same* molecule of DNA as the gene they regulate, they are referred to as *cis*-acting elements. The promoter is a cis-acting element typically lies directly adjacent to the gene that it controls. Enhancers are another class of cis-acting elements that can sometimes lie thousands of base pairs away from a gene. Besides, trans-acting genetic elements encode protein products called *transcription factors*, which interact with cis-acting elements. A trans-acting element is called *trans*-acting because it may be encoded by a gene on a DNA molecule *other* than that containing the gene being regulated.

Gene expression can also be regulated at the level of translation. Attenuation is a prokaryotic mechanism for regulating transcription through modulation of nascent RNA secondary structure. In eukaryotes, genes encoding proteins that transport and store iron are regulated at the translational level.

2.1.3 Small non-coding RNAs

In addition to the traditional gene regulatory mechanisms previously explained, recent discoveries indicate that a class of small *non-coding RNAs* (ncRNAs) may control gene expression at the post-transcriptional level. These small ncRNAs mainly consist of two types, *small interfering RNAs* (siRNA) and *microRNAs* (miRNAs), and mediate gene regulation through *RNA interference* (RNAi) pathways. This section provides a short introduction to microRNAs. The interested reader is directed to [8].

MicroRNAs are endogenous 21-22-nucleotide ncRNAs that can play crucial regulatory roles in animals and plants by targeting transcripts for cleavage or translational repression. In contrast to the conventional mRNAs which will be translated into proteins, miRNAs do not generate proteins. Instead, they regulate the expression of protein-coding genes by binding to partially complementary 3' *untranslated region* (UTR) of their mRNAs and suppressing their translation without degrading the

[4]Most organisms that have a well-defined nucleus within each cell.

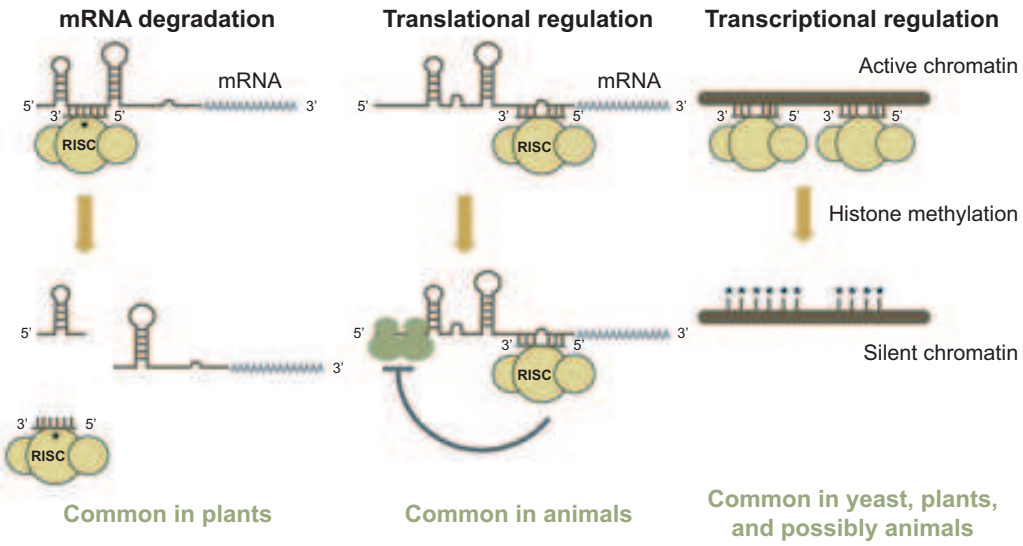


Figure 2.4: MicroRNAs are evolutionarily conserved, small noncoding RNA molecules found in eukaryotes that regulate gene expression at the level of translation. (`http://ambion.com/techlib/resources/miRNA/mirna_fun.html`)

target mRNAs. Figure 2.4 illustrates the mode of action of miRNAs in plants and animals.

MicroRNAs have been shown to play important roles in animal and plant development. In mammals, some miRNAs have been shown to regulate hematopoietic lineage differentiation, insulin secretion, and adipocyte differentiation. Hundreds of different miRNAs have now been isolated or computationally predicted in complex eukaryotes, and it has been estimated that microRNAs constitute nearly 1% of the genes in the worm, mouse, and human genomes. Given the abundance and evolutionary conservation in sequences in animal and plant genomes, miRNA are expected to play important roles in a wide range of gene regulatory pathways.

Some details on bioinformatics approaches to miRNA studies can be found in Section 7.2.1.

2.2 High-throughput biology

This section provides examples of high-throughput biology such as DNA sequencing and gene expression measurement by DNA microarrays.

2.2.1 DNA sequencing

DNA sequencing is the determination of the precise sequence of bases in a sample of DNA. Sequencing has yielded a wealth of information concerning gene architecture, the control of gene expression, as well as protein structure. One major challenge is that there is no machine that takes long DNA as input and provides the complete sequence as output. That is, a typical system can only sequence a few hundred up to a thousand bases at a time. To sequence the entire genome, the DNA is first cut into smaller pieces, and each piece is sequenced separately. The resulting set of sequences are then computationally assembled into one contiguous genome. Today DNA sequencing is highly automated, and it has become almost routine to sequence whole genomes.

A related technology is the *polymerase chain reaction* (PCR), which leads to a billionfold amplification of a segment of DNA [12]. One molecule of DNA can be amplified to quantities that permit characterization and manipulation. This powerful technique is being used to detect pathogens and genetic disease.

2.2.2 Gene expression measurement

Most genes are present in the same quantity in every cell. However, the level at which a gene is expressed, as indicated by mRNA quantities, can vary widely. Gene-expression patterns vary from cell type to cell type. Even within the same cell, gene expression levels may vary as the cell responds to changes in physiological circumstances. Using our knowledge of complete genome sequences, it is now possible to analyze the pattern and level of expression of all genes in a particular cell type or tissue.

One of the most powerful methods for this purpose developed to date is based

on DNA hybridization and is called the *DNA microarray* or *gene chip* [27, 62]. This technique enables us to monitor the expression levels of a large number of genes simultaneously, providing a global view of gene expression information of the organism under study [6, 52, 78]. Depending upon the specific technology used, DNA microarrays can reflect either absolute expression levels (*e.g.,* Affymetrix GeneChip® arrays [62]) or relative expression ratios (*e.g.,* cDNA microarrays [27]).

A DNA microarray consists of a solid substrate and arrays of probes arranged in grids. Each *probe* is composed of many identical copies of single-stranded DNAs with a specific gene sequence. In the cDNA microarray technology, probes are fabricated by depositing cDNAs or previously synthesized oligonucleotides. These probes are then chemically processed to keep them unfolded. In Affymetrix GeneChip® arrays, probes are synthesized directly on the substrate by light-directed chemical synthesis carried out with photolithographic microfabrication techniques used in the semiconductor industry (see Figure 2.5 for a detailed explanation of the probe synthesis in Affymetrix GeneChip® arrays). The messenger RNA extracted from a particular biological sample under study is transformed into synthetic DNA called *complementary DNA* (cDNA). These cDNA molecules, termed *targets*, are labeled with fluorescent tags and hybridize with the probes on a DNA microarray. The amount of fluorescence emitted by each spot on the DNA microarray will be proportional to the amount of mRNA produced from the gene with the DNA sequence identical to the sequence in the spot. Finally, the hybridized microarray is scanned to create an image, and the light intensity of each spot on the image is converted into a numerical value, which represents the expression level of a gene under a given condition.

2.2.3 Miniaturized biochips

Microfluidics-based biochips are becoming increasingly popular for DNA analysis, clinical diagnostics, and the detection and manipulation of bio-molecules. These systems automate highly repetitive laboratory tasks by replacing cumbersome equipment with miniaturized and integrated systems. Thus, they are able to provide ultra-sensitive detection at significantly lower costs per assay than traditional methods. For instance,

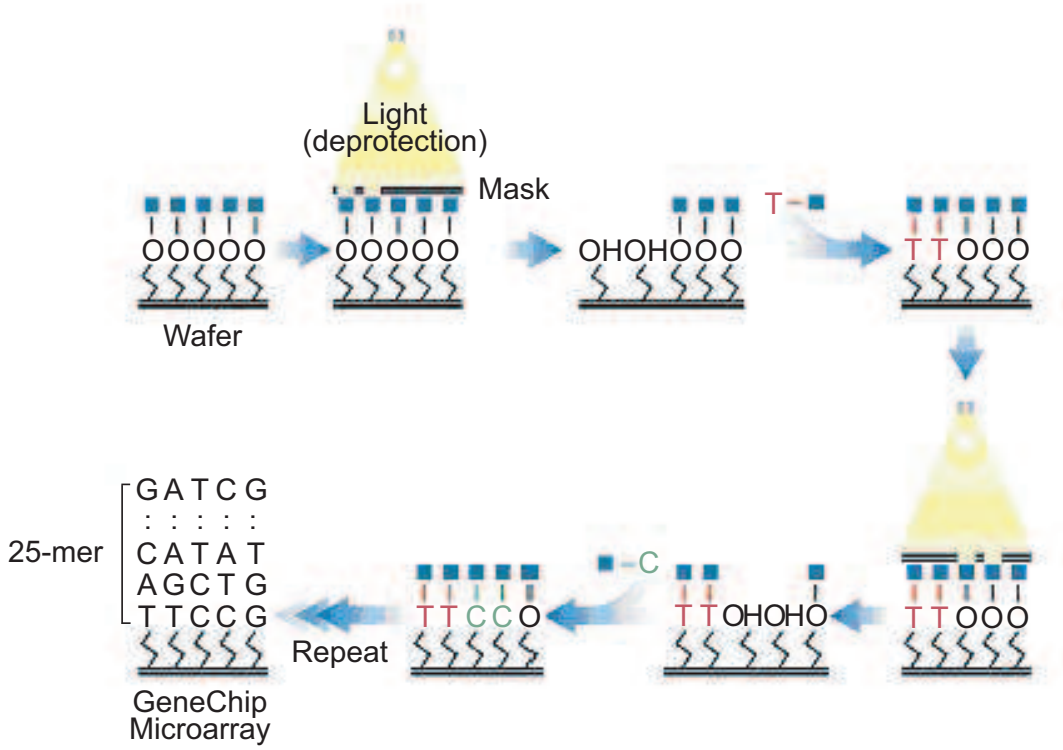


Figure 2.5: Affymetrix uses a combination of photolithography and combinatorial chemistry to manufacture GeneChip® arrays (`https://www.affymetrix.com/technology/manufacturing/index.affx`). Probe synthesis occurs in parallel, resulting in the addition of an A, C, T, or G nucleotide to multiple growing chains simultaneously. To define which oligonucleotide chains will receive a nucleotide in each step, photolithographic masks, carrying 18 to 20 square micron windows that correspond to the dimensions of individual features, are placed over the coated wafer. The windows are distributed over the mask based on the desired sequence of each probe. When ultraviolet light is shone over the mask in the first step of synthesis, the exposed linkers become deprotected and are available for nucleotide coupling. Critical to this step is the precise alignment of the mask with the wafer before each synthesis step. The nucleotide attaches to the activated linkers, initiating the synthesis process. In the following synthesis step, another mask is placed over the wafer to allow the next round of deprotection and coupling. The process is repeated until the probes reach their full length, usually 25 nucleotides.

CMOS-based integrated biosensor arrays [40,89] have been recently developed for gene expression measurement, and the scale of integration will continue to grow.

Another technique called the *tissue microarray* (TMA) enables researchers to extract small cylinders of tissue from histological sections and arrange them in a matrix configuration on a recipient paraffin block such that hundreds can be analyzed simultaneously [20, 91]. TMA thus allows the rapid and cost effective validation of novel markers in multiple pathological tissue specimens.

Another example, the protein microarray, is a crucial biomaterial for the rapid and high-throughput assay of many biological events where proteins are involved. In contrast to the DNA microarray, it has not been sufficiently established because of protein instability under the conventional dry conditions [50]. However, protein microarrays will eventually reveal vast amount of information essential to the understanding of gene functions and products.

Other examples include the fluorescence-based multiplexed cytokine immunoassays [106] and ligand chips [84]. In particular, using the cytokine chip, cytokine expression in breast cancer cells were examined and the chemokines associated with human cervical cancers were successfully identified [106].

2.3 Biological data analysis and mining

In this section, an overview of machine learning technique is provided, followed by discussions on the challenges encountered when we analyze large-scale genomic data sets. Previous work on biclustering is also presented. More details on the material covered in this section can be found in [5, 7, 37, 41, 85].

2.3.1 Overview of machine learning

The field of machine learning typically distinguishes three cases: supervised, unsupervised, and reinforcement learning [85]. The problem of *supervised learning* involves learning a function from examples of its inputs and outputs. It makes sense to think of using the inputs to predict the output. The distinction in output type has led to

a naming convention for the prediction task: *regression* when we predict quantitative outputs, and *classification* when we predict qualitative outputs. These two tasks have much in common, and in particular both can be viewed as a task in function approximation.

In *unsupervised learning*, we concern about learning patterns in the input when no specific output values are supplied. The aim is to find the regularities in the input. There is a structure to the input space such that certain patterns occur more often than others, and we want to see what generally happens and what does not. In statistics, this is called *density estimation*. One method for density estimation is *cluster analysis* or clustering where the aim is to decompose or partition a data set into groups so that the points in one group are similar to each other and are as different as possible from the points in other groups. Besides this, association rule analysis has emerged as a popular tool for mining commercial databases. The goal is to find joint values of the variables that appear most frequently in the database.

The task of *reinforcement learning* is the most general of the three categories. Rather than being told what to do by a teacher, a reinforcement learning agent must learn from reinforcement or reward. In some applications, the output of the system is a sequence of *actions*. In such a case, a single action is not important; what is important is the *policy* that is the sequence of correct actions to reach the goal. The machine learning program should be able to assess the goodness of policies and learn from past good action sequences to be able to generate a policy.

2.3.2 Challenges in large-scale data analysis

One of the most frequently encountered problems is the curse of dimensionality [9]. In essence, this means that the amount of data needed to sustain a given spatial density increases exponentially with the dimensionality of the input space. Figure 2.6 shows an example [41] of the curse of dimensionality, in which we are considering a subcubical neighborhood for uniform data in a unit cube. When the number of dimension is one ($d = 1$ in Figure 2.6(b)), covering 10% of the range of the input variable is sufficient to capture 10% of the data to form a local average. However,

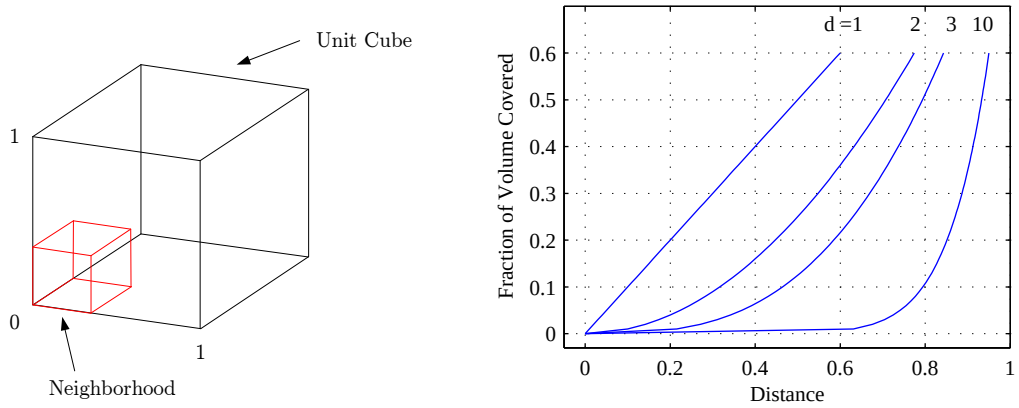


Figure 2.6: The curse of dimensionality is well illustrated by a subcubical neighborhood for uniform data in a unit cube [41]. The plot in (b) shows the fraction of the volume of the data captured by considering the side-length of the subcube for different dimensions d.

when $d = 10$, we must cover nearly 80% of the range of each input variable in order to capture 10% of the data. Such neighborhoods are no longer "local," and this will adversely impact any method based on spatial density (e.g., k-nearest-neighbor averaging and many others), and algorithms as well as our intuition breaks down in high dimensions, unless the data follows certain simple distributions.

In addition to the curse of dimensionality, we often encounter in typical genomic studies the problem of identifying a highly underdetermined system [52], as already mentioned in Chapter 1. Typical genomic data often exhibit different characteristics from traditional clinical data. For instance, as seen in Figure 2.7, the number of variables involved in a typical genomic study is far more than that of the observations, in contrast to a typical clinical study where there are normally more observations than variables [52]. This may be one reason why multiple new methods have been newly proposed to analyze genomic data despite the mature literature on traditional clinical data analysis.

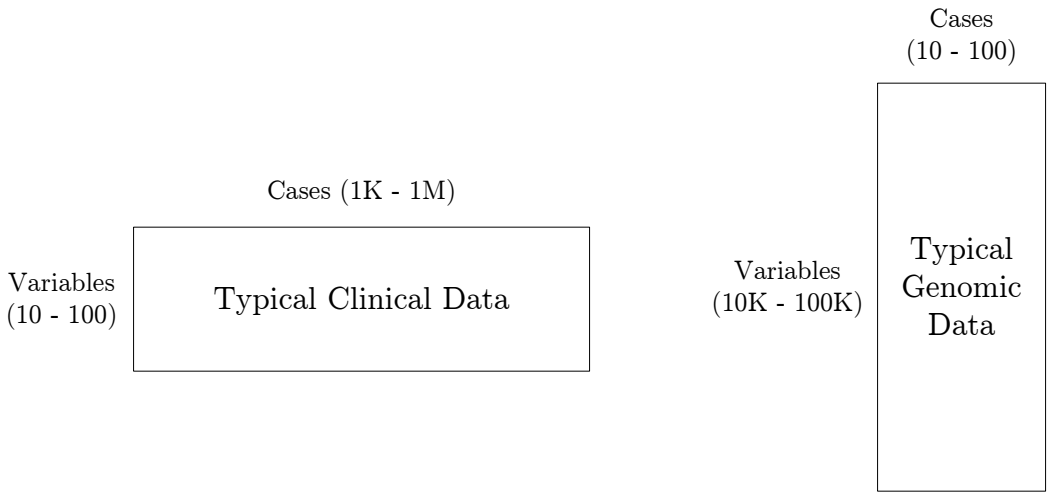


Figure 2.7: A major difference between classic clinical studies and genomic studies [52]. In contrast to clinical data, genomic data often results in a highly underdetermined system.

2.3.3 Previous work on biclustering

As introduced in Section 1.1, biclustering is a clustering method that can find local structures appearing in the form of (possibly overlapping) matrices in an unsupervised fashion. The elements of a bicluster show *similar behavior*. Depending upon the biclustering method used, the definition of this "similar behavior" varies. According to Madeira and Oliveira [65], we can identify four major classes of biclusters: (1) biclusters with constant values; (2) biclusters with constant values in rows (genes) or columns (experiments); (3) biclusters with coherent values; and (4) biclusters with coherent evolutions.

Califano et al. [18] modeled a bicluster with constant rows by the δ-valid kj-pattern, a $k \times j$ matrix in which the maximum and minimum values of each row differ by less than δ. Wu et al. [112] proposed a similar definition of biclusters in which every gene is expressed within a small range δ across all experimental conditions. Both methods aimed at finding *maximal* biclusters, in the sense that they are not contained by other biclusters of the same type.

The δ-biclustering approach by Cheng and Church [21] employed the concept of a *residual* to find a bicluster with coherent values. In *analysis of variance* (ANOVA) models, a residual is the difference between an actual value and the mean score for the group or category from which that value was taken [80, 93]. Thus, a low value of residual can show a high degree of coherence, while a high value reveals the opposite. The residual of element a_{ij} in the matrix A denoted by a pair of sets (I, J) is

$$r_{ij} = a_{ij} - \overline{a}_{i\bullet} - \overline{a}_{\bullet j} + \overline{a}_{\bullet\bullet}, \tag{2.1}$$

where $\overline{a}_{i\bullet}$ is the mean of the ith row, $\overline{a}_{\bullet j}$ the mean of the jth column, and $\overline{a}_{\bullet\bullet}$ the mean of all elements in A. The pair (I, J) specifies a *δ-bicluster* if the following *mean squared residual (MSR)* of the elements in A is lower than δ, a given threshold:

$$MSR(I, J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} r_{ij}^2. \tag{2.2}$$

The pClustering technique by Wang et al. [107] also aimed at finding a bicluster with coherent values. The matrix A denoted by pair (I, J) is called a *δ-pCluster* if the value of $|x - z - y + w|$ is lower than some δ for any 2×2 submatrix $\begin{bmatrix} x & y \\ z & w \end{bmatrix}$ in A.

Some biclustering algorithms seek to find biclusters with coherent evolutions across the rows regardless of their exact numerical values. Ben-Dor et al. [10] looked for *order-preserving submatrices* (OPSMs), in which the expression levels of all genes induce the same linear ordering of the experiments. An OPSM represents a bicluster with coherent evolutions on its columns, and they wanted to find large OPSMs. Murali and Kasif [74] proposed a representation for gene expression data called *conserved gene expression motifs* (xMOTIFs). A xMOTIF is a subset of genes that is simultaneously conserved across a subset of experimental conditions. They assumed that the expression level of a gene is conserved across some experimental conditions if the gene is in the same "state," or a range of expression levels, under each different condition. They aimed at finding the largest xMOTIF.

As already stated in Chapter 1, the biclustering problem is inherently intractable.

Thus, most biclustering approaches employ some heuristics to reduce this computational burden. However, these algorithms can provide only a partial solution in that only some of the possible biclusters from a given data set can be found. In contrast, some biclustering methods are exact algorithms in that they aim at finding all possible biclusters. However, these exact approaches suffer from the problem of combinatorial explosion and are typically not scalable to practical data sets. To address this issue, Chapters 3 and 4 will describe a novel biclustering method that are exact as well as scalable to large problems.

2.4 Symbolic manipulation of Boolean functions

Machine-learning methods are computationally intensive and benefit greatly from progress in computer speed. Chapters 3 and 4 will describe a method to implicitly represent and manipulate massive data represented by Boolean functions. This method is based upon an efficient data structure called the *zero-suppressed binary decision diagram* (ZBDD) [70]. To facilitate further explanation in these chapters, the fundamentals of ZBDDs and related concepts are explained here. More extensive background material on this subject can be found in [26, 67, 69, 70, 88].

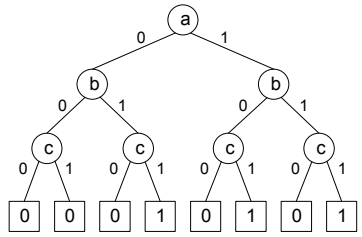
2.4.1 Representations of Boolean functions

A Boolean function is a function whose range is $\mathbb{B} = \{0, 1\}$. It can be understood to evaluate the truth or falsity of each element of its domain. Boolean functions play a role in questions of complexity theory as well as the design of digital circuits and cryptography.

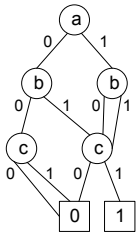
Boolean functions can be represented in several ways. For example, Figure 2.8(a) and 2.8(b) show the truth table and the *binary decision diagram* (BDD) of $f = (a + b)c$, respectively. Decision diagrams, which are arranged so that variables are in any given order, can be reduced and made into a canonical representation of the function [15]. Reduction rules are (1) merge equivalent sub-graphs and (2) remove vertices with identical sub-graphs. For example, we can apply the first rule to the

a	b	c	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

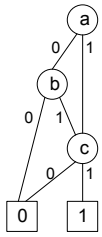
(a)



(b)



(c)



(d)

Figure 2.8: Representations of a Boolean logic function $f = (a + b)c$. (a) Truth table. (b) BDD for the variable order (a, b, c). (c) After applying the first reduction rule. (d) After applying the second reduction rule. This corresponds to the ROBDD for the variable order (a, b, c).

BDD in Figure 2.8(b) and obtain the BDD in Figure 2.8(c). Applying the second rule to the BDD in Figure 2.8(c) finally gives the *reduced ordered BDD* (ROBDD) representation in Figure 2.8(d).

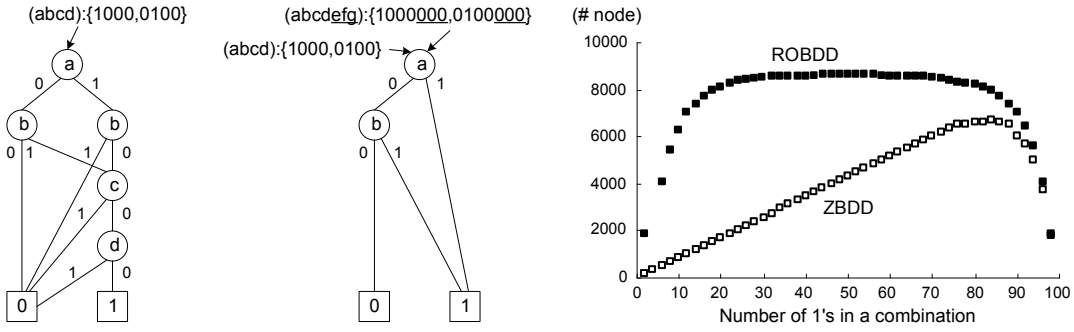
ROBDDs have found widespread use in the optimization and verification of VLSI design [16] since they provide a canonical representation of Boolean functions. Furthermore, when ROBDDs are used, the computational complexity of a problem depends on the size of its ROBDD representations, which often have mild growth with the problem size [15, 26].

2.4.2 Zero-suppressed BDDs

Zero-suppressed BDDs (ZBDDs) are a variant of ROBDDs that represent a set of combinations [69,70]. A *combination* of n elements is an n-bit vector $(x_1, x_2, \ldots, x_n) \in \mathbb{B}^n$, where $\mathbb{B} = \{0, 1\}$. The i-th bit reports whether the i-th element is contained in the combination. Thus, a set of combinations can be represented by a Boolean function $f : \mathbb{B}^n \to \mathbb{B}$. A combination given by the input vector $(x_1, x_2, \ldots, x_n)$ is contained in the set if and only if $f(x_1, x_2, \ldots, x_n) = 1$.

In most combinatorial applications, sets of combinations are *sparse*, which is defined as follows [67]:

- The sets contain only a small fraction of the 2^n possible bit vectors;



(a) (b) (c)

Figure 2.9: Representation of a set of combinations. (a) ROBDD representation. (b) ZBDD representation. (c) Comparison of ROBDD and ZBDD [70].

- Each bit vector in the sets has many zeroes.

The ZBDD is an efficient data structure used to represent and manipulate a set of combinations [69, 70]. Minato proposed two reduction rules to reduce ordinary BDDs to ZBDDs [69, 70]: (1) merge equivalent sub-graphs, and (2) if the 1-edge of a node v points to the 0-terminal vertex, then eliminate v and redirect all incoming edges of v to the 0-successor of v. Consequently, ZBDDs can exploit both of the aforementioned sparsity and provide an efficient representation for manipulating large-scale sets of combinations [67].

For instance, the ROBDD in Figure 2.9(a) represents a set of combinations {1000, 0100} for four input variables (*abcd*). Each path from the root vertex to the 1-leaf corresponds to a combination. By applying the ZBDD reduction rules, we can reduce the BDD in Figure 2.9(a) to the ZBDD in Figure 2.9(b), which is more compact in terms of the number of vertices. As shown in Figure 2.9(c), Minato [70] compared the size of a ZBDD with that of an ROBDD for a large set of combinations and showed that ZBDDs provide a much more compact representation of sets of combinations in most cases.

ZBDD representations are independent of the number of input variables as long as the combination remains the same, which is due to the "zero-suppression" effect. Consequently, we do not need to fix the number of input variables before generating graphs, and ZBDDs automatically suppress the variables that never appear in any

combination [70]. For example, a set of combinations $\{1000000, 0100000\}$ for seven variables ($abcdefg$) is represented by the same ZBDD in Figure 2.9(b). This property does not hold for other types of BDDs.

Chapter 3

A ZBDD-based Biclustering Algorithm

As already stated in the previous chapter, the cluster search problem is in general NP-hard [97], and the problem of biclustering is no exception [21, 98, 107]. This chapter describes a biclustering algorithm that exploits a compact data structure called *zero-suppressed binary decision diagrams* (ZBDDs) [67, 69, 70] to cope with this computational challenge.

This chapter focuses on the description of an algorithm to analyze gene expression data sets. This is for simplicity of explanation, and the proposed method is not limited to gene expression analysis. Chapter 4 will provide a more generalized description of the algorithm, which can analyze a variety of biological data sets. The experimental results obtained by applying the proposed method to gene expression data sets can be found in Chapter 5.

The organization of this chapter is as follows. Section 3.1 explains how to characterize a bicluster and provides a precise definition of biclusters the proposed algorithm aims to find. Section 3.2 introduces a special type of biclusters that act as a "seed" and explains an efficient algorithm to find them. Section 3.3 shows several properties every bicluster in our definition has. These properties are essential to understand the proposed biclustering method. Section 3.4 describes the ZBDD-based biclustering approach at length.

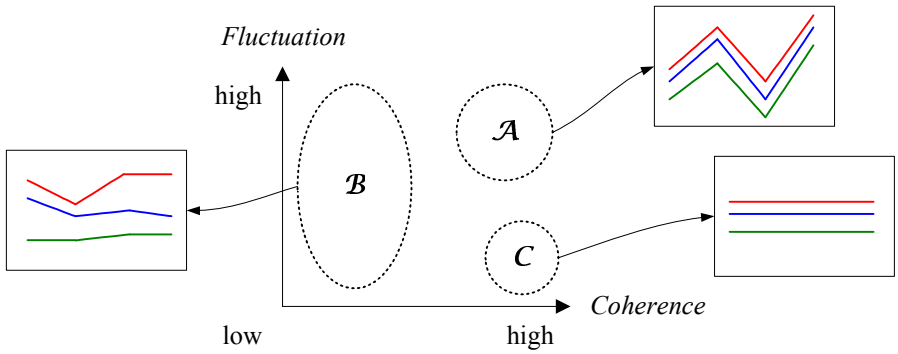


Figure 3.1: Characterization of biclusters. In some applications, such as gene co-regulation analysis, the biclusters in area $\mathcal{A}$ are most interesting. On the other hand, the biclusters in area $\mathcal{C}$ are important in other applications, such as marker gene identification.

3.1 Preliminaries

3.1.1 Characterization of biclusters

We can characterize a bicluster by several criteria. The most common method is to measure the degree of *coherence*, or similarity in behavior, among the objects in a bicluster. One way to measure coherence is to use the *mean squared residue* (MSR) score described in Section 2.3.3. As shown in the examples in Figure 5.2 of Chapter 5, a low value of MSR typically means a high level of coherence, and vice versa [21]. In addition to coherence, it may be helpful to characterize biclusters by the degree of *fluctuation* in gene expression levels, which cannot be captured by coherence measurement. For instance, the lowest MSR value (zero) indicates that the gene expression levels fluctuate in unison (*e.g.*, Figure 5.2(a)). However, flat biclusters with no fluctuation can also have an MSR value of zero (*e.g.*, Figure 5.2(b)). Taken together, Figure 3.1 shows a plot in which biclusters can be placed according to their degree of coherence and fluctuation.

In some applications, such as gene co-regulation analysis, the biclusters in area $\mathcal{A}$ would be the most interesting because similar behavior between highly-expressed genes is much more important than that between two poorly-expressed genes [35]. On the other hand, the "flat" biclusters in area $\mathcal{C}$ are important and need to be

considered in other applications, such as the identification of marker genes. Suppose that we are interested in correlating the activity of one or more genes to specific sub-phenotypes. If specific genes are expressed in some phenotypes and not in others, and if we eliminate the genes whose expression levels do not change much over the range of experimental conditions, then the emerging biclusters will be flat. Qualitatively speaking, the biclusters in area $\mathcal{B}$ are less interesting because they have a lower level of coherence than those in areas $\mathcal{A}$ or $\mathcal{C}$.

Many techniques have been proposed to find biclusters with a high level of coherence, particularly those that can be placed in areas $\mathcal{A}$ or $\mathcal{C}$ on the characterization plot. Some methods define a bicluster in a way such that any sub-bicluster of the bicluster is yet another bicluster under the same definition and input parameters. Examples include the δ-valid kj-patterns [18], OPSMs [10], xMOTIFs [74], δ-pClusters [107], and GEMS [112]. By measuring coherence with fine granularity, these biclusters can potentially exhibit high degrees of coherence [107]. Our approach also takes advantage of this property to find coherent biclusters.

When removing flat biclusters is beneficial, we can employ the following *average row variance* (ARV) to eliminate them:

$$ARV(I, J) = \frac{1}{|I||J|} \sum_{i \in I} \sum_{j \in J} (a_{ij} - a_{i\bullet})^2. \tag{3.1}$$

3.1.2 Definitions

Let $U_G = \{g_0, g_1, \ldots, g_{n-1}\}$ and $U_E = \{e_0, e_1, \ldots, e_{m-1}\}$ represent a set of genes and a set of experimental conditions involved in gene expression measurement, respectively. The result can be represented by the matrix $D \in \mathbb{R}^{|U_G| \times |U_E|}$, with the set of rows U_G and set of columns U_E. Each element d_{ij} in D corresponds to the expression information of gene g_i in experiment e_j. We can denote D by the pair (U_G, U_E). Depending on the microarray technology used, the information reflects either absolute expression levels (*e.g.,* Affymetrix GeneChips) or relative expression ratios (*e.g.,* cDNA microarrays) [51]. Our method is applicable to both.

A bicluster is defined to be a subset of genes that exhibit similar behavior under a

	e_0	e_1	e_2	e_3	e_4	e_5
g_0	2.0	2.0	9.0	2.0	3.0	4.0
g_1	3.0	7.0	3.0	1.0	9.0	3.0
g_2	2.0	2.0	7.0	2.0	6.0	3.0
g_3	3.0	2.0	3.0	2.0	1.0	3.0
g_4	2.0	1.0	5.0	1.0	0.0	4.0
g_5	3.0	5.0	5.0	8.0	2.0	3.0

(a)

#	G	E
0	$\{g_0, g_2, g_3, g_4\}$	$\{e_0, e_1, e_3\}$
1	$\{g_0, g_2, g_3\}$	$\{e_1, e_3, e_5\}$

(b)

Figure 3.2: A running example. (a) Gene expression data matrix $D = (U_G, U_E)$, where $U_G = \{g_0, g_1, g_2, g_3, g_4, g_5\}$ and $U_E = \{e_0, e_1, e_2, e_3, e_4, e_5\}$. (b) Two maximal biclusters on D we are going to find. The parameters (see Section 3.1.3) used are $\delta = 1$ and $M_G = M_E = 3$.

subset of experimental conditions, and vice versa. Thus, in the gene expression data matrix D, a bicluster will appear as a submatrix of D. We denote this submatrix by pair $B = (G, E)$, where $G \subseteq U_G$ and $E \subseteq U_E$. We specify the size of bicluster B by $|G| \times |E|$.

Example 3.1. An example of data matrix $D = (U_G, U_E)$ and biclusters on D are shown in Figure 3.2(a) and 3.2(b), respectively. Throughout this chapter, it will be explained how to find these two biclusters from the matrix D.

3.1.3 Formal definition of a bicluster and problem statement

Definition 3.1. *For the gene expression matrix $D \in \mathbb{R}^{|U_G| \times |U_E|}$, let the pair $B = (G, E)$ represent a submatrix of D. That is, $G \subseteq U_G$ and $E \subseteq U_E$. The matrix B is called a* bicluster *if the value of $|x - z - y + w|$ is less than or equal to some δ for any 2×2 submatrix $\begin{bmatrix} x & y \\ z & w \end{bmatrix}$ in B.*

Definition 3.2. *Given the gene expression data matrix D, the objective is to find every submatrix $B = (G, E)$ of D that is: (1) a bicluster with respect to a given δ; (2) not too small, namely $|G| \geq M_G$ and $|E| \geq M_E$ for given values of M_G and M_E; and (3) maximal, or not contained by other biclusters that satisfy the previous conditions.*

Example 3.2. The two maximal biclusters in Figure 3.2(b) are found in the data matrix in Figure 3.2(a) by our algorithm with the parameters $\delta = 1$, $M_G = M_E = 3$.

In essence, our definition of a bicluster is equivalent to that of the δ-pCluster [107]. The rationale of choosing this particular bicluster model is that δ-pClusters can have multiple desirable properties. According to Wang et al. [107], δ-pClusters are more resilient to outliers and more coherent than alternatives. Another very important property is that a sub-bicluster of a δ-pCluster is yet another δ-pCluster, which often results in a high level of coherence.

However, our algorithm is significantly different from the pClustering technique. The experimental results in Chapter 5 will show that a substantial speed-up is possible by our method even with a reasonably optimized method such as the pClustering algorithm. Furthermore, our algorithm can find biclusters that can be placed in area A as well as area C (see Figure 3.1), whereas the biclusters found by pClustering tend to be located mainly in area C [119]. The next paragraph explains this reasoning.

The threshold parameter δ affects many aspects of the biclustering problem, as shown in Figure 3.3. First, the difficulty of a biclustering problem depends to some extent on the value of δ, since the amount of intermediate data is proportional to the size of this threshold value. Second, as the value of δ grows, the capability of the algorithm to find coherent patterns decreases. In contrast, the wide dynamic range exhibited by fluctuating patterns can better be captured by a larger value of δ. Figure 3.3(a) and 3.3(b) depict these observations. Consequently, as shown in Figure 3.3(c), a large value of δ typically results in biclusters in area A, whereas a small value usually produces biclusters in area C. According to our experiments, our algorithm can handle larger values of δ than the pClustering algorithm. Consequently, our algorithm can find biclusters in area A as well as those in area C.

3.2 Pairwise maximal biclusters (PMBs)

In this section, we introduce a special kind of bicluster called *pairwise maximal biclusters*, which play a crucial role in our method. Although the biclustering problem is, in

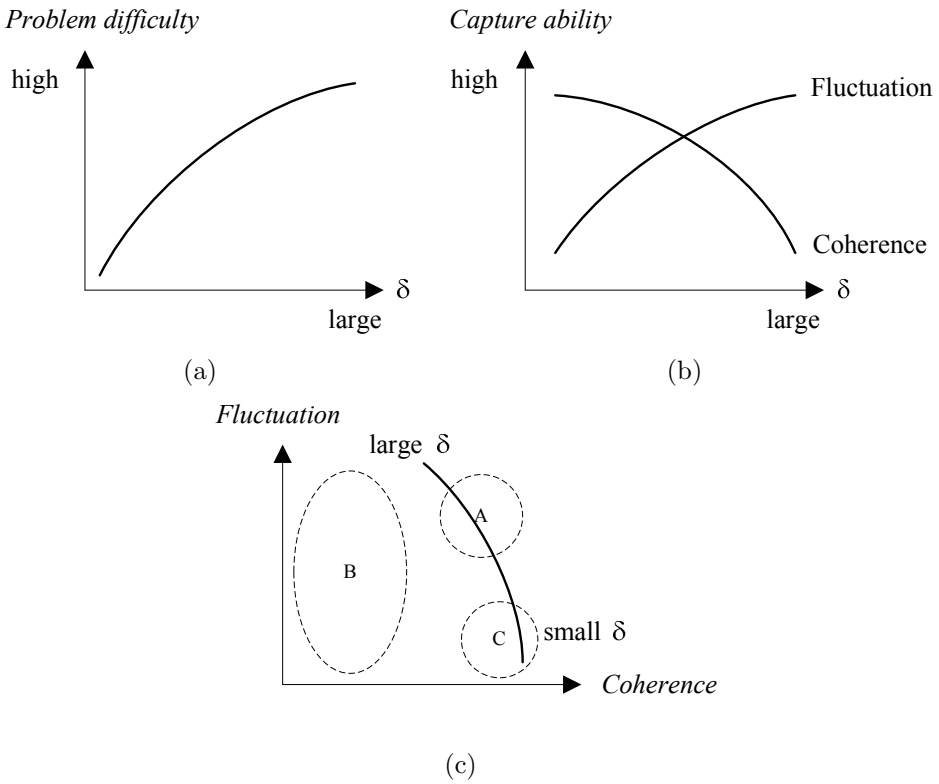


Figure 3.3: Qualitative analysis of dependency on δ. (a) A larger value of δ means a more difficult problem. (b) The ability to capture fluctuation is roughly proportional to the value of δ, whereas the ability to capture coherence decreases as the parameter δ becomes larger. (c) A small value of δ tends to find biclusters in area C, while a large value of δ typically finds biclusters in areas A or B.

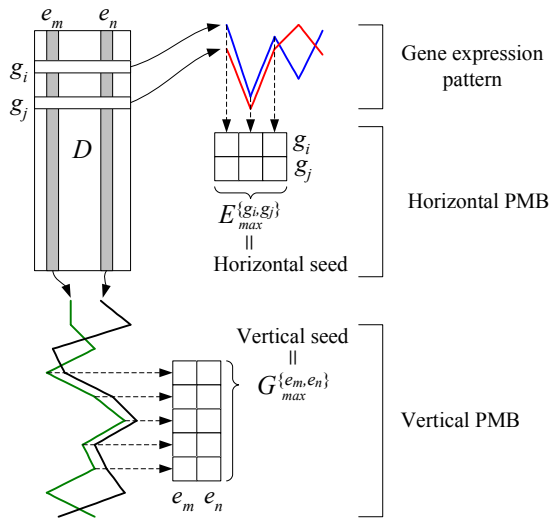


Figure 3.4: Pairwise maximal biclusters (PMBs).

Table 3.1: Notations for PMB and seed.

Notation	Meaning
$E^{\{g_i,g_j\}}_{max}$	Horizontal seed for genes $\{g_i, g_j\}$
$\{E^{\{g_i,g_j\}}_{max}\}$	Set of all horizontal seeds for $\{g_i, g_j\}$
$(\{g_i, g_j\}, E^{\{g_i,g_j\}}_{max})$	Horizontal PMB
$G^{\{e_m,e_n\}}_{max}$	Vertical seed for experiments $\{e_m, e_n\}$
$\{G^{\{e_m,e_n\}}_{max}\}$	Set of all vertical seeds for $\{e_m, e_n\}$
$(G^{\{e_m,e_n\}}_{max}, \{e_m, e_n\})$	Vertical PMB

general, intractable [21, 107], these special biclusters can be discovered in polynomial time.

3.2.1 Definition of PMBs

We refer to $2 \times |E|$ or $|G| \times 2$ maximal biclusters as *pairwise maximal biclusters* (PMBs). As shown in Figure 3.4, a *horizontal PMB* is a bicluster that consists of two genes and a maximal (but not necessarily unique) set of experiments in which the two genes show a similar behavior. We refer to this maximal set as a *horizontal seed*

for the two genes. More formally, horizontal PMBs and seeds are defined as follows.

Definition 3.3 (Horizontal PMB and seed)**.** *Assume* $B = (\{g_i, g_j\}, E)$ *is a* $2 \times |E|$ *bicluster. If there does not exist* $E' \supset E$ *such that* $(\{g_i, g_j\}, E')$ *is also a* $2 \times |E'|$ *bicluster, then the set of experiments* E *is called a* horizontal seed *and is denoted by* $E_{max}^{\{g_i,g_j\}}$*. In this case, we call* B *a* horizontal *PMB for two genes* $\{g_i, g_j\}$*, and denote it by* $B = \left(\{g_i, g_j\}, E_{max}^{\{g_i,g_j\}}\right)$.

As will be shown shortly, multiple instances of $E_{max}^{\{g_i,g_j\}}$ can exist for a given pair $\{g_i, g_j\}$. We denote the set of all $E_{max}^{\{g_i,g_j\}}$ as $\left\{E_{max}^{\{g_i,g_j\}}\right\}$.

By switching the roles of genes and experiments, *vertical PMBs* and *vertical seeds* are similarly defined. Table 3.1 summarizes the notations defined in this section.

3.2.2 Generation of PMBs

Wang et al. [107] proposed a biclustering method called pClustering. The first step of their algorithm is to find all the maximal $n \times 2$ biclusters that satisfy specific input conditions in polynomial time. In order to generate PMBs, we use a similar approach. Our biclustering method then differs completely in the remaining steps.

Algorithm 3.1 describes how to generate vertical seeds for a given pair of experiments. An algorithm to generate horizontal seeds is similar but is not shown here. The worst-case time complexity of Algorithm 3.1 is $O(nlogn)$ [107], where n is the number of genes in the input gene expression matrix. Moreover, the maximum number of vertical seeds that can be generated by the algorithm for each pair of experiments is $(n - 1)$. Consequently, Algorithm 3.1 is efficient, and the number of seeds discovered by this algorithm does not grow exponentially.

Example 3.3. Tables 3.2(b) and 3.2(c) show the vertical seeds and the horizontal seeds generated from the data set in Figure 3.2(a), which is repeated in Table 3.2(a) for convenience.

Algorithm 3.1: Generating vertical seeds

input : X_i, gene expression values in experiment e_i
input : X_j, gene expression values in experiment e_j
input : δ, cluster threshold
input : M_G, minimum number of genes in a seed
output: $G_{max}^{\{e_i,e_j\}}$, vertical seeds for e_i and e_j

```
1  for k = 0 to |X| − 1 do
2      s[k].v := X_i[k] − X_j[k];
3      s[k].i := k;
4  Sort array s in ascending order with respect to the field v;
5  start := 0; end := 1;
6  new := TRUE;
7  repeat
8      v := s[end].v − s[start].v;
9      if (|v| ≤ δ) then
10         if (end − start ≥ M_G AND new = TRUE) then
11             Report
               {s[start].i, s[start + 1].i, ..., s[end − 1].i};
12         end := end + 1;
13         new := TRUE;
14     else
15         start := start + 1;
16         new := FALSE;
17 until (end ≥ |X|);
18 if (end − start ≥ M_G AND new = TRUE) then
19     Report {s[start].i, s[start + 1].i, ..., s[end − 1].i};
```

Figure 3.5: Algorithm to find vertical seeds.

Table 3.2: Example. ($\delta = 1, M_G = M_E = 3$)

(a) Data matrix

	e_0	e_1	e_2	e_3	e_4	e_5
g_0	2.0	2.0	9.0	2.0	3.0	4.0
g_1	3.0	7.0	3.0	1.0	9.0	3.0
g_2	2.0	2.0	7.0	2.0	6.0	3.0
g_3	3.0	2.0	3.0	2.0	1.0	3.0
g_4	2.0	1.0	5.0	1.0	0.0	4.0
g_5	3.0	5.0	5.0	8.0	2.0	3.0

(b) Horizontal PMBs

Genes	Horizontal seed: $E_{max}^{\{g_i,g_j\}}$
$\{g_0, g_2\}$	$\{e_0, e_1, e_3, e_5\}$
$\{g_0, g_3\}$	$\{e_0, e_1, e_3\}$,$\{e_1, e_3, e_5\}$
$\{g_0, g_4\}$	$\{e_0, e_1, e_3, e_5\}$
$\{g_1, g_3\}$	$\{e_0, e_2, e_3, e_5\}$
$\{g_2, g_3\}$	$\{e_0, e_1, e_3, e_5\}$
$\{g_2, g_4\}$	$\{e_0, e_1, e_3\}$,$\{e_1, e_2, e_3\}$
$\{g_3, g_4\}$	$\{e_0, e_1, e_3, e_4\}$
$\{g_3, g_5\}$	$\{e_0, e_4, e_5\}$

(c) Vertical PMBs

Experiments	Vertical seed: $G_{max}^{\{e_m,e_n\}}$
$\{e_0, e_1\}$	$\{g_0, g_2, g_3, g_4\}$
$\{e_0, e_3\}$	$\{g_0, g_2, g_3, g_4\}$,$\{g_1, g_3, g_4\}$
$\{e_0, e_4\}$	$\{g_3, g_4, g_5\}$
$\{e_0, e_5\}$	$\{g_0, g_2, g_4\}$,$\{g_1, g_2, g_3, g_5\}$
$\{e_1, e_3\}$	$\{g_0, g_2, g_3, g_4\}$
$\{e_1, e_5\}$	$\{g_0, g_2, g_3\}$
$\{e_2, e_5\}$	$\{g_1, g_3, g_4\}$
$\{e_3, e_5\}$	$\{g_0, g_1, g_4\}$,$\{g_0, g_1, g_2, g_3\}$
$\{e_4, e_5\}$	$\{g_0, g_3, g_5\}$

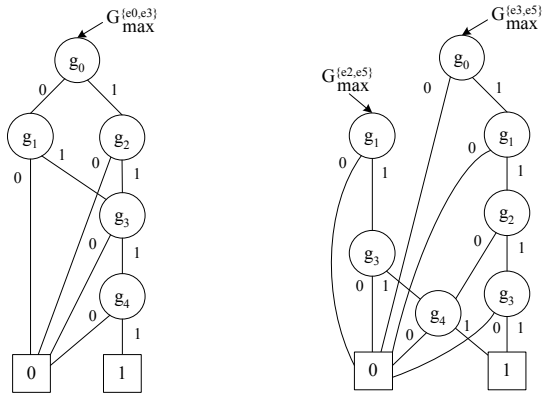


Figure 3.6: ZBDDs for vertical seeds.

3.2.3 Representation of vertical seeds

Details on the representation of horizontal seeds will be provided in Section 3.4.1. Here we explain how to represent vertical seeds. In particular, we utilize *zero-suppressed binary decision diagrams* (ZBDDs) [69], an efficient data structure for large-scale sets. This ZBDD-based representation is crucial to keeping the entire algorithm computationally manageable.

The key observation is that a set of vertical seeds can be regarded as a set of combinations and thus represented compactly by the ZBDDs. The set of vertical seeds $\left\{G_{max}^{\{e_m,e_n\}}\right\}$ normally has much fewer elements than $2^{|U_G|}$. In addition, $|G_{max}^{\{e_m,e_n\}}| \ll |U_G|$ for typical $G_{max}^{\{e_m,e_n\}}$. In other words, both types of sparsity introduced in Section 2.4.2 hold true. Hence, the symbolic representation using ZBDDs is more compact than the traditional data structures for sets. Furthermore, as shown in Section 3.4.2, the manipulation of vertical seeds, such as union and intersection, is implicitly performed on ZBDDs, thus resulting in high efficiency. Refer to Section 2.4.2 for details on how to construct ZBDDs.

Example 3.4. In Table 3.2(c) we showed the vertical seeds for our running example. The ZBDD in Figure 3.6(a) represents the set of vertical seeds $\left\{G_{max}^{\{e_0,e_3\}}\right\} = \{\{g_0, g_2, g_3, g_4\}, \{g_1, g_3, g_4\}\}$.

Example 3.5. The ZBDD representation of $\left\{G_{max}^{\{e_3,e_5\}}\right\} = \{\{g_0, g_1, g_2, g_3\}, \{g_0, g_1, g_4\}\}$

is shown in Figure 3.6(b), along with that of $\left\{G_{max}^{\{e_2,e_5\}}\right\} = \{\{g_1, g_3, g_4\}\}$.

3.3 Properties of biclusters

Recall that a bicluster is composed of gene set G and experiment set E. We first show how G and E are related to vertical and horizontal seeds, respectively. Then, we present the property that reveals how G and E are mathematically related to each other.

3.3.1 Relationship between G, E, and seeds

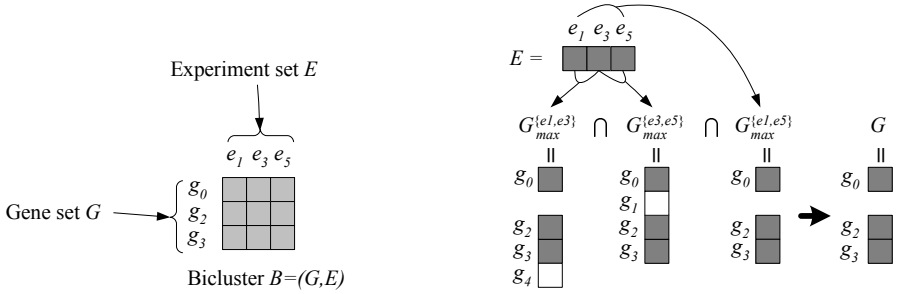
For the gene set G in bicluster (G, E), G is a subset of a certain vertical seed. More formally, the following proposition holds.

Proposition 3.1. *Let (G, E) be a bicluster. If $E \supseteq \{e_m, e_n\}$, then there exists $G_s \in \left\{G_{max}^{\{e_m,e_n\}}\right\}$ such that $G \subseteq G_s$.*

Proof. Assume $G \supset G_s$ for all $G_s \in \left\{G_{max}^{\{e_m,e_n\}}\right\}$. Since (G, E) is a bicluster and $E \supseteq \{e_m, e_n\}$, its sub-bicluster $(G, \{e_m, e_n\})$ is also a bicluster for a given value of δ. By definition, if $G_s \in \left\{G_{max}^{\{e_m,e_n\}}\right\}$, then there exists no $G' \supset G_s$ such that $(G', \{e_m, e_n\})$ is yet another bicluster for the given value of δ. We have reached a contradiction and thus our original assumption that $G \supset G_s$ for all $G_s \in \left\{G_{max}^{\{e_m,e_n\}}\right\}$ must be false. Therefore, there must be at least one instance of $G_s \in \left\{G_{max}^{\{e_m,e_n\}}\right\}$ such that $G \subseteq G_s$. □

Example 3.6. Consider bicluster #1 in Figure 3.2(b), in which $G = \{g_0, g_2, g_3\}$ and $E = \{e_1, e_3, e_5\}$. From Table 3.2(c), $\left\{G_{max}^{\{e_3,e_5\}}\right\} = \{\{g_0, g_1, g_4\}, \{g_0, g_1, g_2, g_3\}\}$. There exists $G_s \in \left\{G_{max}^{\{e_3,e_5\}}\right\}$ such that $G \subseteq G_s$. That is, $G \subseteq G_s = \{g_0, g_1, g_2, g_3\}$.

Similarly, for any experiment set E in bicluster (G, E), the set E is a subset of a certain horizontal seed, as formally stated in the following proposition. (Its proof is similar to the proof of Proposition 3.1.)



(a) Definitions (b) Deriving G from E

Figure 3.7: Relationship between G and E in a bicluster $B = (G, E)$.

Proposition 3.2. *Let (G, E) be a bicluster. If $G \supseteq \{g_i, g_j\}$, then there exists $E_s \in \left\{E_{max}^{\{g_i,g_j\}}\right\}$ such that $E \subseteq E_s$.*

Example 3.7. Consider bicluster #0 in Figure 3.2(b), in which $G = \{g_0, g_2, g_3, g_4\}$ and $E = \{e_0, e_1, e_3\}$. From Table 3.2(b), $\left\{E_{max}^{\{g_0,g_3\}}\right\} = \{\{e_0, e_1, e_3\}, \{e_1, e_3, e_5\}\}$. There exists $E_s \in \left\{E_{max}^{\{g_0,g_3\}}\right\}$ such that $E \subseteq E_s$. That is, $E \subseteq E_s = \{e_0, e_1, e_3\}$.

3.3.2 Relationship between G and E

We first define $\otimes$, a pairwise intersection operator on two sets of subsets $\mathcal{A}$ and $\mathcal{B}$:

$$\mathcal{A} \otimes \mathcal{B} = \{I | I = A \cap B,\ \forall A \in \mathcal{A} \text{ and } \forall B \in \mathcal{B}\}. \tag{3.2}$$

For instance, $\{\{0, 1, 2\}, \{2, 3, 4\}\} \otimes \{\{0, 2\}, \{4, 5\}\} = \{\{0, 2\}, \{2\}, \{4\}\}$.

Now we use an example to reveal the relationship between G and E by Proposition 4.1. We use bicluster #1 in Figure 3.2(b), which is depicted in Figure 3.7(a). By Proposition 4.1, there exists $G_1 \in \left\{G_{max}^{\{e_1,e_3\}}\right\}$ such that $G \subseteq G_1$. Similarly, there exists $G_2 \in \left\{G_{max}^{\{e_3,e_5\}}\right\}$ such that $G \subseteq G_2$. Also, there exists $G_3 \in \left\{G_{max}^{\{e_1,e_5\}}\right\}$ such that $G \subseteq G_3$. Thus, $G \subseteq G_1 \cap G_2 \cap G_3$. If we assume that bicluster B is maximal, then there is no G' such that $G' \supset G$ and $G' \supset G_1 \cap G_2 \cap G_3$. Therefore, as shown in Figure 3.7(b), $G = G_1 \cap G_2 \cap G_3$, assuming that we know the sets G_1, G_2, G_3. In practice, there can be multiple G_1, G_2, G_3, and we need to use the operator $\otimes$ instead

of the operator $\cap$. That is,

$$\begin{aligned} \mathcal{G} &= \{\text{all } G \text{ derivable from } E\} \\ &= \left\{G_{max}^{\{e_1,e_3\}}\right\} \otimes \left\{G_{max}^{\{e_3,e_5\}}\right\} \otimes \left\{G_{max}^{\{e_1,e_5\}}\right\}. \end{aligned}$$

In general, the following equation holds:

$$\begin{aligned} \mathcal{G} &= \{\text{all } G \text{ derivable from } E\} && (3.3) \\ &= \bigotimes_{\forall(e_m,e_n)\in E} \left\{G_{max}^{\{e_m,e_n\}}\right\}, && (3.4) \end{aligned}$$

and by symmetry,

$$\begin{aligned} \mathcal{E} &= \{\text{all } E \text{ derivable from } G\} && (3.5) \\ &= \bigotimes_{\forall(g_i,g_j)\in G} \left\{E_{max}^{\{g_i,g_j\}}\right\}. && (3.6) \end{aligned}$$

In this work, we use Eqn. 3.4 but not Eqn. 3.6 because, in most gene expression data, $|U_E| \ll |U_G|$, which makes the evaluation of Eqn. 3.4 much faster.

3.4 Our biclustering algorithm

We first repeat the problem statement presented in Section 3.1.3. For the given gene expression matrix $D \in \mathbb{R}^{|U_G|\times|U_E|}$, the objective is to find every matrix $B = (G, E)$ such that (i) $G \subseteq U_G$ and $E \subseteq U_E$ and (ii) the value of $|x - z - y + w| \leq \delta$ for any 2×2 submatrix $\begin{bmatrix} x & y \\ z & w \end{bmatrix}$ in B for a given $\delta \geq 0$. In particular, we are interested in finding B such that (i) B is not too small, namely $|G| \geq M_G$ and $|E| \geq M_E$ and (ii) B is maximal in the sense that it is not contained by others that satisfy the previous conditions.

Our approach is to generate the horizontal and vertical PMBs from a data matrix and then to derive other biclusters from them, as shown informally in Figure 3.8. Sections 3.3.1 and 3.3.2 together suggest a way to derive biclusters from PMBs: we first determine E from the horizontal seeds and then compute G by Eqn. 3.4 with

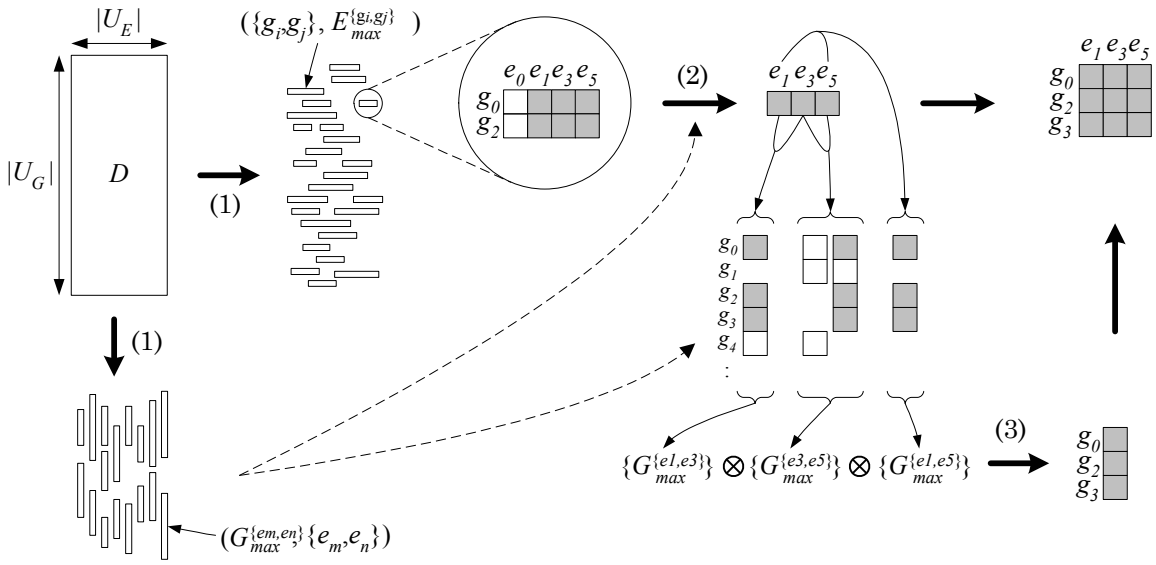


Figure 3.8: Overview. (1) Generating horizontal and vertical PMBs: Section 3.2.2. (2) Predicting the experiment set E: Section 3.4.1. (3) Calculating the gene set G from E: Section 3.4.2.

reference to the vertical seeds. This idea is elaborated upon in this section.

Figure 3.9 shows a flowchart of the proposed method. As described in Section 3.2.2, Algorithm 3.1 is used to find vertical and horizontal PMBs. Section 3.2.3 already presented how to represent vertical seeds by ZBDDs. The representation for horizontal seeds will be explained in Section 3.4.1. Algorithm 3.2, presented in Section 3.4.1, is used to predict E from horizontal seeds. Section 3.4.2 describes Algorithm 3.3, which can derive G from E by Eqn. 3.4. For very large-scale gene expression data, we can optionally split input data by the method introduced in Section 3.4.3 before starting Algorithms 3.2 and 3.3 in order to reduce computational burden.

3.4.1 Predicting the experiment set E

For bicluster $B = (G, E)$, assume that $G \supseteq \{g_i, g_j\}$. Then, $E \subseteq E_{max}^{\{g_i,g_j\}}$ by Proposition 3.2 in Section 3.3.1. In the current setup, what we have is $E_{max}^{\{g_i,g_j\}}$ and what we are finding is E. Examining every subset of $E_{max}^{\{g_i,g_j\}}$ could eventually allow us to find E. However, it is time-consuming to probe every subset. Thus, we present a

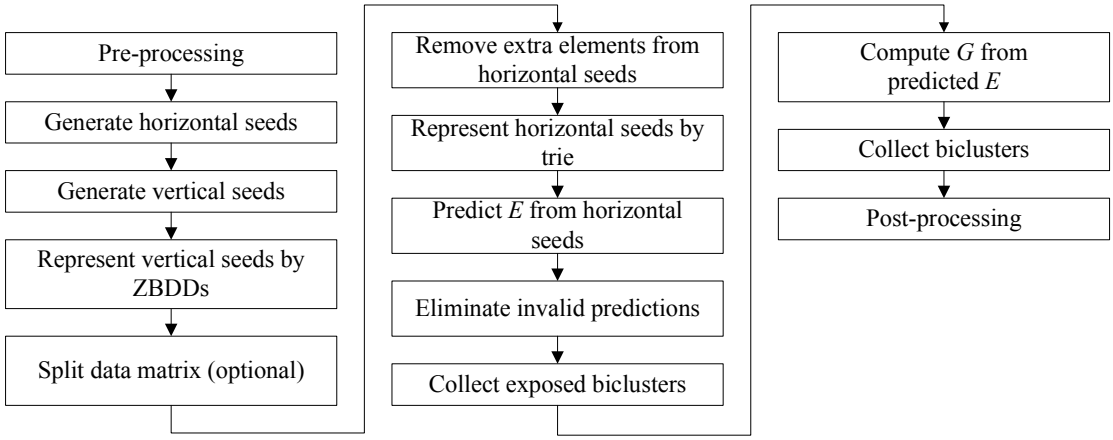


Figure 3.9: Overall flow of the proposed method.

technique to avoid exhaustive enumeration of subsets in Algorithm 3.2. This algorithm considers multiple instances of E simultaneously. Next we show each step in detail and with examples using the data matrix and the seeds in Table 3.2 with the parameters $\delta = 1$ and $M_G = M_E = 3$.

Step 1: Removing the extra elements in a horizontal seed

We only consider the subsets of horizontal seeds that are *valid*, according to the following definition, in order to find E.

Definition 3.4. *Let V be a subset of the horizontal seed $E_{max}^{\{g_i,g_j\}}$. The set V is called* valid *if the following conditions are both met: (1) $|V| \geq M_E$ and (2) for all $\{e_m, e_n\}$ in V, there exists at least one*[1] *set $G_{max}^{\{e_m,e_n\}}$ such that $G_{max}^{\{e_m,e_n\}} \supseteq \{g_i, g_j\}$.*

The invalid subsets need not be examined, because they either contain too few elements (Condition 1) or they cannot produce any gene set by applying Eqn. 3.4 to them (Condition 2).

Example 3.8. Let E_1 and E_2 be the two instances of the seed $E_{max}^{\{g_2,g_4\}}$ in Table 3.2(b). In particular, assume that $E_1 = \{e_0, e_1, e_3\}$ and $E_2 = \{e_1, e_2, e_3\}$. E_1 is valid

[1] Although it seems that any $G_{max}^{\{e_m,e_n\}}$ always has to contain at least g_i and g_j, the set $G_{max}^{\{e_m,e_n\}}$ may not exist for some $\{e_m, e_n\}$. This is because Algorithm 3.1 does not generate $G_{max}^{\{e_m,e_n\}}$ at all if it has less than M_G elements.

Algorithm 3.2: Predicting experiment set E

input : Horizontal seeds; minimum bicluster size (M_G, M_E)
output: Candidates for experiment set E

/* *Step 1: removing extra elements* */
foreach $E_{max}^{\{g_i,g_j\}}$ **do**
    $\mathbf{V} = E_{max}^{\{g_i,g_j\}}$;
    $\mathbf{E} = \{\}$;
    foreach (e_m, e_n) *in* $\mathbf{V}$ **do**
        if $\exists G_{max}^{\{e_m,e_n\}} \supseteq \{g_i, g_j\}$ **then** $\mathbf{E} = \mathbf{E} \cup \{(e_m, e_n)\}$;
    Construct an undirected graph $(\mathbf{V}, \mathbf{E})$;
    $E_{max}^{\{g_i,g_j\}} = \mathbf{V} - \{$the vertices with the degree less than $M_E - 1\}$;
    if $E_{max}^{\{g_i,g_j\}}$ *has less than* M_E *experiments* **then**
        Remove $E_{max}^{\{g_i,g_j\}}$;

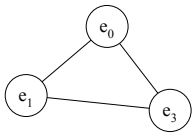
/* *Step 2: representing horizontal seeds by trie* */
foreach $E_{max}^{\{g_i,g_j\}}$ **do**
    Sort the elements in $E_{max}^{\{g_i,g_j\}}$;
    Locate node n whose path is specified by the ordered elements;
    $n.G = n.G \cup \{g_i, g_j\}$;
    $n.E = E_{max}^{\{g_i,g_j\}}$;

/* *Step 3: predicting experiment set* E *from horizontal seeds* */
foreach *node* n *in the post-order traversal of the trie* **do**
    Set $m.G = m.G \cup n.G$ for every node m in which
    $|m.E| = |n.E| - 1$ and $|m.E| \geq M_E$;

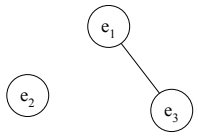
/* *Step 4: eliminating invalid predictions* */
foreach *node* n *in the pre-order traversal of the trie* **do**
    if $|n.G| < M_G$ **then** Remove n and its subtrie rooted at n;

/* *Step 5: collecting exposed biclusters* */
foreach *node* n *in the pre-order traversal of the trie* **do**
    if $(n.G, n.E)$ *is a bicluster* **then** Collect $(n.G, n.E)$;

Figure 3.10: Algorithm to predict experiment set E.



(a) $E_{max}^{\{g_2,g_4\}} = \{e_0, e_1, e_3\}$



(b) $E_{max}^{\{g_2,g_4\}} = \{e_1, e_2, e_3\}$

Genes	Horizontal seed: $E_{max}^{\{g_i,g_j\}}$
$\{g_0, g_2\}$	$\{e_0, e_1, e_3, e_5\}$
$\{g_0, g_3\}$	$\{e_0, e_1, e_3\}$,$\{e_1, e_3, e_5\}$
$\{g_0, g_4\}$	$\{e_0, e_1, e_3, e_5\}$
$\{g_1, g_3\}$	$\{e_0, e_3, e_5\}$
$\{g_2, g_3\}$	$\{e_0, e_1, e_3, e_5\}$
$\{g_2, g_4\}$	$\{e_0, e_1, e_3\}$
$\{g_3, g_4\}$	$\{e_0, e_1, e_3\}$
$\{g_3, g_5\}$	$\{e_0, e_4, e_5\}$

(c) Horizontal seeds after Step 1

Figure 3.11: Example for Step 1.

because there is at least one instance of each $G_{max}^{\{e_0,e_1\}}$, $G_{max}^{\{e_1,e_3\}}$, and $G_{max}^{\{e_0,e_3\}}$ containing $\{g_2, g_4\}$. In contrast, E_2 is not valid because $G_{max}^{\{e_1,e_2\}}$ and $G_{max}^{\{e_2,e_3\}}$ do not exist.

We can identify valid subsets using the notion of cliques on an undirected graph. Suppose we construct an undirected graph in which the vertices are the elements in $E_{max}^{\{g_i,g_j\}}$ and the edges exist according to the following: the edge between two vertices e_m and e_n exists if there is at least one set $G_{max}^{\{e_m,e_n\}}$ such that $G_{max}^{\{e_m,e_n\}} \supseteq \{g_i, g_j\}$, as described in **Lines 3–7** of Algorithm 3.2. Then a valid subset of $E_{max}^{\{g_i,g_j\}}$ corresponds to a clique (or complete subgraph) of at least M_E vertices.

Example 3.9. Figure 3.11(a) and 3.11(b) present the graphs for the sets E_1 and E_2 in the previous example, respectively. Only the former represents a valid subset.

However, the clique finding problem cannot in general be solved in polynomial time [24]. Thus, we instead remove the elements that *cannot* belong to a clique on the graph as an efficient heuristic. These elements are represented by vertices with a degree (the number of incident edges) less than $M_E - 1$. **Line 8** of the algorithm

is to remove these elements. If removing them from the set $E_{max}^{\{g_i,g_j\}}$ makes it contain fewer than M_E elements, then we eliminate the whole $E_{max}^{\{g_i,g_j\}}$ (**Lines 9-10**).

Example 3.10. In Figure 3.11(b), all vertices should be removed because none of them can belong to a clique of at least $M_E = 3$ vertices. Removing them makes the set E_2 empty and we therefore eliminate E_2 from further consideration.

Removing unnecessary elements from the seed $E_{max}^{\{g_i,g_j\}}$ is beneficial because the resulting set will have fewer elements, thus allowing us to examine a smaller number of subsets. Note that this heuristic aims at reducing the amount of data to be processed while preserving the quality of the biclustering results.

Example 3.11. Figure 3.11(c) lists our running example of the horizontal seeds from which unnecessary elements have been removed. In this simple example, only a few elements have been removed. However, in practical data, there exist many extra elements, and this step is helpful for improving the response time.

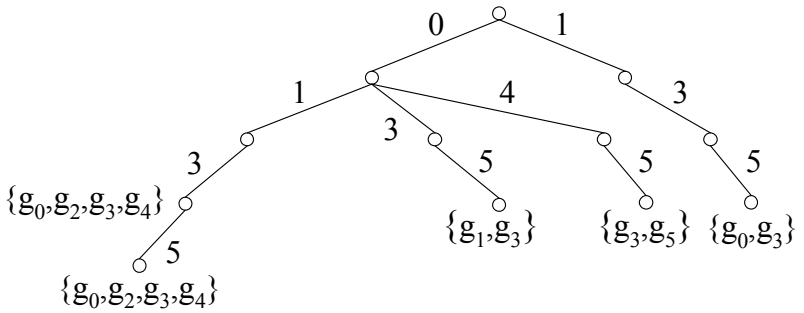
Step 2: Representation of horizontal seeds by a trie

In **Lines 12–16** of Algorithm 3.2, the horizontal seeds are collectively represented by a *trie*, a data structure used to represent sets of character strings [2]. Many overlaps occur between horizontal seeds, and the trie provides compact representations.

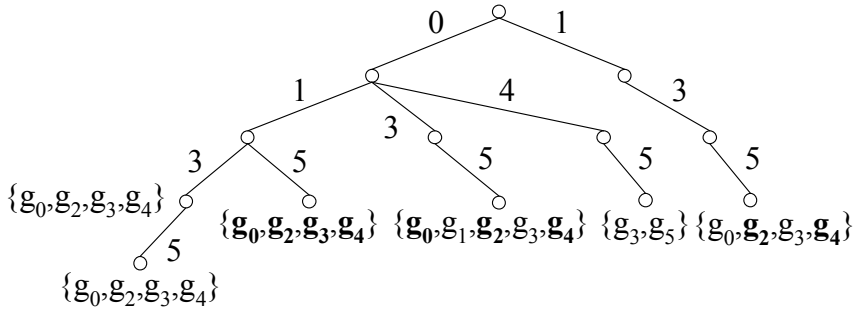
In a trie, each path from the root to a leaf corresponds to one word or character string in the represented set. This way, the nodes of the trie correspond to the prefixes of words in the set. For each seed $E_{max}^{\{g_i,g_j\}}$ found in the previous step, we first sort its elements assuming a total order among the elements, such as $e_0 \prec e_1 \prec \cdots \prec e_n$. Now the sorted seed can be regarded as a word made up of the characters $e_0, e_1, \ldots, e_n$, and we can insert it into the node whose path is specified by the ordered elements.

Suppose that the seed $E_{max}^{\{g_i,g_j\}}$ is into be inserted to node n. In **Lines 15–16**, we associate two sets $n.G$ and $n.E$ with the node n. We let the gene set $n.G = \{g_i, g_j\}$ and the experiment set $n.E = E_{max}^{\{g_i,g_j\}}$. (If the node n already exists, we let $n.G = n.G \cup \{g_i, g_j\}$.)

Example 3.12. Figure 6.8(a) shows the trie representation of the horizontal seeds in Figure 3.11(c). For instance, $E_{max}^{\{g_0,g_2\}} = \{e_0, e_1, e_3, e_5\}$ and $E_{max}^{\{g_2,g_3\}} = \{e_0, e_1, e_3, e_5\}$



(a) Horizontal seeds



(b) Predicted E sets

Figure 3.12: The trie representation of horizontal seeds and the experiment sets predicted from them. The edge labeled with i corresponds to the experiment e_i. The path from the root to node n represents the set of experiments $n.E$. The set associated with each node is the set of genes $n.G$. (a) Horizontal seeds from Figure 3.11(c). (b) The trie has been expanded to examine possible experiment sets. $M_E = 3$.

are inserted into the leftmost leaf by following the path "0,1,3,5." Hence, $n.G = \{g_0, g_2\} \cup \{g_2, g_3\}$ and $n.E = \{e_0, e_1, e_3, e_5\}$, assuming that this node is denoted by n.

Step 3: Predicting E from horizontal seeds

In **Lines 18–19**, the algorithm predicts the experiment set E by examining subsets of horizontal seeds. To this end, we exploit the property of the trie: for each node n encountered in the *post-order* traversal of the trie, the gene set $n.G$ is distributed to every node m in which $|m.E| = |n.E| - 1$ and $|m.E| \geq M_E$. Figure 6.8(b) shows the trie representation after Step 3 is performed on the trie in Figure 6.8(a) with $M_E = 3$.

After Step 3, the set $n.G$ represents an upper bound of the gene set that can form a bicluster with the experiment set $n.E$.

Step 4: Eliminating invalid predictions

In **Lines 21–22**, every node n in which $|n.G| < M_G$ is deleted. This step can be performed efficiently by a *pre-order* traversal of the trie. Genes were distributed in post-order in Step 3, and thus node n in the trie always has a superset of the genes its children have. Thus, if the node n has less than M_G genes, then none of its children can have more. For this reason, we can safely remove the entire subtree whose root is located at the node n. Figure 6.8(c) shows the trie after this step for the running example.

Step 5: Collecting exposed biclusters

After Step 4, it is possible that the pair $(n.G, n.E)$ can already be forming a bicluster for a certain node n. That is, for any 2×2 submatrix $\begin{bmatrix} x & y \\ z & w \end{bmatrix}$ in the matrix denoted by the pair $(n.G, n.E)$, $|x - z - y + w| \leq \delta$ for the parameter δ. In **Lines 24–25**, these biclusters are collected (and the node n is removed if it is a leaf). Figure 3.13(b) presents the trie after this step. Bicluster #0 in Figure 3.2(b) is found here.

The biclusters found in this step are only a by-product of our algorithm to predict experiment sets. In Section 3.4.2, we describe our main approach that derives gene set G from experiment set E by Eqn. 3.4.

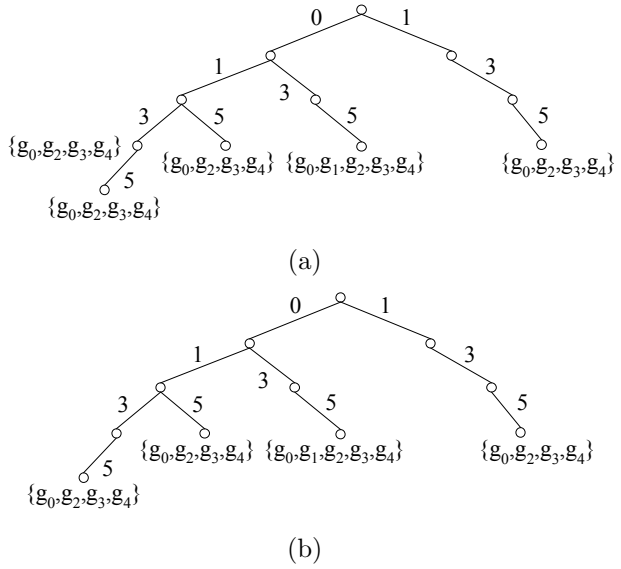


Figure 3.13: Continuation of the trie example in Figure 7.5. (a) After eliminating invalid predictions (Step 4). $M_G = 3$. (b) After collecting the exposed bicluster $(\{g_0, g_2, g_3, g_4\}, \{e_0, e_1, e_3\})$ from the parent node of the leftmost leaf (Step 5). This corresponds to bicluster #0 in Figure 3.2(b).

3.4.2 Calculating the gene set G

After completing Steps 1–5 of Algorithm 3.2, we apply Eqn. 3.4 to the experiment set $n.E$ of each remaining node n in the trie in order to find G, thus finalizing the biclustering process.

The worst-case complexity of the entire biclustering algorithm is due to the series of $\otimes$ operations in Eqn. 3.4. The scalability of the algorithm thus depends on how much the total number of $\otimes$ operations can be reduced and how efficiently a single $\otimes$ operation can be performed.

To reduce the number of $\otimes$ operations, we take an approach similar to dynamic programming. That is, the repetitive calculations are minimized by storing and reusing previously obtained partial results. For an efficient implementation of the operator $\otimes$, we take advantage of the ZBDDs, by which we can *symbolically* represent vertical seeds and *implicitly* perform $\otimes$ operations on them without enumerating all of the intermediate results.

Reducing the number of $\bigotimes$ operations

We use the following example to introduce this idea.

Example 3.13. Suppose that Eqn. 3.4 is applied to $E = \{e_0, e_1, e_3, e_5\}$. We then need to perform the $\otimes$ operations $\binom{4}{2} - 1 = 5$ times:

$$\mathcal{G} = \left\{G_{max}^{\{e_0,e_1\}}\right\} \otimes \left\{G_{max}^{\{e_0,e_3\}}\right\} \otimes \left\{G_{max}^{\{e_1,e_3\}}\right\} \otimes \left\{G_{max}^{\{e_0,e_5\}}\right\} \otimes \left\{G_{max}^{\{e_1,e_5\}}\right\} \otimes \left\{G_{max}^{\{e_3,e_5\}}\right\}.$$

In contrast, if we had already applied Eqn. 3.4 to $E' = \{e_0, e_1, e_3\}$ and saved the result into $\mathcal{G}'$, then we only need the following three $\otimes$ operations:

$$\mathcal{G}' \otimes \left(\left\{G_{max}^{\{e_0,e_5\}}\right\} \otimes \left\{G_{max}^{\{e_1,e_5\}}\right\} \otimes \left\{G_{max}^{\{e_3,e_5\}}\right\}\right). \tag{3.7}$$

In general, when applying Eqn. 3.4 to E with N elements, we can reduce the number of $\otimes$ operations from $\binom{N}{2} - 1$ to $(N-1)$ by exploiting previous results. This idea can be realized efficiently by the trie, since it has a hierarchical structure. Algorithm 3.3 shows the outline of our approach.

In the algorithm, each node n is associated with $n.\mathcal{G}$, a set of gene sets. For each node n visited in the *pre-order* traversal of the trie, we perform the following procedure. If the set $n.E$ has only two elements e_m, e_n, then the set $n.\mathcal{G}$ is made equal to the set of vertical seeds $G_{max}^{\{e_m,e_n\}}$ in **Lines 2–3**. This is the base case. Otherwise, the intermediate result is computed in **Lines 5–9**, which corresponds to Eqn. 3.7 $\left(\left\{G_{max}^{\{e_0,e_5\}}\right\} \otimes \left\{G_{max}^{\{e_1,e_5\}}\right\} \otimes \left\{G_{max}^{\{e_3,e_5\}}\right\}\right)$ in Example 3.13. If any intermediate result is empty or has fewer than M_G elements, we stop and remove the entire subtree rooted at n.

Example 3.14. In Figure 3.15(a), the intermediate results for the leaves are indicated by $\mathcal{G}_{0135}$, $\mathcal{G}_{015}$, $\mathcal{G}_{035}$, and $\mathcal{G}_{135}$. Here, $\mathcal{G}_{0135} = \left\{G_{max}^{\{e_0,e_5\}}\right\} \otimes \left\{G_{max}^{\{e_1,e_5\}}\right\} \otimes \left\{G_{max}^{\{e_3,e_5\}}\right\}$ $= \{\{g_0\}, \{g_0, g_2\}, \{g_2, g_3\}\}$. However, all the sets in $\mathcal{G}_{0135}$ have less than M_G elements. Thus, we set $\mathcal{G}_{0135} = \emptyset$ and remove the leftmost leaf. Similarly, $\mathcal{G}_{015} = \emptyset$, and its corresponding leaf is deleted. On the other hand, $\mathcal{G}_{035} = \{\{g_1, g_2, g_3\}\}$ and $\mathcal{G}_{135} = \{\{g_0, g_2, g_3\}\}$ (after the sets with too few elements are removed). Figure

Algorithm 3.3: Calculating gene set G

input : The trie from Algorithm 3.2; Vertical seeds
output: Maximal biclusters

foreach *node* n *in the pre-order traversal of the trie* **do**
  if $|n.E| == 2$ **then**
    $n.\mathcal{G} = \{G_{max}^{\{n.E\}}\}$;
  else
    $E = n.E$;
    $E' = n'.E$, where n' is the parent node of n;
    $e_n = E - E'$;
    foreach e *in* E' **do**
      $n.\mathcal{G} = n.\mathcal{G} \otimes \{G_{max}^{\{e,e_n\}}\}$;
    $n.\mathcal{G} = n.\mathcal{G} \otimes n'.\mathcal{G}$;
  if $n.\mathcal{G} = \emptyset$ **then**
    Remove n and its subtrie rooted at n;
  else if $|E| \geq M_E$ **then**
    foreach G *in* $n.\mathcal{G}$ **do**
      Collect (G, E);
  Report maximal biclusters;

Figure 3.14: Algorithm to calculate gene set G.

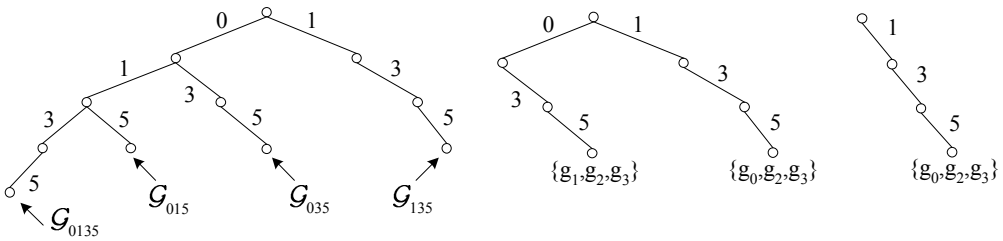


Figure 3.15: Continuation of the trie example in Figure 3.13. Bicluster #1 in Figure 3.2(b) is found from the trie in (c).

3.15(b) finally shows the trie in which the two left leaves are removed from the trie in Figure 3.15(a).

In **Line 10**, the resulting $\mathcal{G}_n$, corresponding to Eqn. 3.7 itself, is calculated. If this set $\mathcal{G}_n$ is empty, we prune the entire subtree rooted at n (**Lines 11–12**). We otherwise collect biclusters (**Lines 13–15**). Figure 3.15(c) presents the final trie in which bicluster #1 in Figure 3.2(b) is found at the leaf node.

Efficient implementation of $\otimes$

The operator $\otimes$ is implemented using the basic set operators on ZBDDs, such as $\cap$ and $\cup$. Typically, these operators are recursively defined on ZBDDs with trivial terminal cases, such as $\mathcal{P} \cap \emptyset = \emptyset$ or $\mathcal{P} \cup \emptyset = \mathcal{P}$. Thus, the operator $\otimes$ is also recursively defined on ZBDDs. That is, the initial ZBDDs are recursively partitioned into two smaller ZBDDs until the terminal cases are encountered. Then, by merging the solutions to smaller subproblems in a bottom-up manner, we can obtain the result of the $\otimes$ operation on the original ZBDDs.

We first show how to partition a set of subsets into two smaller sets. Let $\mathcal{P}$ be a set of subsets. We partition $\mathcal{P}$ into $\mathcal{P}_1$ and $\mathcal{P}_0$ with respect to the variable x in such a way that $\mathcal{P}_1$ contains all of the subsets that include x, while $\mathcal{P}_0$ includes all of the other subsets.

Example 3.15. Let $\mathcal{P} = \left\{ G_{max}^{\{e_0,e_5\}} \right\} = \{\{g_0, g_2, g_4\}, \{g_1, g_2, g_3, g_5\}\}$, from the example in Table 3.2(c). Then $\mathcal{P}_1 = \{\{g_0, g_2, g_4\}\}$ and $\mathcal{P}_0 = \{\{g_1, g_2, g_3, g_5\}\}$, assuming $x = g_0$.

With respect to the topmost vertex of a ZBDD, we can perform this partitioning by simply recognizing two subgraphs – the subgraphs connected by the 1-edge and 0-edge correspond to $\mathcal{P}_1$ and $\mathcal{P}_0$, respectively.

Based on this partitioning, we can recursively perform various operations on ZBDDs. For example, $\mathcal{P} \cup \mathcal{Q} = (\mathcal{P}_0 \cup \mathcal{P}_1) \cup (\mathcal{Q}_0 \cup \mathcal{Q}_1) = (\mathcal{P}_0 \cup \mathcal{Q}_0) \cup (\mathcal{P}_1 \cup \mathcal{Q}_1)$, as shown in Figure 3.16(a) . The problem of $\mathcal{P} \cup \mathcal{Q}$ can now become two smaller problems, $(\mathcal{P}_0 \cup \mathcal{Q}_0)$ and $(\mathcal{P}_1 \cup \mathcal{Q}_1)$.

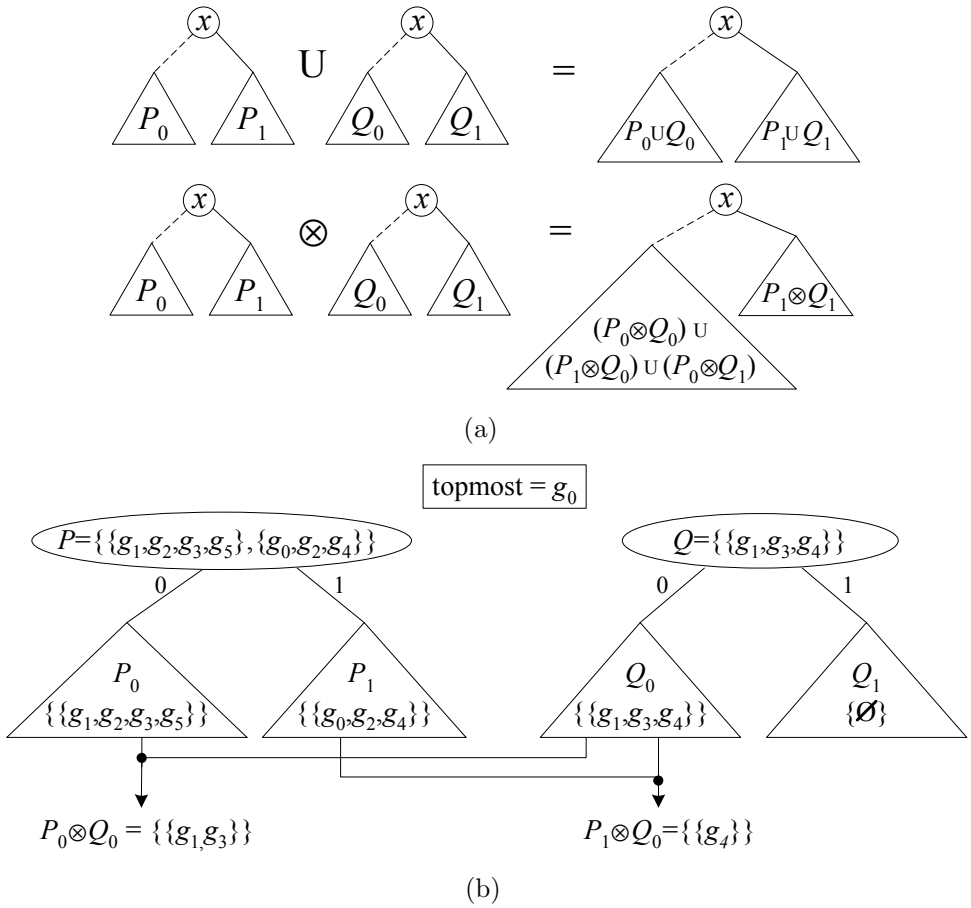


Figure 3.16: The operators ∪ and ⊗ on ZBDDs. (a) The set operators are recursively defined on the ZBDDs. The operator ∩ is defined in the same way as the operator ⊗ but is not shown here. (b) An example.

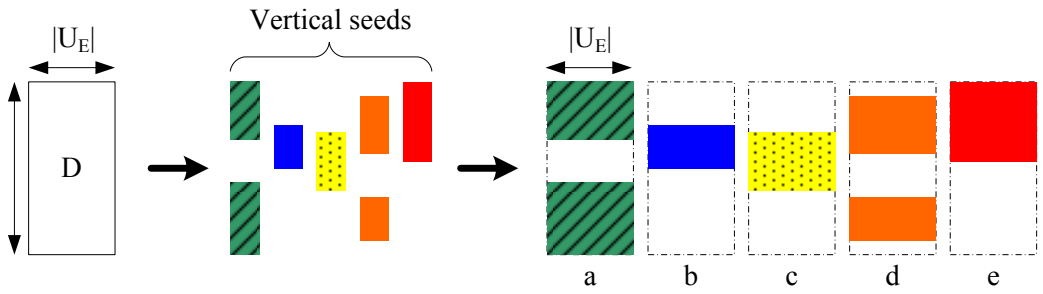


Figure 3.17: Dividing a large data matrix into submatrices of manageable sizes.

We can similarly compute $\mathcal{P} \otimes \mathcal{Q}$ by recursively solving and merging four subproblems: $(\mathcal{P}_0 \otimes \mathcal{Q}_0)$, $(\mathcal{P}_1 \otimes \mathcal{Q}_0)$, $(\mathcal{P}_0 \otimes \mathcal{Q}_1)$, and $(\mathcal{P}_1 \otimes \mathcal{Q}_1)$. Figure 3.16(b) depicts the decomposition with respect to the topmost variable. The right subgraph includes $(\mathcal{P}_1 \otimes \mathcal{Q}_1)$, and the left subgraph contains the others, since only $(\mathcal{P}_1 \otimes \mathcal{Q}_1)$ can have a subset with the topmost variable. Further implementation details can be found in [14, 15, 69, 70].

Example 3.16. Let $\mathcal{P} = \left\{ G_{max}^{\{e_0,e_5\}} \right\} = \{\{g_0, g_2, g_4\}, \{g_1, g_2, g_3, g_5\}\}$ and $\mathcal{Q} = \left\{ G_{max}^{\{e_2,e_5\}} \right\}$ $= \{\{g_1, g_3, g_4\}\}$ from the examples in Table 3.2. Then $\mathcal{P} \otimes \mathcal{Q} = (\mathcal{P}_0 \otimes \mathcal{Q}_0) \cup (\mathcal{P}_1 \otimes \mathcal{Q}_0)$ $= \{\{g_1, g_3\}, \{g_4\}\}$, as shown in Figure 3.16(b), where we partition the sets with respect to the variable g_0. Both $(\mathcal{P}_0 \otimes \mathcal{Q}_1)$ and $(\mathcal{P}_1 \otimes \mathcal{Q}_1)$ are empty sets and are not shown in the figure.

3.4.3 Considerations for very large-scale expression data

We present a divide-and-conquer technique that is useful in the analysis of very large-scale data sets. This technique enables us to split the whole expression data matrix into submatrices, without any compromise in cluster discovery. (Thus, we can still find all maximal biclusters on the data matrix.) The resulting submatrices will be small enough to apply our algorithm as explained in the previous sections, even if the original data matrix is so huge that the algorithm is not applicable.

The basic idea is to split the data matrix into submatrices specified by $\left(G_{max}^{\{e_m,e_n\}}, U_E \right)$ for all $\{e_m, e_n\}$ in U_E.

Example 3.17. In Figure 3.17, we assume that five vertical PMBs exist in data

matrix D. For each vertical PMB, we can expand it and generate a submatrix in which the rows are the genes in the PMB and the columns are U_E.

The rationale is that for any bicluster (G, E), G is always a subset of a certain vertical seed $G_{max}^{\{\cdot\}}$. Thus, the biclusters found from all possible submatrices $(G_{max}^{\{\cdot\}}, U_E)$ are equivalent to those discovered from the whole matrix (U_G, U_E). Moreover, this split can be done very quickly, since $G_{max}^{\{\cdot\}}$ generation is trivial even for large-scale data, as explained in Section 3.2.2. For most gene expression data, $G_{max}^{\{\cdot\}} \ll |U_G|$ and $|U_E| \ll |U_G|$, so the size of each submatrix is manageable. We summarize our approach as follows:

(1) All vertical PMBs are discovered from data matrix D.

(2) A submatrix $(G_{max}^{\{\cdot\}}, U_E)$ is created for each $G_{max}^{\{\cdot\}}$ in a vertical PMB.

(3) The biclustering procedure presented in the previous sections is executed on each submatrix.

3.4.4 Algorithm complexity

The problem of biclustering is inherently intractable [21, 98, 107], and the worst-case complexity of our algorithm is exponential in the number of columns in the input data matrix. However, the execution time on typical benchmarks is practical, as will be shown in Chapter 5. Furthermore, our algorithm can find all the biclusters that satisfy specific input conditions. This is due to efficient techniques such as the ZBDD-based manipulations and the dynamic programming approach, which enable us to avoid the exhaustive and explicit enumeration of the intermediate results. When ZBDDs are used, the computational complexity of a problem depends on the size of its ZBDD representation, which often has mild growth with the problem size. Thus, it has been reported by numerous independent research studies that ZBDDs may be used to efficiently solve many practical instances of intractable problems [15, 16, 26, 67, 69, 70, 88].

Additionally, our algorithm works in polynomial time if the input data matrix and parameters satisfy a certain condition. Suppose that, for any bicluster (G, E)

defined under a specific input parameters, there always exist at least one $G_{max}^{\{\cdot\}}$ and at least one $E_{max}^{\{\cdot\}}$ such that $G_{max}^{\{\cdot\}} = G$ and $E_{max}^{\{\cdot\}} = E$. Then, we can perform biclustering by simply seeing if each pair $\left(G_{max}^{\{\cdot\}}, E_{max}^{\{\cdot\}}\right)$ is a legitimate bicluster without invoking Algorithms 3.2 and 3.3. Then, the entire biclustering process can be done in polynomial time. We refer the interested reader to [119] for further details.

3.5 Summary

In this chapter, we investigated the problem of finding biclusters from gene expression data sets. We mathematically characterized this problem and proposed a suite of new algorithms to find biclusters with coherent values. The proposed method employed dynamic programming and a divide-and-conquer technique, as well as efficient data structures such as the trie and zero-suppressed decision diagrams (ZBDDs). In particular, the use of ZBDDs enabled us to substantially extend the scalability of the method. We conducted experimental studies including statistical and biological validations of the biclusters produced by our method. The results, which will be covered in Chapter 5, demonstrates the effectiveness of our ZBDD-based biclustering approach.

Chapter 4

Finding Nested Biclusters

Chapter 3 described a ZBDD-based biclustering algorithm. There, a bicluster was modeled by a matrix in which every 2×2 submatrix satisfies a constraint. Consequently, any sub-bicluster of a bicluster in this definition is yet another bicluster satisfying the same constraint. In general, suppose that $\mathfrak{D}$ is a certain condition under which a bicluster, P, is defined. Let the bicluster P be called *nested* if any legitimate sub-bicluster of P also satisfies the condition $\mathfrak{D}$. Then, many biclusters appearing in the literature are in fact nested biclusters. Examples of nested biclusters in the literature include *xMOTIFs* [74], *δ-valid kj-patterns* [18], *GEMS* [112], *OPSMs* [10], *OP-Clusters* [61], and *δ-pClusters* [107], just to name a few.

This chapter describes an algorithm to find nested biclusters. This algorithm is a generalized version of the algorithm presented in Chapter 3 and also depends upon the ZBDDs to represent and manipulate massive data. We will see that biclusters having different definitions can be found in a unified framework as long as these biclusters are nested.

This chapter is organized as follows. Section 4.1 presents the formal definition of nested biclusters our method can find. In Sections 4.2 and 4.3, we explain at length the proposed method, which consists of essentially two stages. The first stage, which is detailed in Section 4.2, is to find special nested biclusters called *atomic biclusters*. Section 4.3 presents the second stage of our method, which derives general (non-atomic) nested biclusters from the atomic biclusters previously found. Our

experimental results can be found in the subsequent chapters.

4.1 Definitions and overview

Within a unified framework, our approach can find various types of nested biclusters. In particular, we focus on finding three specific types of nested biclusters in this chapter. Their formal definitions are provided in Section 4.1.1. Some biological intuition behind these definitions is presented in Section 4.1.2. The problem statement and an overview of our approach will follow in Sections 4.1.3 and 4.1.4, respectively.

4.1.1 Definition of nested biclusters

Let A denote an input data matrix of real numbers with set of rows $R = \{1, 2, \ldots, n\}$ and set of columns $C = \{1, 2, \ldots, m\}$. That is, $A \in \mathbb{R}^{n \times m}$. We also denote the matrix A by pair (R, C). We first provide a formal definition of a nested bicluster.

Definition 4.1. *Given $A = (R, C)$, an input matrix, and $\mathfrak{D}$, a certain condition defined on a matrix, let pair $P = (I, J)$ denote a submatrix of A, namely, $I \subseteq R$ and $J \subseteq C$. The submatrix P is called a* bicluster *appearing in A under $\mathfrak{D}$, if P satisfies the condition $\mathfrak{D}$.*

Example 4.1. *Figure 4.1(a) presents matrix $A \in \mathbb{R}^{4 \times 4}$ with $R = C = \{1, 2, 3, 4\}$. Let $\mathfrak{D}$ define a matrix in which the values on each row are constant. Figure 4.1(b) shows P_1, P_2, P_3, some biclusters appearing in the matrix A under the condition $\mathfrak{D}$.*

Definition 4.2. *Let P be a bicluster appearing in matrix A under condition $\mathfrak{D}$. The bicluster P is called* nested*, if any subset (or submatrix) of P is also a bicluster appearing in the matrix A under the condition $\mathfrak{D}$.*

Example 4.2. *In Figure 4.1(b), it can be easily verified that any submatrix of P_1, P_2, and P_3 is another bicluster appearing in the matrix A under the same condition $\mathfrak{D}$, since the values on each row of such a submatrix remain constant. Thus, P_1, P_2, and P_3 are all nested biclusters.*

	1	2	3	4
1	1.0	1.0	1.0	1.0
2	1.0	1.0	1.0	3.0
3	2.0	2.0	3.0	2.0
4	3.0	3.0	2.0	3.0

(a) Input matrix A

	1	2	3
1	1.0	1.0	1.0
2	1.0	1.0	1.0

P_1

	1	2	4
1	1.0	1.0	1.0
3	2.0	2.0	2.0
4	3.0	3.0	3.0

P_2

	1	2
1	1.0	1.0
2	1.0	1.0
3	2.0	2.0
4	3.0	3.0

P_3

(b) Biclusters P_1, P_2, and P_3

Figure 4.1: Example of an input matrix and some biclusters appearing in the matrix. The condition $\mathfrak{D}$ here defines a matrix in which the values on each row are constant.

Table 4.1: Classification of nested biclusters.

Bicluster	Definition	Property	Example	Related biclusters in the literature
Type 1	Definition 4.4	Property 4.1	Figure 4.1, 4.2	xMOTIFs [74], δ-valid kj-pattern [18], GEMS [112]
Type 2	Definition 4.5	Property 4.2	Figure 4.3	OPSM [10], OP-Cluster [61]
Type 3	Definition 4.6	–	Figure 4.4	δ-bicluster [21], δ-pCluster [107, 119], FLOC cluster [113]

We introduce three types of nested biclusters that can be found by our bicluster mining method. Table 4.1 is for a quick lookup of related information. In what follows, the term *bicluster* always means a nested bicluster, unless otherwise stated.

Type 1 biclusters

Definition 4.3. *For any set S on $\mathbb{R}$, the range of S, denoted by* $\text{RANGE}(S)$*, is the difference between the largest and the smallest elements of S.*

Definition 4.4. *Given matrix $A = (R, C)$ and threshold $\tau \geq 0$, a* Type 1 bicluster *is a matrix denoted by (I, J) such that* (1) *$I \subseteq R$ and $J \subseteq C$; and* (2) *for each $i \in I$, $\text{RANGE}(\{a_{ij} | \forall j \in J\}) \leq \tau$.*

Example 4.3. *Figure 4.2 presents an input matrix and some Type 1 biclusters appearing in the matrix with respect to the parameter $\tau = 0.5$.*

Type 1 biclusters are a representative example of the biclusters that have a one-row-based or one-column-based definition. Examples include a bicluster with constant values on rows, as seen in Figure 4.1, or with constant values on columns. In the literature, biclusters such as *xMOTIFs* [74], *δ-valid kj-patterns* [18] and *GEMS* [112] belong to this type.

The reader can easily verify that any Type 1 bicluster is nested. Furthermore, the following property holds for Type 1 biclusters.

Property 4.1. *If (I_1, J_1) and (I_2, J_2) are both Type 1 biclusters with respect to τ, then the bicluster $(I_1 \cup I_2, J_1 \cap J_2)$ is also Type 1 with respect to τ.*

	1	2	3	4
1	1.0	1.8	1.2	1.4
2	2.0	2.2	1.6	2.4
3	3.0	3.4	5.2	1.0
4	2.5	2.7	4.1	3.1

(a) Input matrix A

	1	2
2	2.0	2.2
3	3.0	3.4
4	2.5	2.7

P_1

	2	4
1	1.8	1.4
2	2.2	2.4
4	2.7	3.1

P_2

	1	3
1	1.0	1.2
2	2.0	1.6

P_3

	1	4
1	1.0	1.4
2	2.0	2.4

P_4

(b) Type 1 biclusters P_1, P_2, P_3 and P_4

Figure 4.2: Type 1 biclusters appearing in the input matrix A (the parameter $\tau = 0.5$).

Example 4.4. *The biclusters shown in Figure 4.1(b) satisfies Definition 4.4 with respect to $\tau = 0$, and thus P_1, P_2, and P_3 are all Type 1 biclusters. Let $P_1 = (I_1, J_1)$, $P_2 = (I_2, J_2)$, and $P_3 = (I_3, J_3)$. Then, $I_3 = I_1 \cup I_2$ and $J_3 = J_1 \cap J_2$. Thus, Property 4.1 holds for these biclusters.*

Type 2 biclusters

Definition 4.5. *Given matrix $A = (R, C)$, let $J \subseteq C$ be a set of size $k \geq 2$ and let $\langle o_1, o_2, \ldots, o_k \rangle$ be a linear ordering of J. A* Type 2 bicluster *is a matrix denoted by (I, J) such that (1) $I \subseteq R$; and (2) for each $i \in I$, $a_{io_1} > a_{io_2} > \cdots > a_{io_k}$.*

Example 4.5. *Figure 4.3 presents a data matrix and Type 2 biclusters appearing in it. The order of the values on each row is preserved. For example, for $i \in I = \{1, 2\}$ in P_1, $a_{i1} > a_{i4} > a_{i2}$; for $i \in I = \{1, 2, 3\}$ in P_4, $a_{i3} > a_{i4} > a_{i2}$.*

Type 2 biclusters are a representative example of the biclusters in which the order of the values (or some states defined by them) on a row or column is preserved for

	1	2	3	4
1	3.0	1.0	4.0	2.0
2	4.0	1.0	3.0	2.0
3	2.0	1.0	4.0	3.0

(a) Input matrix A

	1	2	4
1	3.0	1.0	2.0
2	4.0	1.0	2.0

P_1

	2	3	4
1	1.0	4.0	2.0
2	1.0	3.0	2.0

P_2

	2	3	4
2	1.0	3.0	2.0
3	1.0	4.0	3.0

P_3

	2	3	4
1	1.0	4.0	2.0
2	1.0	3.0	2.0
3	1.0	4.0	3.0

P_4

(b) Type 2 biclusters P_1, P_2, P_3 and P_4

Figure 4.3: An example of Type 2 biclusters.

the other rows or columns as well. Examples in the literature include *OPSMs* [10] and *OP-Clusters* [61].

It can easily be verified that a Type 2 bicluster is nested. In addition, the following property holds for Type 2 biclusters.

Property 4.2. *If both (I_1, J_1) and (I_2, J_2) are Type 2 biclusters, then the bicluster $(I_1 \cup I_2, J_1 \cap J_2)$ is also Type 2.*

Example 4.6. *Property 4.2 holds for the biclusters shown in Figure 4.3(b). For example, let $P_2 = (I_2, J_2)$, $P_3 = (I_3, J_3)$, and $P_4 = (I_4, J_4)$. Then, it can be verified that $I_4 = I_2 \cup I_3$ and $J_4 = J_2 \cap J_3$.*

Type 3 biclusters

Definition 4.6. *Given matrix $A = (R, C)$ and threshold $\tau \geq 0$, a* Type 3 bicluster *is a matrix denoted by $P = (I, J)$ such that* (1) *$I \subseteq R$ and $J \subseteq C$; and* (2) *for any 2×2 submatrix $\begin{bmatrix} e & f \\ g & h \end{bmatrix}$ in P, $|e - g - f + h| \leq \tau$.*

	1	2	3	4	5
1	6.0	9.0	2.0	5.0	4.0
2	2.0	3.0	4.0	3.0	6.0
3	2.0	2.0	3.0	6.0	7.0
4	3.0	4.0	5.0	2.0	6.0
5	4.0	7.0	3.0	1.0	4.0

(a) Input matrix A

	3	5
1	2.0	4.0
2	4.0	6.0
4	5.0	6.0
5	3.0	4.0

P_1

	1	2	3
2	2.0	3.0	4.0
3	2.0	2.0	3.0
4	3.0	4.0	5.0

P_2

	4	5
2	3.0	6.0
4	2.0	6.0
5	1.0	4.0

P_3

(b) Type 3 biclusters P_1, P_2, and P_3

Figure 4.4: An example of Type 3 biclusters (the parameter $\tau = 1$).

Example 4.7. *Figure 4.4 shows a data matrix and some Type 3 biclusters appearing in the matrix with respect to the parameter* $\tau = 1$.

Type 3 biclusters are used to model a matrix in which the elements exhibit some coherent behavior. Examples include a matrix in which the value of the elements fluctuate in harmony and a matrix in which all elements have the same value. Type 3 biclusters in Definition 4.6 are in essence equivalent to *δ-pClusters* [107] and closely related to *δ-biclusters*[1] [21] and *FLOC clusters* [113].

The reader can verify that Type 3 biclusters are nested. However, Properties 4.1 and 4.2 do not necessarily hold for Type 3 biclusters. Also note that the same set of Type 3 biclusters can be found from input A and the transpose of A. This is because two matrices $\begin{bmatrix} e & f \\ g & h \end{bmatrix}$ and $\begin{bmatrix} e & g \\ f & h \end{bmatrix}$ are indistinguishable in the definition since $|e - g - f + h| = |(e - g) - (f - h)| = |(e - f) - (g - h)|$.

4.1.2 Biology behind the definitions of biclusters

The three types of nested biclusters were defined in such a way that they can effectively capture important biological phenomena involved in various applications. For example, in gene co-regulation analysis, researchers are often interested in recognizing common fluctuations in the expression levels of multiple genes. Finding Type 2 and Type 3 biclusters from gene expression data matrices may be useful in this application. Discovering Type 1 biclusters can provide some biological insight for applications such as the task of marker gene identification, where we are interested in correlating the activity of one or more genes to specific subphenotypes and thus finding genes expressed only in some phenotypes. For more examples, the reader can refer to [65] as well as the references listed in Table 4.1.

[1] δ-biclusters are not nested biclusters, since a subcluster of a δ-bicluster is not necessarily a δ-bicluster [21, 107]. However, δ-biclusters are included here because they also aim at modeling coherent behavior of matrix elements, and it has been reported that δ-biclusters are closely related to δ-pClusters in many aspects [107, 119].

4.1.3 Problem Statement

Given an input data matrix $A = (R, C)$, a specific definition $\mathfrak{D} \in \{$Definition 4.4, Definition 4.5, Definition 4.6$\}$, and the parameters specified in $\mathfrak{D}$, the problem of bicluster mining is to find all maximal nested biclusters $P = (I, J)$ appearing in A under $\mathfrak{D}$. We search only maximal[2] biclusters or those that are not contained by other biclusters as a submatrix, since non-maximal biclusters contain redundant information. Optionally, we can specify the minimum size of biclusters in order not to generate too small biclusters.

4.1.4 Overview of our approach

Our bicluster mining algorithm consists of essentially two steps. The first step is to find special biclusters called *atomic biclusters*. The second step is to derive other general (non-atomic) biclusters from the atomic biclusters previously found. These two steps are detailed in Sections 4.2 and 4.3, respectively. Fig 4.5 provides a flowchart of our method, and Tables 4.2 and 4.3 list related information for a quick reference.

Table 4.2: Step 1 - finding atomic biclusters.

Atomic bicluster	Definition	Algorithm	Example
Type 1	Definition 4.7	Algorithm 4.1	Figure 4.7
Type 2	Definition 4.8	Algorithm 4.2	Figure 4.9
Type 3	Definition 4.9	Algorithm 4.3	Figure 4.11

Table 4.3: Step 2 - deriving non-atomic biclusters from atomic biclusters.

Method	Details	Algorithm	Based upon
Breadth-first	Section 4.3.3	Algorithm 4.4	Corollary 4.1
Depth-first	Section 4.3.3	Algorithm 4.5	Corollary 4.2

[2]Formally, a bicluster $P = (I, J)$ is called *maximal* if there is no bicluster $P' = (I', J')$ such that $I \subseteq I'$ and $J \subseteq J'$ under the identical input conditions.

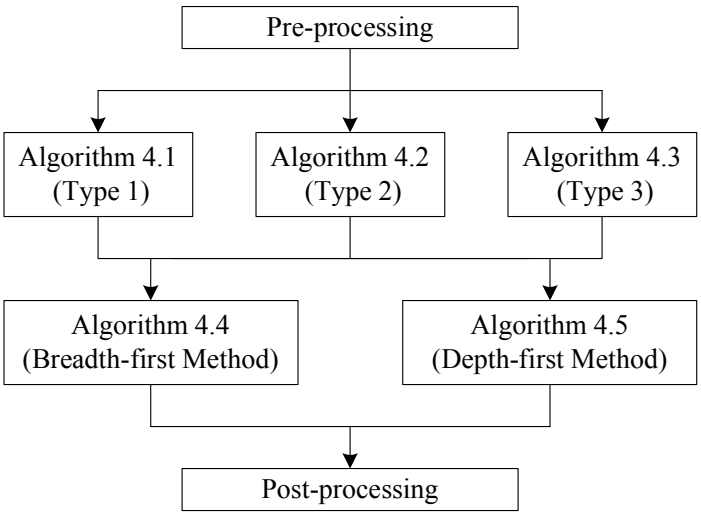


Figure 4.5: A flowchart of our method. The first step (Algorithms 4.1, 4.2, and 4.3) is to find atomic biclusters. The second step (Algorithms 4.4 and 4.5) is to derive non-atomic biclusters.

4.1.5 Notation

Table 4.4 lists some important notations that will be used throughout the chapter, especially in Section 4.3.

4.2 Finding atomic biclusters

Informally, an atomic bicluster is represented by a matrix that has only one row (Type 1) or two rows (Types 2 and 3) but as many columns as possible. In this section, we provide the formal definition of atomic biclusters and specific algorithms to find them.

4.2.1 Finding Type 1 atomic biclusters

Definition 4.7. *Given input matrix $A = (R, C)$ and threshold $\tau \geq 0$, a* Type 1 atomic bicluster *for row $i \in R$ is a one-row matrix, denoted by pair $P = (\{i\}, J)$, that satisfies the following:* (1) *P is a Type 1 bicluster on A; and* (2) *there is no J' such that $J' \supset J$ and $(\{i\}, J')$ is also a Type 1 bicluster.*

Table 4.4: Notations.

Notation	Meaning
$A = (R, C)$	Input matrix A with row set R and column set C; $A \in \mathbb{R}^{\|R\| \times \|C\|}$.
$P = (I, J)$	Bicluster P with row set I and column set J; $I \subseteq R$ and $J \subseteq C$.
$\mathcal{J}$	Set of column index sets.
v	Vertex in the lattice graph (Algorithms 4.4 and 4.5).
$v.I$	Row index set associated with vertex v.
$v.\mathcal{J}$	Set of column index sets associated with vertex v.
$\mathfrak{J}$	Function $\mathfrak{J}$ (Definition 4.11).
$\mathfrak{J}(I)$	Image of set I under function $\mathfrak{J}$; essentially, set of column index sets.
$\mathfrak{J}_1, \mathfrak{J}_2, \mathfrak{J}_3$	Function $\mathfrak{J}$ with explicit type specification.

The condition (2) in the above definition is to avoid generating those atomic biclusters that are contained by others, since such biclusters are redundant.

Algorithm 4.1 details our approach to find Type 1 atomic biclusters of Definition 4.4. The key idea of this algorithm is simple: when the elements of a set S are sorted and arranged in the corresponding order, $range(S)$ is simply the absolute difference between the first and the last elements of S. The worst-case complexity of the algorithm is polynomial in $|C|$, and the maximum number of atomic biclusters found per row by Algorithm 4.1 is $(|C| - 1)$.

In **Lines 1–4**, the column indices are sorted in ascending order according to the value of the corresponding elements. The variables *begin* and *end* in **Lines 5–6** are to point to the first and the last elements of the sub-array under consideration at some point. Inside the while loop in **Lines 7–16**, J, the column set of an atomic bicluster, is generated as the variables *begin* and *end* are incremented. Note that multiple J can exist per row and overlap with each other. Since the array D is sorted, the algorithm only needs to compare in Line 8 the first element ($D[begin]$) and the last element ($D[end]$), in order to see if all the elements in the sub-array are similar. **In Lines 8–9**, the variable *end* is extended as long as $D[end].val - D[begin].val \leq \tau$. The algorithm reports J in **Line 11** or **Line 13**. **Lines 14–16** are to adjust the variable *begin* appropriately after one instance of J is found, because multiple overlapping

Algorithm 4.1: Finding Type 1 atomic biclusters for one row

input : $A = (R, C)$, a data matrix
input : $i \in R$, a row index
input : τ, a threshold
output: $J \subseteq C$, $(\{i\}, J)$ is a Type 1 atomic bicluster

```
1  foreach j ∈ C do
2      D[j].val := a_ij;
3      D[j].ind := j;
4  Sort array D in ascending order with respect to the field val;
5  begin := 1;
6  end := 2;
7  while (end ≤ |C|) do
8      if (D[end].val − D[begin].val ≤ τ) then
9          end := end + 1;
10         if (end > |C|) then
11             Report J = {D[begin].ind, ..., D[end − 1].ind};
12     else
13         Report J = {D[begin].ind, ..., D[end − 1].ind};
14         repeat
15             begin := begin + 1;
16         until (begin = end) or
           (D[end].val − D[begin].val ≤ τ);
```

Figure 4.6: Algorithm to find Type 1 atomic biclusters.

	1	*2*	*3*	*4*
1	1.0	1.8	1.2	1.4
2	2.0	2.2	1.6	2.4
3	3.0	3.4	5.2	1.0
4	2.5	2.7	4.1	3.1

(a)

Row $i \in R$	Column set $J \in 2^C$
1	$\{1, 3, 4\}$, $\{2, 4\}$
2	$\{1, 2, 4\}$, $\{1, 3\}$
3	$\{1, 2\}$
4	$\{1, 2\}$, $\{2, 4\}$

(b)

Figure 4.7: (a) Input matrix A. (b) Type 1 atomic biclusters found from A by Algorithm 4.1 ($\tau = 0.5$).

Algorithm 4.2: Finding Type 2 atomic biclusters for pair of rows

input : $A = (R, C)$, a data matrix
input : $q, r \in R$, row indices ($q \neq r$)
input : $min_J \geq 0$, minimum column size
output: $J \subseteq C$, $(\{q, r\}, J)$ is an atomic Type 2 bicluster

1 **foreach** $j \in C$ **do**
2 $\quad \hat{X}[j].val := a_{qj}$;
3 $\quad \hat{Y}[j].val := a_{rj}$;
4 $\quad \hat{X}[j].ind := \hat{Y}[j].ind := j$;
5 Sort arrays $\hat{X}$ and $\hat{Y}$ in descending order with respect to the field val;
6 Construct sequence $X = \langle x_1, x_2, \ldots, x_{|C|} \rangle$ such that $x_i = \hat{X}[i].ind$;
7 Construct sequence $Y = \langle y_1, y_2, \ldots, y_{|C|} \rangle$ such that $y_i = \hat{Y}[i].ind$;
8 Find $\mathcal{Z} = \{Z | Z$ is an MCS of X and Y, $|J| >= min_J\}$;
9 **foreach** $Z \in \mathcal{Z}$ **do**
10 $\quad$ Convert $Z = \langle z_1, z_2, \ldots \rangle$ to $J = \{z_1, z_2, \ldots\}$;
11 $\quad$ Return J;

Figure 4.8: Algorithm to find Type 2 atomic biclusters.

instances of J can be found for each row.

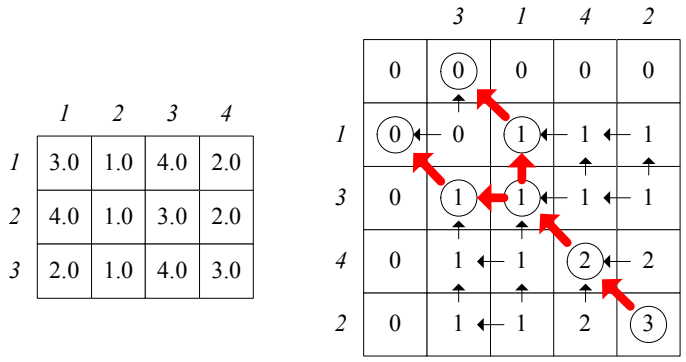
Example 4.8. *Figure 4.7(b) presents the Type 1 atomic biclusters discovered by Algorithm 4.1 from the data matrix in Figure 4.2(a), repeated here in Figure 4.7(a) for convenience. The parameter used is* $\tau = 0.5$.

4.2.2 Finding Type 2 atomic biclusters

Definition 4.8. *Given input matrix* $A = (R, C)$, *a* Type 2 atomic bicluster *for rows* $i, k \in R$ $(i \neq k)$ *is a two-row matrix, denoted by pair* $P = (\{i, k\}, J)$, *that satisfies the following:* (1) *P is a Type 2 bicluster on A; and* (2) *there is no J' such that $J' \supset J$ and $(\{i, k\}, J')$ is also a Type 2 bicluster.*

Our approach to find Type 2 atomic biclusters is outlined in Algorithm 4.2. The main problem is to find a largest two-row matrix in which the order of the values on each row is preserved. The key is to exploit algorithms to solve the problem of finding *maximal common subsequences* (MCS) of two sequences [24, 36].

Besides an input data matrix, Algorithm 4.2 takes an additional input parameter,



(a) (b)

Row set $\{q, r\}$	Column set $J \in 2^C$
$\{1, 2\}$	$\{1, 2, 4\}$, $\{2, 3, 4\}$
$\{2, 3\}$	$\{2, 3, 4\}$
$\{1, 3\}$	$\{1, 2, 3\}$, $\{2, 3, 4\}$

(c)

Figure 4.9: (a) Input data. (b) Finding MCS for rows 1 and 2. (c) Type 2 atomic biclusters found by Algorithm 4.2 ($min_J = 3$).

min_J, to specify the minimum cardinality of the column set of an atomic bicluster. This is to limit the total number of atomic biclusters per pair of rows.

In **Lines 1–5**, the elements of each row are sorted with respect to their value, and the column indices are ordered accordingly. **Lines 6–7** are to convert arrays of column indices to sequences. In **Line 8**, an MCS-search algorithm is invoked. In **Lines 9-11**, each MCS found is converted to a set and returned.

The MCS problem has been extensively studied in the literature, and the typical solution relies on dynamic programming [24, 36]. The worst-case complexity of an algorithm to solve the MCS problem is polynomial in the length of sequences [24,36]. Some details of an MCS algorithm can be found in the following example.

Example 4.9. *Figure 4.9(c) presents the Type 2 atomic biclusters discovered by Algorithm 4.2 from the data matrix in Figure 4.3(a), repeated here in Figure 4.9(a) for convenience. The parameter used is* $\min_J = 3$. *We can solve the MCS problem by modeling it as a sequence alignment problem [36]. In a sequence alignment problem,*

the scores for a match, a mismatch, and a space should first be assigned. For the MCS problem, the scores for a match, a mismatch, and a space are one, zero, and zero, respectively [36]. Figure 4.9(b) shows the dynamic programming table for computing the MCS of two sequences $X = \langle 3, 1, 4, 2\rangle$ *and* $Y = \langle 1, 3, 4, 2\rangle$*, derived from rows 1 and 2 of the input matrix A, respectively. We denote the entry in the i-th row and the j-th column by* $D[i, j]$*. We index the topmost row by* $i = 0$ *and use* $j = 0$ *to indicate the leftmost column. Let* x_i *and* y_j *denote the i-th and the j-th element of X and Y, respectively. Then, the optimal substructure of the MCS problem gives the following recursive formula [24]*

$$D[i,j] = \begin{cases} 0 & : \ \textit{if } i = 0 \textit{ or } j = 0, \\ D[i-1, j-1] + 1 & : \ \textit{if } i, j > 0 \textit{ and } x_i = y_j, \\ max(D[i-1, j], D[i, j-1]) & : \ \textit{if } i, j > 0 \textit{ and } x_i \neq y_j, \end{cases}$$

In addition, we place a traceback pointer ($\nwarrow, \uparrow, \longleftarrow$) *in every entry* $D[i, j]$ *for* $i > 0$ *and* $j > 0$*, indicating where the value in the entry* $D[i, j]$ *originated (i.e.,* $D[i-1, j-1]$*,* $D[i-1, j]$*, or* $D[i, j-1]$*). Each MCS corresponds to a traceback path from the largest element in the table, and this path is obtained by following the traceback pointers, which are indicated by the bold arrows in Figure 4.9(b). In this particular example, two MCS exist, namely,* $\langle 3, 4, 2\rangle$ *and* $\langle 1, 4, 2\rangle$*. More details on this procedure can be found in [24, 36].*

One possible improvement of Algorithm 4.2 would be to consider 'noisy ordering.' That is, we can devise an algorithm that can rearrange elements with similar values in such a way that a longer MCS can emerge. This heuristic will help to find atomic biclusters with more columns, from which larger Type 2 biclusters can potentially be derived.

4.2.3 Finding Type 3 atomic biclusters

Definition 4.9. *Given input matrix* $A = (R, C)$ *and threshold* $\tau \geq 0$*, a* Type 3 atomic bicluster *for rows* $i, k \in R$ $(i \neq k)$ *is a two-row matrix, denoted by pair*

Algorithm 4.3: Finding Type 3 atomic biclusters for pair of rows

input : $A = (R, C)$, a data matrix
input : $q, r \in R$, row indices ($q \neq r$)
input : τ, a threshold
output: $J \subseteq C$, $(\{q, r\}, J)$ is an atomic Type 3 bicluster

```
1  foreach j ∈ C do
2      D[j].val := a_qj − a_rj;
3      D[j].ind := j;
4  Sort array s in ascending order with respect to the field val;
5  begin := 1;
6  end := 2;
7  while (end ≤ |C|) do
8      if (D[end].val − D[begin].val ≤ τ) then
9          end := end + 1;
10         if (end > |C|) then
11             Report J = {D[begin].ind, ..., D[end − 1].ind};
12     else
13         Report J = {D[begin].ind, ..., D[end − 1].ind};
14         repeat
15             begin := begin + 1;
16         until (begin = end) or (D[end].val − D[begin].val ≤ τ);
```

Figure 4.10: Algorithm to find Type 3 atomic biclusters.

$P = (\{i, k\}, J)$, that satisfies the following: (1) *P is a Type 3 bicluster on A; and* (2) *there is no J' such that $J' \supset J$ and $(\{i, k\}, J')$ is also a Type 3 bicluster.*

Algorithm 4.3 details our approach to find Type 3 atomic biclusters defined in Definition 4.6. This algorithm is equivalent to Algorithm 4.1, except for Line 2. An informal explanation is as follows. Algorithm 4.1 is used to find a Type 1 atomic bicluster, or a one-row matrix in which the elements have similar values. Algorithm 4.3 is used to find a two-row matrix in which any 2×2 submatrix $\begin{bmatrix} e & f \\ g & h \end{bmatrix}$ has similar values of $(e - g)$ and $(f - h)$, since $|e - g - f + h| = |(e - g) - (f - h)| \leq \tau$. Thus, we can use Algorithm 4.1 to find Type 3 atomic biclusters, simply by subtracting the values in one row from the values in another and considering the result as a one-row matrix. This subtraction occurs in Line 2 of Algorithm 4.3. Some details helpful for understanding our informal proof can be found in [107].

	1	*2*	*3*	*4*	*5*
1	6.0	9.0	2.0	5.0	4.0
2	2.0	3.0	4.0	3.0	6.0
3	2.0	2.0	3.0	6.0	7.0
4	3.0	4.0	5.0	2.0	6.0
5	4.0	7.0	3.0	1.0	4.0

(a)

Row set $\{q, r\}$	Column set $J \in 2^C$
$\{1,2\}$	$\{3,5\}$
$\{1,3\}$	$\{3,4\}$
$\{1,4\}$	$\{1,4\},\{3,5\}$
$\{1,5\}$	$\{1,2\},\{3,5\}$
$\{2,3\}$	$\{1,5\},\{1,2,3\}$
$\{2,4\}$	$\{1,2,3,5\},\{4,5\}$
$\{2,5\}$	$\{3,4,5\}$
$\{3,4\}$	$\{1,2,3\}$
$\{3,5\}$	$\emptyset$
$\{4,5\}$	$\{3,4,5\}$

(b)

Figure 4.11: (a) Input matrix A. (b) Type 3 atomic biclusters found from A by Algorithm 4.3 ($\tau = 1$).

Example 4.10. *Figure 4.11(b) presents the Type 3 atomic biclusters found by Algorithm 4.3 from the data in Figure 4.4(a), repeated in Figure 4.11(a). The parameter used is* $\tau = 1$.

4.3 Our bicluster mining algorithm

4.3.1 Overview

We can formulate the bicluster mining problem in terms of a binary relation.

Definition 4.10. *Given* $A = (R, C)$*, an input data matrix, and* $\mathfrak{D}$*, a specific definition of a bicluster,* $\mathfrak{R}_{\mathfrak{D}}$ *is a binary relation on* $2^R \times 2^C$*:*

$$\mathfrak{R}_{\mathfrak{D}} = \{(I, J) | \textit{The pair } (I, J) \textit{ forms a bicluster appearing in } A \textit{ under the definition } \mathfrak{D}\}. \tag{4.1}$$

Under this definition, the objective of bicluster mining is to find the elements of the relation $\mathfrak{R}_{\mathfrak{D}}$. We aim at finding only maximal biclusters, as stated in Section 4.1.3.

Assume that we can find a function, denoted by $\mathfrak{J}$, that accepts as input $I \in 2^R$ and produces all maximal $J \in 2^C$ such that $(I, J) \in \mathfrak{R}_{\mathfrak{D}}$. Then, we may devise a naive algorithm that can provide all elements of $\mathfrak{R}_{\mathfrak{D}}$: First enumerate every $I \in 2^R$ and then feed it to the function $\mathfrak{J}$. Obviously, this approach is not feasible for a data matrix of non-trivial size since the powerset 2^R grows exponentially. Here we explain how to improve this idea of exploiting the function $\mathfrak{J}$ so that we can apply it to mining nested biclusters appearing in large-scale data matrices. Formally, the definition of $\mathfrak{J}$ is as follows.

Definition 4.11. *Given matrix $A = (R, C)$, $\mathfrak{J}$ is a function that maps $I \in 2^R$ to the image $\mathfrak{J}(I)$, where*

$$\mathfrak{J}(I) = \{J \in 2^C | (I, J) \in \mathfrak{R}_{\mathfrak{D}} \text{ and } \nexists J' \supset J \text{ s.t. } (I, J') \in \mathfrak{R}_{\mathfrak{D}}\}.$$

In Section 4.3.2, we first explain how to define the function $\mathfrak{J}$ using the atomic biclusters previously developed. In addition, we propose a ZBDD-based technique to implement the function efficiently. Section 4.3.3 then presents how to exploit the function $\mathfrak{J}$ to find nested biclusters, avoiding the exhaustive enumeration of $I \in 2^R$. We propose two algorithms: One uses a breadth-first approach and the other employs a depth-first approach. Finally, Section 4.3.4 provides remarks on algorithm complexity and other issues.

4.3.2 Representation and implementation of the function $\mathfrak{J}$

We first introduce the operator $\odot$, which is essentially equivalent to the operator $\otimes$ defined in Chapter 3 but does not contain redundant subsets.

Definition 4.12. *Let $\mathcal{T}$ and $\mathcal{U}$ be two sets of subsets. Also let $\mathcal{Q} = \{T \cap U | \forall T \in \mathcal{T}, \forall U \in \mathcal{U}\}$. Then, the binary operator $\odot$ on $\mathcal{T}$ and $\mathcal{U}$ is defined as follows:*

$$\mathcal{T} \odot \mathcal{U} = \mathcal{Q} - \{Q | \exists Q' \in \mathcal{Q} \text{ s.t. } Q' \supset Q\}. \tag{4.2}$$

Theorem 4.1. *Let $\mathcal{T}, \mathcal{U}, \mathcal{W}$ be sets of sets. Then, $(\mathcal{T} \odot \mathcal{U}) \odot \mathcal{W} = \mathcal{T} \odot (\mathcal{U} \odot \mathcal{W})$.*

Proof. We can prove the theorem by showing that (1) $(\mathcal{T} \odot \mathcal{U}) \odot \mathcal{W} \supseteq \mathcal{T} \odot (\mathcal{U} \odot \mathcal{W})$ and (2) $(\mathcal{T} \odot \mathcal{U}) \odot \mathcal{W} \subseteq \mathcal{T} \odot (\mathcal{U} \odot \mathcal{W})$. We first prove (1). For the sake of contradiction, assume that $(\mathcal{T} \odot \mathcal{U}) \odot \mathcal{W} \subset \mathcal{T} \odot (\mathcal{U} \odot \mathcal{W})$. This means that there exists a set S such that $S \in \mathcal{T} \odot (\mathcal{U} \odot \mathcal{W})$ and $S \notin (\mathcal{T} \odot \mathcal{U}) \odot \mathcal{W}$. Assume that $S = T \cap (U \cap W)$, where $T \in \mathcal{T}$, $U \in \mathcal{U}$, and $W \in \mathcal{W}$. Since $S \notin (\mathcal{T} \odot \mathcal{U}) \odot \mathcal{W}$, there must exist $W' \in \mathcal{W}$ such that $(T \cap U) \cap W \subset (T \cap U) \cap W'$. By the associative law for basic set intersection, $T \cap (U \cap W) = (T \cap U) \cap W \subset (T \cap U) \cap W' = T \cap (U \cap W')$. In other words, if $S \notin (\mathcal{T} \odot \mathcal{U}) \odot \mathcal{W}$, then there must exist $W' \in \mathcal{W}$ such that $(U \cap W) \subset (U \cap W')$. However, since $S \in \mathcal{T} \odot (\mathcal{U} \odot \mathcal{W})$, there cannot exist $W' \in \mathcal{W}$ such that $(U \cap W) \subset (U \cap W')$. We have reached a contradiction and thus our original assumption that $(\mathcal{T} \odot \mathcal{U}) \odot \mathcal{W} \subset \mathcal{T} \odot (\mathcal{U} \odot \mathcal{W})$ must be false. Therefore, $(\mathcal{T} \odot \mathcal{U}) \odot \mathcal{W} \supseteq \mathcal{T} \odot (\mathcal{U} \odot \mathcal{W})$. By symmetry, we can prove $(\mathcal{T} \odot \mathcal{U}) \odot \mathcal{W} \subseteq \mathcal{T} \odot (\mathcal{U} \odot \mathcal{W})$ in a similar way.

We have shown that $(\mathcal{T} \odot \mathcal{U}) \odot \mathcal{W} \supseteq \mathcal{T} \odot (\mathcal{U} \odot \mathcal{W})$ and $(\mathcal{T} \odot \mathcal{U}) \odot \mathcal{W} \subseteq \mathcal{T} \odot (\mathcal{U} \odot \mathcal{W})$, which completes the proof. □

The associative law thus holds for the operator $\odot$, and it is trivial to show that the commutative law, $\mathcal{T} \odot \mathcal{U} = \mathcal{U} \odot \mathcal{T}$, holds. Consequently, we can develop the following notation.

Definition 4.13. *The pairwise intersection of the k sets of sets $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k$ is denoted by*

$$\mathcal{T}_1 \odot \mathcal{T}_2 \odot \cdots \odot \mathcal{T}_k = \bigodot_{i=1}^{k} \mathcal{T}_i. \tag{4.3}$$

In addition, we define the operator $\textsc{cover}(S)$ for a set S in order to facilitate further explanation.

Definition 4.14. *Given a set $S = \{s_1, s_2, \ldots, s_k\}$ with $k \geq 2$, $\textsc{cover}(S)$ is a minimum edge cover of K_k, the complete graph with k vertices, in which the set of vertices corresponds to S.*

Example 4.11. $\{\{0,1,2\},\{2,3,4\}\} \odot \{\{0,2\},\{4,5\}\} = \{\{0,2\},\{4\}\}$. *Let* $S_1 = \{1,2,3,4\}$ *and* $S_2 = \{10,11,12\}$. *Then, a possible instance of* $\textsc{cover}(S_1) = \{\{1,3\}, \{2,4\}\}$, *and an example of* $\textsc{cover}(S_2) = \{\{10,11\},\{10,12\}\}$.

Re-defining $\mathfrak{J}$ in terms of atomic biclusters

The image $\mathfrak{J}(I)$ defined in Definition 4.11 can be re-defined using atomic biclusters by the following theorem.

Theorem 4.2. *Let $\mathfrak{J}_1$, $\mathfrak{J}_2$, and $\mathfrak{J}_3$ denote the function $\mathfrak{J}$ for Types 1, 2, and 3, respectively. Given input data $A = (R, C)$, the image of $I \in 2^R$, or $\mathfrak{J}(I)$, can be represented as follows.*

- *When the set I has only one or two elements:*

$$\mathfrak{J}_1(\{r\}) = \{J | (\{r\}, J) \text{ is a Type 1 atomic bicluster for } r \in R\} \quad (4.4)$$

$$\mathfrak{J}_2(\{q, r\}) = \{J | (\{q, r\}, J) \text{ is a Type 2 atomic bicluster for } q, r \in R\} \quad (4.5)$$

$$\mathfrak{J}_3(\{q, r\}) = \{J | (\{q, r\}, J) \text{ is a Type 3 atomic bicluster for } q, r \in R\} \quad (4.6)$$

- *Otherwise:*

$$\mathfrak{J}_1(I) = \bigodot_{\forall i \in I} \mathfrak{J}_1(\{i\}) \quad (4.7)$$

$$\mathfrak{J}_2(I) = \bigodot_{\forall I' \in \text{COVER}(I)} \mathfrak{J}_2(I') \quad (4.8)$$

$$\mathfrak{J}_3(I) = \bigodot_{\forall \{i,k\} \subseteq I} \mathfrak{J}_3(\{i, k\}) \quad (4.9)$$

To evaluate Equations 4.7,4.8, and 4.9, we need to invoke the operator $\odot$ at most $(|I| - 1)$, $(\lceil \frac{|I|}{2} \rceil - 1)$, and $\binom{|I|}{2}$ times, respectively.

Example 4.12. *To find the bicluster P_1 in Figure 4.4(b), we can use Theorem 4.2*

and the atomic biclusters presented in Figure 4.11(b) as follows:

$$\begin{aligned}\mathfrak{J}_3(\{1,2,4,5\}) &= \mathfrak{J}_3(\{1,2\}) \odot \mathfrak{J}_3(\{1,4\}) \odot \mathfrak{J}_3(\{1,5\}) \\ &\quad \odot \mathfrak{J}_3(\{2,4\}) \odot \mathfrak{J}_3(\{2,5\}) \odot \mathfrak{J}_3(\{4,5\}) \\ &= \{\{3,5\}\} \odot \{\{1,4\},\{3,5\}\} \odot \{\{1,2\},\{3,5\}\} \\ &\quad \odot \{\{1,2,3,5\},\{4,5\}\} \odot \{\{3,4,5\}\} \odot \{\{3,4,5\}\} \\ &= \{\{3,5\}\}.\end{aligned}$$

Proof of Theorem 4.2

The derivation of Equations 4.4, 4.5, and 4.6 is straightforward from the definition of atomic biclusters. If the set I has only one row (Type 1) or two (Types 2 and 3), the image $\mathfrak{J}(I)$ simply consists of the column set of atomic biclusters for the row(s) in I. Equations 4.7 and 4.8 can be derived from the generalization of Properties 4.1 and 4.2, respectively, by replacing the operator $\cap$ with the operator $\odot$ defined in Section 4.3.2.

Here we focus on the derivation of Equation 4.9. To this end, we first propose the following lemma.

Lemma 4.1. *Let (I, J) be a nested bicluster. If $\{i, k\} \subseteq I$, then there exists at least one set $J' \in \mathfrak{J}_3(\{i, k\})$ such that $J \subseteq J'$.*

Proof. Assume $J \supset J'$ for all $J' \in \mathfrak{J}_3(\{i, k\})$. Since (I, J) is a nested bicluster and $I \supseteq \{i, k\}$, its sub-bicluster $(\{i, k\}, J)$ is also a nested bicluster under the same definition. By definition, if $J' \in \mathfrak{J}_3(\{i, k\})$, then there exists no $J'' \supset J'$ such that $(\{i, k\}, J'')$ is yet another nested bicluster under the same definition. We have reached a contradiction and thus our original assumption that $J \supset J'$ for all $J' \in \mathfrak{J}_3(\{i, k\})$ must be false. Therefore, there must be at least one instance of $J' \in \mathfrak{J}_3(\{i, k\})$ such that $J \subseteq J'$. □

Now we derive Equation 4.9. Let $P = (I, J)$ be a maximal nested bicluster. Then, by Lemma 4.1, for each $\{i, k\} \subseteq I$, there exists *at least one* set $J_{\{i,k\}} \in \mathfrak{J}_3(\{i, k\})$ such

that $J \subseteq J_{\{i,k\}}$. For the sake of explanation, assume for now that *only one* such $J_{\{i,k\}}$ is contained in each $\mathfrak{J}_3(\{i,k\})$. Then, it follows that

$$J \subseteq \bigcap_{\forall\{i,k\}\subseteq I} J_{\{i,k\}}. \tag{4.10}$$

Moreover, since the bicluster P is maximal, there is no J' such that $J' \supset J$ and $J' \subseteq \bigcap_{\forall\{i,k\}\subseteq I} J_{\{i,k\}}$. Thus, the following equation holds for J:

$$J = \bigcap_{\forall\{i,k\}\subseteq I} J_{\{i,k\}}. \tag{4.11}$$

In general, each $\mathfrak{J}_3(\{i,k\})$ can have multiple instances of $J_{\{i,k\}}$, not only one as previously assumed. Thus, we can have multiple instances of Equation 4.11, which can be compactly represented using the operator $\odot$ defined in Section 4.3.2:

$$J \in \bigodot_{\forall\{i,k\}\subseteq I} \left\{ J_{\{i,k\}} | J_{\{i,k\}} \in \mathfrak{J}_3(\{i,k\}), J \subseteq J_{\{i,k\}} \right\}. \tag{4.12}$$

Finally, suppose that we replace the operands of $\odot$ in Relation 4.12 with $\{J_{\{i,k\}}|J_{\{i,k\}} \in \mathfrak{J}_3(\{i,k\})\} = \mathfrak{J}_3(\{i,k\})$, removing the constraint on J. Then, we can find not only the set J but also the other column sets that can form a nested bicluster with the row set I:

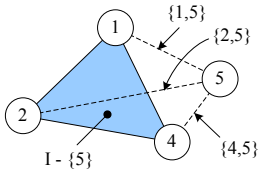
$$\{\text{all column sets that can form a Type 3 bicluster with } I\} = \bigodot_{\forall\{i,k\}\subseteq I} \mathfrak{J}_3(\{i,k\}). \tag{4.13}$$

By definition, the operator $\odot$ gives only maximal sets. Therefore, Equation 4.13 is equivalent to Equation 4.9. We have derived Equation 4.9, and this completes the proof of Theorem 4.2.

Enhancement by dynamic programming

We can reduce the number of the $\odot$ operations required to evaluate the equations in Theorem 4.2 by storing and re-using intermediate results. This idea is similar to

(a) Example 4.12: Theorem 4.2 (b) Example 4.13: Corollary 4.1



(c) Example 4.14: Corollary 4.2

Figure 4.12: Decomposition of the complete graph K_4.

the concept of dynamic programming. In the equations in Theorem 4.2, we can see that the *optimal substructure* [24] appears, which is a hallmark of the applicability of dynamic programming.

For example, the process of realizing $\mathfrak{J}_3$ can be compared to that of decomposing a complete graph into its cliques. We start our explanation with revisiting Example 4.12. Let K_k denote the complete graph with k graph vertices. Suppose that we have the graph K_4, in which the vertices represent the elements of $I = \{1, 2, 4, 5\}$ as shown in Figure 4.12(a). In this figure, we can decompose the graph K_4 into $\binom{4}{2} = 6$ different K_2. This decomposition corresponds to evaluating Equation 4.9. We thus evaluated $\mathfrak{J}_3(\{1, 2, 4, 5\})$ using $\mathfrak{J}_3(\{1, 2\}), \mathfrak{J}_3(\{1, 4\}), \ldots, \mathfrak{J}_3(\{4, 5\})$ in Example 4.12.

Alternatively, we can decompose K_4 into two different K_3 and one K_2 as shown in Figure 4.12(b). The shaded triangle represents the set $I - \{5\} = \{1, 2, 4\}$ and the triangle indicated by bold lines represents $I - \{4\} = \{1, 2, 5\}$. This suggests a different way of evaluating $\mathfrak{J}_3(\{1, 2, 4, 5\})$, namely, the evaluation using $\mathfrak{J}_3(\{1, 2, 4\}), \mathfrak{J}_3(\{1, 2, 5\})$, and $\mathfrak{J}_3(\{4, 5\})$.

The alternative decomposition of $\mathfrak{J}_1$ and $\mathfrak{J}_2$ corresponding to Figure 4.12(b) is simpler. Since $I = (I - \{4\}) \cup (I - \{5\})$, $\mathfrak{J}_t(I)$ is merely $\mathfrak{J}_t(I - \{4\}) \odot \mathfrak{J}_t(I - \{5\})$, for each $t \in \{1, 2\}$.

Corollary 4.1. *Given input data $A = (R, C)$, let set $I \in 2^R$ and suppose that $i, k \in I$*

and $i \neq k$. Then, the image $\mathfrak{J}_t(I)$ for each type $t \in \{1,2,3\}$ can be represented as follows:

$$\mathfrak{J}_1(I) = \mathfrak{J}_1(I-\{i\}) \odot \mathfrak{J}_1(I-\{k\}) \quad (4.14)$$

$$\mathfrak{J}_2(I) = \mathfrak{J}_2(I-\{i\}) \odot \mathfrak{J}_2(I-\{k\}) \quad (4.15)$$

$$\mathfrak{J}_3(I) = \mathfrak{J}_3(I-\{i\}) \odot \mathfrak{J}_3(I-\{k\}) \odot \mathfrak{J}_3(\{i,k\}) \quad (4.16)$$

When applying Corollary 4.1, we need to call the operator $\odot$ only once (Types 1 and 2) or twice (Type 3), as long as the intermediate results $\mathfrak{J}(I-\{i\})$ and $\mathfrak{J}(I-\{k\})$ are available. In Section 4.3.3, we explain how to store and re-use intermediate results efficiently using a breadth-first search algorithm.

Example 4.13. *We can apply Corollary 4.1 to the previous example as follows:*

$$\begin{aligned}\mathfrak{J}_3(\{1,2,4,5\}) &= \mathfrak{J}_3(\{1,2,4\}) \odot \mathfrak{J}_3(\{1,2,5\}) \odot \mathfrak{J}_3(\{4,5\}) \\ &= \{\{3,5\}\} \odot \{\{3,5\}\} \odot \{\{3,4,5\}\} \\ &= \{\{3,5\}\}.\end{aligned}$$

Figure 4.12(c) shows another method to decompose the graph K_4 for Type 3 biclusters. Here K_4 is decomposed into one K_3 and three different K_2. The shaded triangle represents the set $I-\{5\} = \{1,2,4\}$ and the dotted lines the sets $\{1,5\}$, $\{2,5\}$, and $\{4,5\}$. This suggests a different way of evaluating $\mathfrak{J}_3(\{1,2,4,5\})$ using $\mathfrak{J}_3(\{1,2,4\}), \mathfrak{J}_3(\{1,5\})$, $\mathfrak{J}_3(\{2,5\})$, and $\mathfrak{J}_3(\{4,5\})$. The decomposition of $\mathfrak{J}_1$ and $\mathfrak{J}_2$ corresponding to Figure 4.12(c) remains in essence the same as the previous case.

Corollary 4.2. *Given input data $A = (R,C)$, let set $I \in 2^R$ and suppose that $k,l \in I$ and $k \neq l$. Then, the image $\mathfrak{J}_t(I)$ for each type $t \in \{1,2,3\}$ can be represented as*

follows:

$$\mathfrak{J}_1(I) = \mathfrak{J}_1(I-\{k\}) \odot \mathfrak{J}_1(\{k\}) \tag{4.17}$$

$$\mathfrak{J}_2(I) = \mathfrak{J}_2(I-\{k\}) \odot \mathfrak{J}_2(\{k,l\}) \tag{4.18}$$

$$\mathfrak{J}_3(I) = \mathfrak{J}_3(I-\{k\}) \odot \left\{ \bigodot_{\forall i \in I, i \neq k} \mathfrak{J}_3(\{i,k\}) \right\} \tag{4.19}$$

In order to apply Corollary 4.2, we need to execute the operator $\odot$ twice (Types 1 and 2) or at most $(|I|-1)$ times (Type 3), as long as the result of $\mathfrak{J}(I-\{k\})$ is available. The number of $\odot$ operations involved in the computation of $\mathfrak{J}_3$ in Corollary 4.2 is thus more than that in Corollary 4.1. However, it is easier to manage the partial results in Corollary 4.2, thus compensating for the larger number of $\odot$ operations required. Section 4.3.3 presents a depth-first search algorithm, which exploits Corollary 4.2 to evaluate $\mathfrak{J}$ efficiently.

Example 4.14. *We can apply Corollary 4.2 to Example 4.12 as follows:*

$$\begin{aligned}
\mathfrak{J}_3(\{1,2,4,5\}) &= \mathfrak{J}_3(\{1,2,4\}) \odot \mathfrak{J}_3(\{1,5\}) \odot \mathfrak{J}_3(\{2,5\}) \odot \mathfrak{J}_3(\{4,5\}) \\
&= \{\{3,5\}\} \odot \{\{1,2\},\{3,5\}\} \odot \{\{3,4,5\}\} \odot \{\{3,4,5\}\} \\
&= \{\{3,5\}\}.
\end{aligned}$$

Efficient implementation of the operator $\odot$ using ZBDDs

In order to use ZBDDs to implement the operator $\odot$, we first need to represent the operands of $\odot$ by ZBDDs. The operand of $\odot$ is a set of column sets, $\mathfrak{J}(I)$, and each column set $J \in \mathfrak{J}(I)$ can easily be converted to a combination (see Section 2.4) as follows. Given input data $A = (R, C)$, assume $C = \{1, 2, \ldots, m\}$. Then, the set J corresponds to an m-bit vector $\langle b_1, b_2, \ldots, b_m \rangle$, where $b_i = 1$ if $i \in J$, and $b_i = 0$ otherwise. Representing this m-bit vector by ZBDDs is a standard procedure and is thus beyond the scope of this chapter. Further details can be found in Sections 2.4 and 3.4.2 as well as in [67, 69, 70].

Example 4.15. *In Figure 4.11(b),* $\mathfrak{J}_3(\{2,5\}) = \{\{3,4,5\}\}$. *The set* $\{3,4,5\}$ *can be*

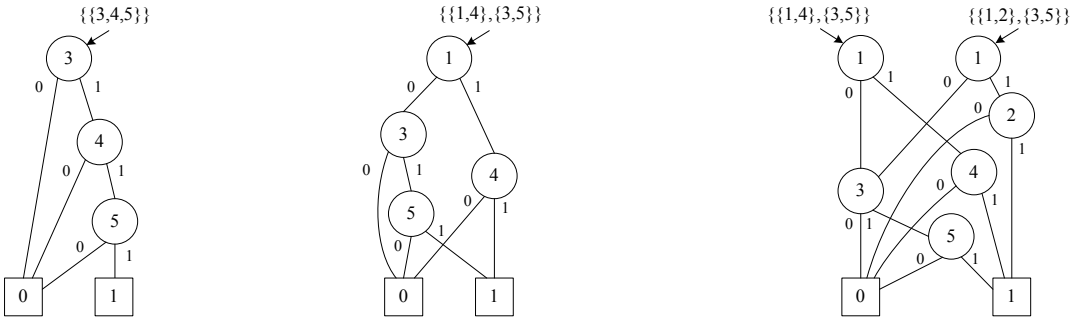


Figure 4.13: ZBDD representation of atomic biclusters. (a) $\mathfrak{J}_3(\{2,5\}) = \{\{3,4,5\}\}$. (b) $\mathfrak{J}_3(\{1,4\}) = \{\{1,4\},\{3,5\}\}$. (c) $\mathfrak{J}_3(\{1,4\}) = \{\{1,4\},\{3,5\}\}$ and $\mathfrak{J}_3(\{1,5\}) = \{\{1,2\},\{3,5\}\}$.

converted to 5-bit vector (00111) *and represented by the ZBDD in Figure 4.13(a). In the same example,* $\mathfrak{J}_3(\{4,5\}) = \mathfrak{J}_3(\{2,5\})$*. Thus,* $\mathfrak{J}_3(\{4,5\})$ *can be represented by the identical ZBDD for* $\mathfrak{J}_3(\{2,5\})$ *without creating a new one.*

Example 4.16. *In Figure 4.11(b),* $\mathfrak{J}_3(\{1,4\}) = \{\{1,4\},\{3,5\}\}$*. This corresponds to the set of combinations* $\{10010, 00101\}$ *and can be represented by the ZBDD in Figure 4.13(b). Also,* $\mathfrak{J}_3(\{1,5\}) = \{\{1,2\},\{3,5\}\}$ *can share the part of the ZBDD for* $\mathfrak{J}_3(\{1,4\})$*, as shown in Figure 4.13(c).*

Next, we implement the operator $\odot$ by directly manipulating the ZBDDs representing the operands. This allows us to avoid explicit enumeration of the intermediate results, thus providing a large speed-up over the conventional methods to represent and manipulate sets [69, 70]. As is often the case with the operators defined on ZBDDs, we define the operator $\odot$ recursively. The implementation of $\odot$ is essentially equivalent to that of $\otimes$, and more details can be found in Section 3.4.2.

4.3.3 Finding nested biclusters

We present two methods to find nested biclusters. Both methods utilize the function $\mathfrak{J}$ previously developed. Before providing the details of these methods in Sections 4.3.3 and 4.3.3, we present an example to explain the fundamental ideas common in both methods.

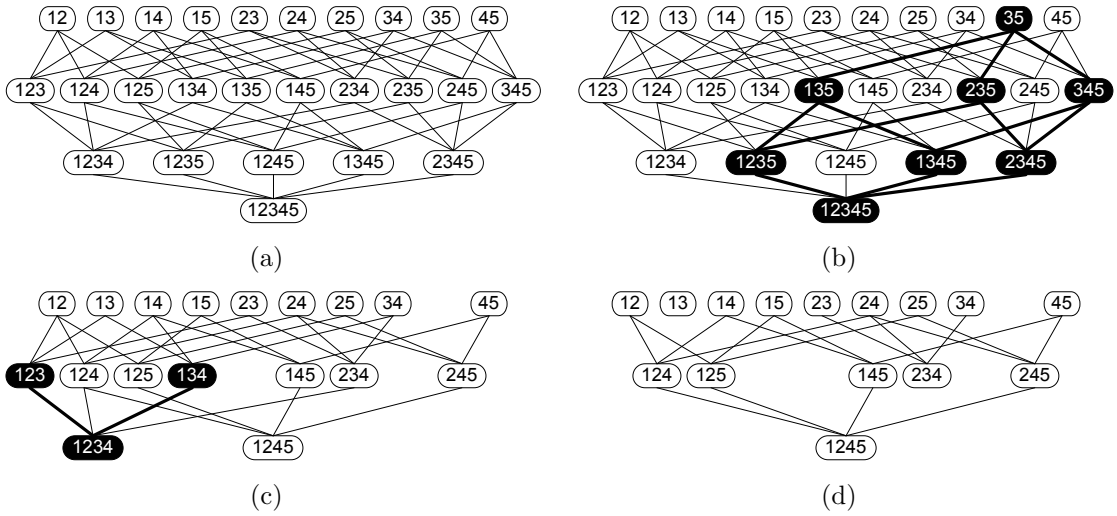


Figure 4.14: The process to find the biclusters presented in Figure 4.4(b). This is only for explanation of the idea, and in practice, we do not need the graph in its entirety all the time. Refer to Algorithms 4.4 and 4.5 for more details.

An example of finding Type 3 biclusters

Figure 4.14 shows the process to find the biclusters in Figure 4.4(b) from the data matrix in Figure 4.4(a), in which the set of rows $R = \{1, 2, 3, 4, 5\}$. In the graphs shown in the figure, each vertex v has two associated fields, namely $v.I$ and $v.\mathcal{J}$. The field $v.I$ is to save a set of rows, and the field $v.\mathcal{J}$ is to store the image $\mathfrak{J}(v.I)$. The *level* of the vertex v is defined as the cardinality of $v.I$. Also, we connect vertex v_1 at level l and vertex v_2 at level $l + 1$ by an edge if $v_1.I \subset v_2.I$.

Figure 4.14(a) presents a graph in which each vertex represents an elements in 2^R and a vertex is connected to others by the above rule. For example, $v.I = \{1, 2\}$ for the vertex v indicated by "12". This vertex is connected to the vertices indicated by "123", "124", and "125".

We can make two key observations in the graph constructed as above. First, not all vertices need to be examined. Thus, we can avoid exhaustive enumeration of $I \in 2^R$. Second, the intermediate results required to apply Corollaries 4.1 and 4.2 are available from the vertices at the previous level.

The first observation is based upon the following fact: If $\mathfrak{J}(I) = \emptyset$, then $\mathfrak{J}(I') = \emptyset$

for all $I' \supseteq I$. This is because if the pair (I, J) does not represent a nested bicluster, then the pair (I', J) with $I' \supseteq I$ cannot be a nested bicluster either. For example, in Figure 4.11(b) we know that $\mathfrak{J}_3(\{3,5\}) = \emptyset$. Thus, it is possible to conclude that $\mathfrak{J}_3(I) = \emptyset$ for all $I \supseteq \{3,5\}$. Consequently, we can eliminate any vertex v such that $v.I \supseteq \{3,5\}$. In the graph in Figure 4.14(b), the vertices to be deleted are indicated.

The example in Figure 4.14(c) shows that another vertex elimination process is possible, starting from the vertices at level 3, namely, "123" and "134". The vertex "123" should be deleted because $\mathfrak{J}_3(\{1,2,3\}) = \mathfrak{J}_3(\{1,2\}) \odot \mathfrak{J}_3(\{1,3\}) \odot \mathfrak{J}_3(\{2,3\}) = \emptyset$. We can remove the vertex "134" similarly. Thus, any vertex v such that $v.I \supseteq \{1,2,3\}$ or $v.I \supseteq \{1,3,4\}$ can be deleted. Finally, the graph in Figure 4.14(d) shows the vertices that remain undeleted. It is these vertices to which we apply the function $\mathfrak{J}$ to find nested biclusters. Since the remaining vertices correspond to all $I \in 2^R$ that can potentially be the row set of a nested bicluster, applying the function $\mathfrak{J}$ to these vertices enables us to find all the nested biclusters that satisfy the input parameters specified.

To exploit the intermediate results stored in the vertices at the previous level, the breadth-first algorithm in Section 4.3.3 starts with the vertices at level 2 and proceed to level $l+1$ from level l only after no vertex at level l is left. This is compatible with the decomposition of $\mathfrak{J}$ in Corollary 4.1. In contrast, the depth-first algorithm in Section 4.3.3 starts with vertex v at level 2 and proceeds until the algorithm examines all the vertices whose I set contains $v.I$. Then the algorithm starts with another vertex at level 2. This algorithm fits with the decomposition of $\mathfrak{J}$ in Corollary 4.2. Both algorithms find the same nested biclusters, although one can be faster than the other, depending upon the specific input data matrix and parameters used.

One important comment is in order. Obviously, it is not realistic to construct the graph like the one in Figure 4.14(a) in its entirety, especially when the set R has many elements. The examples in Figure 4.14 are only for explanation. As will be described in Algorithms 4.4 and 4.5, the breadth-first and the depth-first algorithms do not need to examine all the vertices simultaneously.

Algorithm 4.4: Breadth-first bicluster mining algorithm

input : $A = (R, C)$, a data matrix
input : $type \in \{1, 2, 3\}$, bicluster type
input : parameters for atomic bicluster generation
output: nested biclusters

```
1  Generate atomic biclusters (see Section 4.2);
2  foreach {q, r} ⊆ R do
3      if type = 1 then
4          J := ℑ1({q}) ⊙ ℑ1({r});
5      else
6          J := ℑ_type({q, r});
7      if J ≠ ∅ then
8          Create vertex v;
9          v.I := {q, r};
10         v.J := J;
11         v.level := 2;
12 maxLevel := |R|;
13 for l = 2 to maxLevel do
14     for i = 1 to numVertices in level l do
15         v_i := i-th vertex in level l;
16         foreach J ∈ v_i.J do
17             Collect bicluster (v_i.I, J);
18         if l < maxLevel then
19             for j = i + 1 to numVertices in level l do
20                 v_j := j-th vertex in level l;
21                 I := v_i.I ∪ v_j.I;
22                 if |I| ≠ l + 1 then next;
23                 if a vertex for I exists then next;
24                 if type = 3 then
25                     e_i := the element in I − v_i.I;
26                     e_j := the element in I − v_j.I;
27                     J := v_i.J ⊙ v_j.J ⊙ ℑ3({e_i, e_j});
28                 else
29                     J := v_i.J ⊙ v_j.J;
30                 if J ≠ ∅ then
31                     Create vertex v;
32                     v.I := I;
33                     v.J := J;
34                     v.level := l + 1;
35         Remove v_i;
36 Remove redundancy and return remaining biclusters;
```

Figure 4.15: Breadth-first algorithm

Breadth-first algorithm

Algorithm 4.4 details our breadth-first approach to find nested biclusters. The input is a data matrix, bicluster type, and parameters for atomic bicluster generation. The output are nested biclusters found from the input data matrix.

In **Line 1**, atomic biclusters are generated by the algorithms explained in Section 4.2 with the input parameters.

In **Lines 2–11**, the base vertices at level 2 are generated. Each vertex v has three associated data fields. The fields $v.I$ and $v.\mathcal{J}$ are the same as explained in the previous section. The field $v.level$ is to store the level of the vertex v. For Type 2 or Type 3 biclusters, a new vertex is created for each pair of rows, unless no atomic bicluster exists for the pair. The base vertices for Type 1 biclusters also start at level 2 by merging two atomic biclusters.

In **Lines 13–34**, the algorithm iterates for each level and performs the following for each vertex at level l. **In Lines 16–17**, the algorithm reports any candidate biclusters obtained from the previous iteration. In **Lines 18–34**, new vertices appearing at level $l+1$ are generated. To this end, the algorithm examines two vertices v_i and v_j at level l. **Lines 21–22** are to test if the two vertices are qualified to create a new vertex at level $l+1$. As long as the sets $v_i.I$ and $v_j.I$ have the same elements but one, the vertices v_i and v_j can create a new vertex at the next level. Since a vertex at level $l+1$ should have only one more row than a vertex at level l, if the union of $v_i.I$ and $v_j.I$ has more than $l+1$ elements, the two vertices v_i and v_j cannot spawn a new vertex in the next level. For example, if $v_i.I = \{1,2,3\}$ and $v_j.I = \{1,2,4\}$ then these two vertex can create a new vertex, v, at level 4 with $v.I = \{1,2,3,4\}$. In contrast, if $v_i.I = \{1,2,3\}$ and $v_j.I = \{1,4,5\}$, then they cannot generate a new vertex at level 4, because the row sets differ by two elements. This way of creating new vertices avoids exhaustive enumeration. In **Line 23**, if the two vertices v_i and v_j are eligible for creating a new vertex, the algorithm sees whether the corresponding vertex already exists or not. If not, the algorithm computes the set $\mathcal{J}$ for this new vertex by Corollary 4.1 in **Lines 24–29**. In **Lines 30–34**, the new vertex v is actually created and stored for further reference in the next iteration, if the set $\mathcal{J}$ is not empty. Otherwise, no new vertex is created. This corresponds to removing all the

downstream of the vertex v in which $v.\mathcal{J} = \emptyset$ (e.g., Figure 4.14(b) and 4.14(c)).

In **Line 35**, a vertex is deleted as soon as it becomes of no use. Thus, the algorithm can keep at most two levels of the vertices at a time, rather than the entire graph.

In **Line 36**, any redundant biclusters are removed and the remaining biclusters are returned.

Depth-first algorithm

In Section 4.3.3, we explained our breadth-first approach. In this section, we introduce an alternative bicluster mining algorithm using a depth-first approach. We start the description with the examples in Figure 4.14 and 4.16. In order to visit the vertices in the depth-first sense, we need to restructure the graph. In particular, we remove some edges from the graph in Figure 4.14(a) so that the graph becomes the trie in Figure 4.16(a). A *trie* [2] is a special structure for representing sets of words. Here we regard the set $I \in 2^R$ as a word assuming a total order among the elements in R. For instance, we can assume the total order $1 \prec 2 \prec 3 \prec 4 \prec 5$ for the set $R = \{1, 2, 3, 4, 5\}$. Hence, the set $I = \{1, 2, 3\}$ corresponds to the word "123". This word is inserted in to the trie as the descendant of the word "12" and as the parent of the words "1234" and "1235", as shown in Figure 4.16(a).

Algorithm 4.4 provides the details of our depth-first approach. In **Lines 2–6**, the algorithm traverse the trie in preorder. More precisely, our algorithm *constructs* the trie in preorder rather than *traverses* it. In other words, the algorithm creates vertices whenever necessary and deletes them afterwards rather than keeping the trie in its entirety all the time. In **Line 7**, the algorithm reports the nested biclusters produced after removing redundant biclusters, if any.

For each vertex v encountered in this *preorder* construction of the trie, the algorithm performs the following (**Lines 8–36**). The algorithm computes $v.\mathcal{J}$ by Corollary 4.2 in **Line 18–24**. If the set $v.\mathcal{J}$ is empty, the algorithm does not proceed to examining the descendant vertices and returns to the parent vertex (**Line 25**). This is equivalent to deleting all the descendant vertices in Figure 4.16(b). If the set $v.\mathcal{J}$ is not empty, the algorithm produces nested biclusters $(v.I, J)$ for all $J \in \mathfrak{J}(v.I)$ in **Lines 26–27**. Then the algorithm creates a list of the descendant vertices in **Lines**

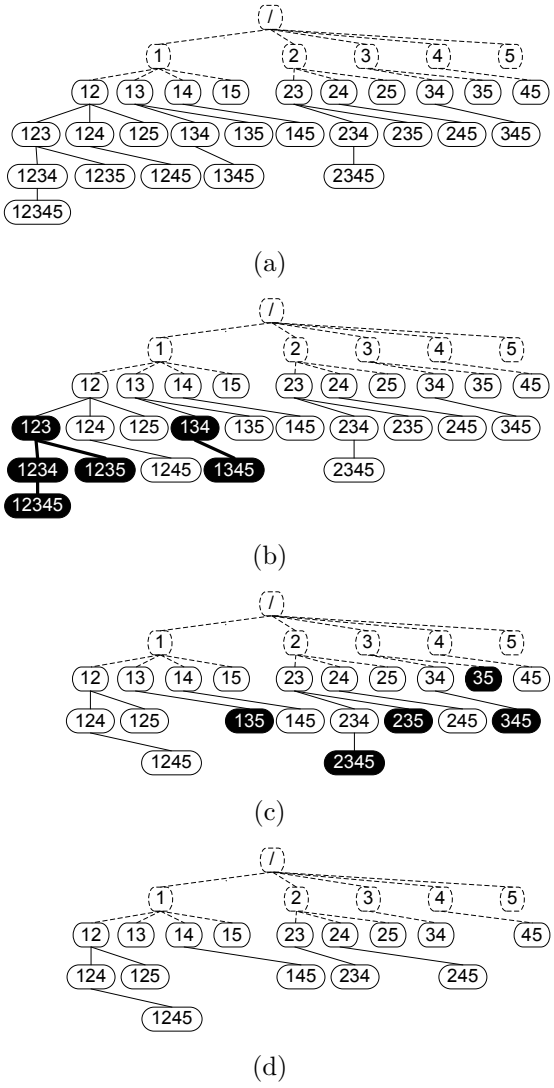


Figure 4.16: An example to explain the depth-first bicluster mining algorithm.

Algorithm 4.5: Depth-first Bicluster Mining Algorithm

input : $A = (R, C)$, a data matrix
input : $type \in \{1, 2, 3\}$, bicluster type
input : parameters for atomic bicluster generation
output: nested biclusters

```
1  Generate atomic biclusters (see Section4.2);
2  foreach {q, r} ⊆ R do
3      Create vertex v;
4      v.I := {q, r};
5      ConstructTrieInPreorder(v);
6      delete v;
7  Remove redundancy and return remaining biclusters;

8  procedure ConstructTrieInPreorder (vertex v)
9  begin
10     if |v.I| = 2 then
11         if type = 1 then
12             v.J := J_1({q}) ⊙ J_1({r});
13         else
14             v.J := J_type(v.I);
15     else
16         vertex p := v.parent;
17         k := the element in v.I − p.I;
18         if type = 1 then
19             v.J := p.J ⊙ J_1({k});
20         else if type = 2 then
21             k' := any element in p.I;
22             v.J := p.J ⊙ J_2({k, k'});
23         else
24             v.J := p.J ⊙ (⊙_{∀i∈p.I} J_3({i, k}));
25     if v.J = ∅ then return;
26     foreach J in v.J do
27         Collect bicluster (v.I, J);
28     l := the "largest" element in v.I wrt a total order ≺;
29     I := {i | i ∈ R and l ≺ i};
30     foreach i ∈ I do
31         create vertex w;
32         w.I := v.I ∪ {i};
33         w.parent := v;
34         ConstructTrieInPreorder(w);
35         delete w;
36 end
```

Figure 4.17: Depth-first algorithm

28–29. This step is necessary because the algorithm does not keep the entire trie all the time, and thus the vertex v is not already connected to its children. The "largest" element in Line 28 means the "largest" element in the total order we are assuming among the elements of R. For example, the largest element of the set $\{1, 2, 4\}$ is the element "4", assuming $1 \prec 2 \prec 4$. In **Lines 30–35**, the algorithm creates the descendant vertices and visits them to repeat the steps performed in Lines 8–36.

4.3.4 Remarks

The bicluster mining problem addressed in this dissertation is related to the problem of finding the maximum edge biclique in a bipartite graph, a problem known to be NP-complete [65, 76]. Although the worst-case complexity of Algorithms 4.4 and 4.5 is exponential in the number of rows in the input data matrix, the execution time on typical benchmarks is practical, as will be shown in subsequent chapters. This is due to the efficient techniques such as the ZBDD-based symbolic manipulations and the dynamic programming approach, which enable us to avoid the exhaustive and explicit enumeration of the intermediate results. In particular, the role of the ZBDDs is crucial in this study. Without using ZBDDs, it would not be possible to achieve the efficiency that the current implementation of our algorithm shows.

The bicluster mining algorithms discussed so far are exact in the sense that they can find all the biclusters that satisfy specific input parameters. If desired, it is possible to employ a heuristic algorithm that runs quickly but can find only a subset of the possible biclusters. For example, we can implement a "greedy" $\odot$ operator that reports only k largest (in terms of cardinality) sets, which will make the cardinality of $\mathcal{J}$ decrease. We can also utilize a measure of overlap such as Jaccard's coefficient [66] to avoid generating "similar-looking" atomic biclusters, thus reducing the number of atomic biclusters considered in later steps.

4.4 Summary

This chapter has proposed an effective computational method that can be useful for a variety of data mining applications. Given a data matrix, the proposed method can find biclusters appearing as a submatrix of the data matrix. In particular, we introduced the notion of nested biclusters and formulated the problem of finding three types of nested biclusters frequently encountered in the literature. We also mathematically characterized the problem and developed a novel method applicable to large-scale biological data. The proposed method employs dynamic programming as well as efficient data structures such as zero-suppressed decision diagrams (ZBDDs), which were particularly useful in extending the scalability of our method. Consequently, given a data matrix of practical scale, our approach can find with great efficiency all the nested biclusters that satisfy specific input parameters. The application of this method to several genomic data mining problems will follow in subsequent chapters.

Chapter 5

DNA Microarray Data Analysis

Chapters 3 and 4 described a ZBDD-based biclustering technique. This computational method was tested with large-scale genomic data sets, and the experimental results are presented in the remaining chapters. This chapter shows the details of analyzing gene expression data sets obtained from DNA microarray [27,62] experiments. DNA microarray technology allows us to monitor transcription levels of thousands of genes simultaneously and helps us to annotate gene functions, reconstruct gene regulatory networks, diagnose disease conditions, and characterize effects of medical treatments.

Section 5.1 explains the data preparation and evaluation criteria, and Section 5.2 presents the experimental results.

5.1 Experiment design

5.1.1 Data preparation

To verify the correctness of the proposed method, it was first tested with synthetic data sets that contain pre-defined embedded biclusters. Then, actual gene expression data sets were used for performance evaluation.

Synthetic data

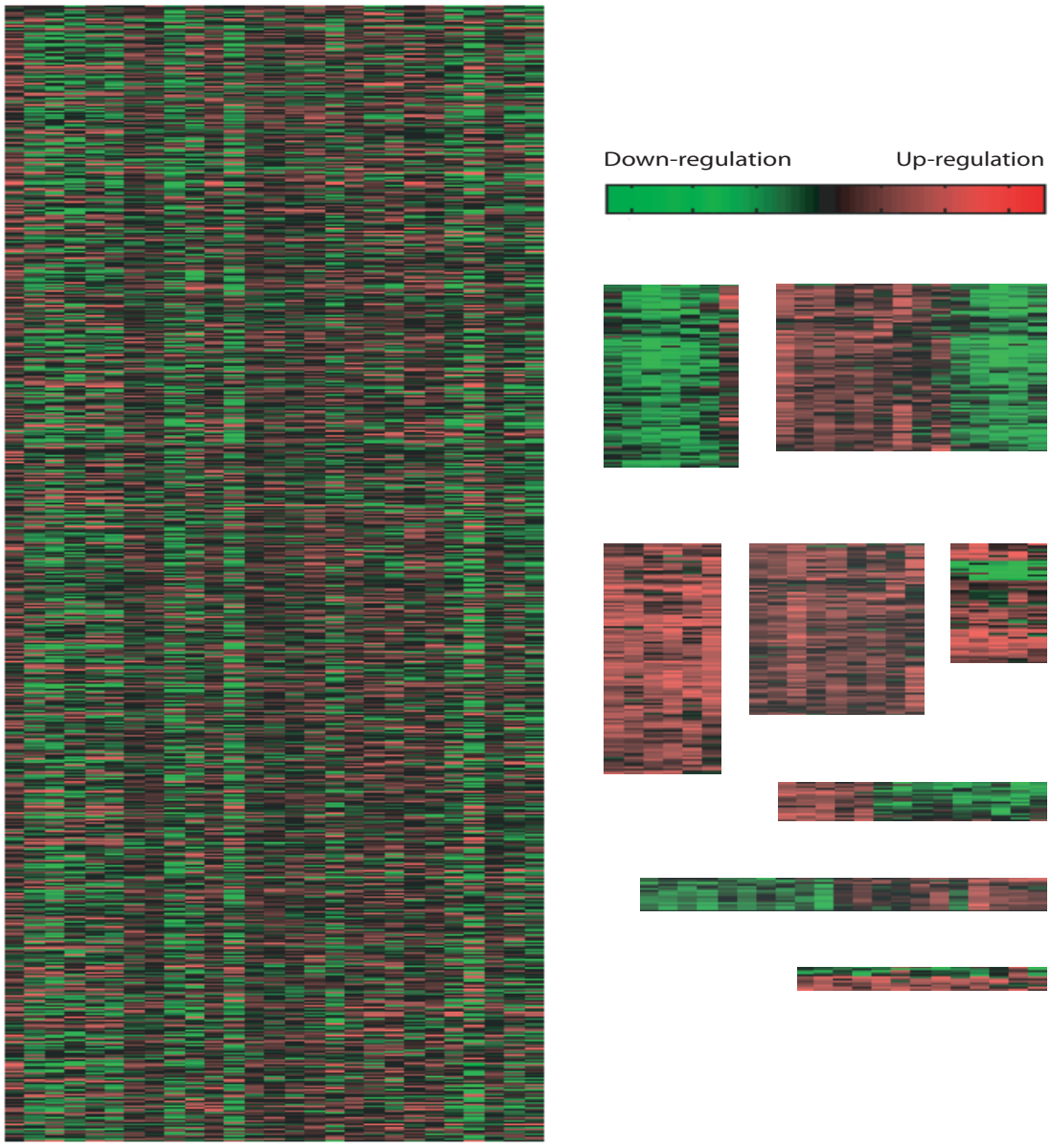
The synthetic data were prepared as follows. I first created null matrices of 100 rows and 5 different numbers of columns (1K, 3K, 6K, 9K and 12K). I then replaced the elements of each matrix with random numbers ranging from 0 to 500. For the matrix of $n = 100$ rows and $m \in \{1\text{K}, 3\text{K}, 6\text{K}, 9\text{K}, 12\text{K}\}$ columns, we embedded $0.05m$ pre-defined biclusters that have at least $0.1n$ rows and at least $0.01m$ columns. Each pre-defined bicluster was created in such a way that the values in every row or column fluctuate in harmony[1] and that all the methods involved in the experiment can detect the bicluster.

Biological data

Three different gene expression data sets were prepared for experiments. One was the yeast *Saccharomyces cerevisiae* cell cycle expression data [22, 99] produced by Affymetrix gene chip experiments. This data set contains the expression information of 2,884 genes under 17 experimental conditions. The second data set was the cDNA microarray data for renal cell carcinoma [42], which represents the expression levels of 1,876 genes under 27 different experimental conditions. The third was the B-cell lymphoma data set [4], which contains normal and cancerous cell-line samples from 119 patients for 4,026 genes.

Usually, gene expression data is arranged in a data matrix, in which each row corresponds to one gene and each column to one experimental condition. For more information on gene expression data, we refer the interested to [52]. For example, Figure 5.1 shows the heat map of the renal cell carcinoma data set as well as some biclusters found by our method.

[1]Every row or column is a shifted version of each other; examples are shown in Figure 5.2(a) and 5.2(b).



(a) Renal cell carcinoma data [42] (b) Example biclusters

Figure 5.1: The heat map of the renal cell carcinoma data [42] and some biclusters found by our method. The legend for the heat map is also presented in the upper right corner. The red color indicates up-regulation whereas the green color represents down-regulation. The black color means no change in regulation level. (a) The entire data matrix with 1,876 rows (genes) and 27 columns (experimental conditions). (b) Some biclusters (submatrices) discovered by our method.

5.1.2 Evaluation criteria

Basic performance indicators

The performance of the algorithm was evaluated in terms of response time, the number of biclusters discovered, and the input data size that can be handled.

Coherence in terms of MSR scores

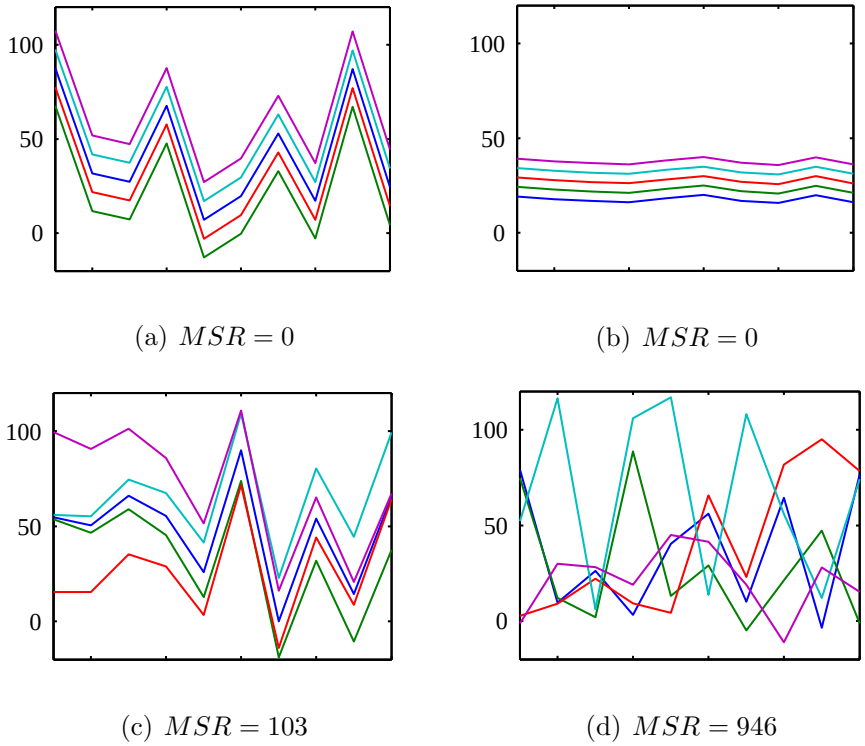
As mentioned in Section 2.3.3, the *mean squared residue* (MSR) scores can measure the degree of coherence exhibited by the elements in a matrix. There, a *residue* was defined for each element in a matrix as the difference between the element and the mean of all elements of the matrix. The residue of element a_{ij} of a matrix denoted by pair (I, J) is $r_{ij} = a_{ij} - \overline{a}_{i\bullet} - \overline{a}_{\bullet j} + \overline{a}_{\bullet\bullet}$, where $\overline{a}_{i\bullet}$ is the mean of the ith row, $\overline{a}_{\bullet j}$ the mean of the jth column, and $\overline{a}_{\bullet\bullet}$ is the mean of all elements in A. The MSR of the matrix was then defined as

$$MSR(I, J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} r_{ij}^2. \tag{5.1}$$

Thus, a low value of residue typically means a high level of coherence, and vice versa [21]. For example, the MSR score of the biclusters depicted in Figure 5.2(a) and 5.2(b) is zero, since the values fluctuate in harmony. In contrast, the bicluster shown in Figure 5.2(d) is very noisy and thus has a higher MSR score. The bicluster in Figure 5.2(c) has an intermediate MSR score. Consequently, the MSR scores can be useful to evaluate the quality of biclusters of all types defined in this study.

Statistical significance

To assess the statistical significance and biological meaning of discovered biclusters, I employed a technique [98] that can compute the p-value of each bicluster with respect to known (putatively correct) biological knowledge. Suppose prior knowledge classifies N genes into M classes, $H_1, H_2, \ldots H_M$. Let P be a bicluster with g genes and assume that out of those g genes, g_j genes belong to class H_j. Assuming the



(a) $MSR = 0$ (b) $MSR = 0$

(c) $MSR = 103$ (d) $MSR = 946$

Figure 5.2: MSR scores as a measure of bicluster quality. A low MSR value typically means a high level of coherence, and vice versa [21].

most abundant class for the genes in P is H_i, the hypergeometric distribution is used to calculate p, the p-value of the bicluster P:

$$p = \sum_{k=g_i}^{g} \frac{\binom{|H_i|}{k}\binom{N-|H_i|}{g-k}}{\binom{N}{g}}. \tag{5.2}$$

That is, the p-value corresponds to the probability of obtaining at least g_i elements of the class H_i in a random set of size g. As the known biological knowledge for the experiments, the categories of yeast genes proposed by Tavazoie et al. [99] and the human genes classes reported by Higgins et al. [42] were used.

Receiver operating characteristic curve

As will be seen in Figure 5.6(b), a *receiver operating characteristic* (ROC) curve is a plot of the *true positive rate* $\left(\frac{TP}{TP+FN}\right)$ versus the *false positive rate* $\left(\frac{FP}{FP+TN}\right)$ of a screening test, where the different points on the curve correspond to different cut-off points used to designate test positives [30, 83]. The two axes of the ROC curve thus represent trade-offs between errors (false positives) and benefits (true positives) that a classifier makes between two classes [32].

The *area under the ROC curve* (AUC) is a reasonable summary of the overall diagnostic accuracy of the test [83]. Consequently, the closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test. Likewise, the closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test.

5.1.3 Implementation

The proposed algorithm was implemented in ANSI C++ with the ZBDD libraries provided by the CUDD[2] and EXTRA[3] packages. For comparison, four different biclustering techniques were implemented as listed in Table 5.1. The experiments were conducted on a 3.02 GHz Linux machine with 4 GB RAM. The specific algorithm

[2] `http://vlsi.colorado.edu/~fabio/CUDD/cuddIntro.html`
[3] `http://www.ee.pdx.edu/~alanmi/research/extra.htm`

Table 5.1: The bicluster mining methods tested in the experiments.

ID	Name/description	Algorithm employed
METHOD 1	δ-biclustering [21]	Greedy iterative search
METHOD 2	δ-pClustering [107]	Exhaustive enumeration
METHOD 3	GEMS [112]	Gibbs sampling
METHOD 4	Our method implemented without ZBDDs	Algorithms 4.4, 4.5
OUR METHOD	Our method fully implemented	Algorithms 4.4, 4.5

Table 5.2: The algorithm parameters used for the experiments.

Experiments		Algorithms and parameters							
Figure	Data	OUR METHOD	METHOD 1		METHOD 2			METHOD 3	
		τ	α	δ	δ	nr	nc	a	w
5.3	Synthetic	0	1.2	1000	0	10	10–120	0.01	250
5.4(a), 5.5(a), 5.6(a)	Yeast [22, 99]	75–80	1.2	300	10–15	80	10–12	0.25	15–30
5.1, 5.4(b), 5.5(b)	Kidney [42]	20–25	1.2	150	20–25	10–14	18–20	0.01–0.1	25–75
5.6(b)	B-cell [4]	70–75	1.2	1200	10–20	45–55	4–7	–	–

parameters used for the experiments are shown in Table 5.2.

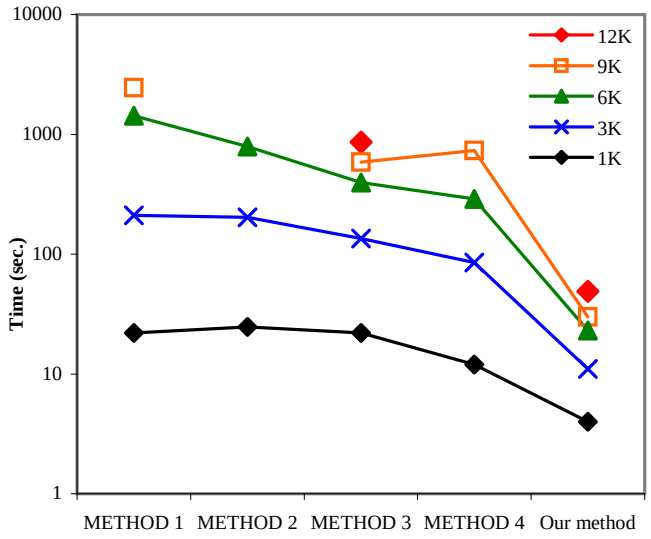
5.2 Experimental results

5.2.1 Algorithm performance evaluation

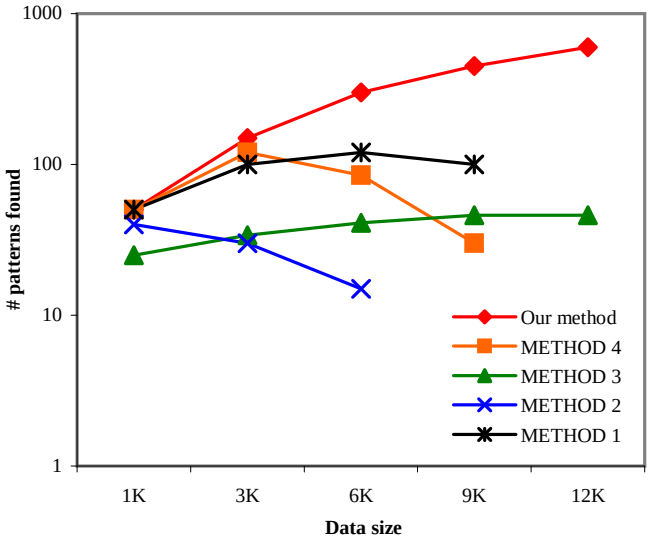
Response time and the number of discovered biclusters

We started our experiments with synthetic data sets to validate the correctness of our method. In addition, synthetic data sets can serve as convenient benchmarks to compare different algorithms. We invoked the methods listed in Table 5.1 with the parameters specified in Table 5.2. Figure 5.3(a) shows the response time spent by each method in order to find all the embedded biclusters. Figure 5.3(b) shows the plot of the total number of biclusters discovered by each method given the same time as spent by our method implemented with ZBDDs.

We then tested the algorithms with the actual gene expression data sets listed in Table 5.1, using the parameters specified in Table 5.2. In the plots in Figure 5.4(a) and 5.4(b) we compared the time to find the first k biclusters from the yeast cell



(a)



(b)

Figure 5.3: Performance comparison using synthetic data sets. The missing points on the plots mean that the corresponding experiment could not be finished in reasonable time. (a) The response time spent by each method in order to find all the embedded biclusters from the synthetic data sets of various sizes. (b) The number of biclusters found by each method within the same time spent as our method.

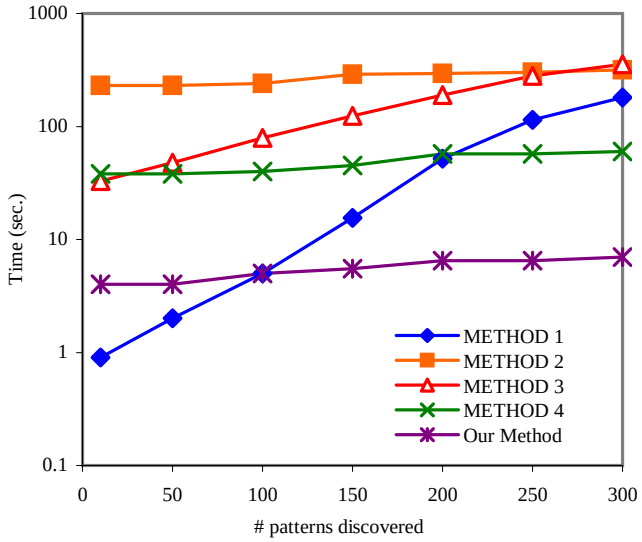
cycle data and the renal cell carcinoma data, respectively. The x-axis is the number of biclusters produced and the y-axis is the response time to find these biclusters. Our methods as well as METHOD 2 and METHOD 4 do not take as input the exact number of biclusters to find. Thus, we ran these algorithms multiple times with different parameter values to find approximately k biclusters. For METHOD 1 and METHOD 3, the exact number of biclusters to find was specified as input parameters.

We can see in the experiments that it takes less time for our method to find all the embedded biclusters and that our method can find more biclusters given the same time, compared with the other methods tested. Especially, we observed that the use of ZBDDs indeed provides a substantial speed-up over the alternative implementation without ZBDDs.

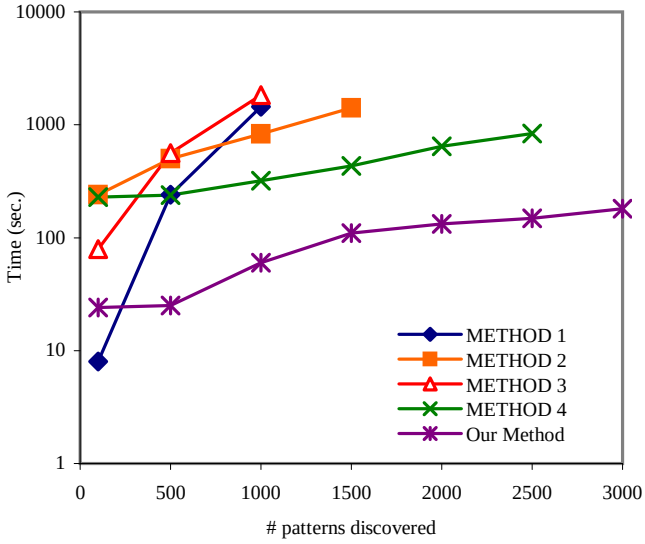
Coherence in terms of MSR scores

Figure 5.5(a) and 5.5(b) show box plots that compare the MSR scores of the biclusters discovered from the yeast cell cycle and the renal cell carcinoma data, respectively. A *box plot* is a plot that represents graphically several descriptive statistics such as median and percentiles of a data sample [29]. The reader can refer to the caption of Figure 5.5 to find out how to read a box plot.

As is evident from the box plots, the biclusters found by our method have the lowest median MSR scores in our experiments. To quantitatively establish this observation, we performed the Wilcoxon rank sum test [29] to compare the biclusters discovered by our method with the others. We generated approximately 500 biclusters per method for each data set and compared a group of biclusters found by our method with another group of biclusters detected by an alternative method. In all the cases we tested, the difference in the median was statistically significant at 0.01% level ($P < 0.0001$). This result shows that our method tends to find better biclusters with respect to the MSR scores.

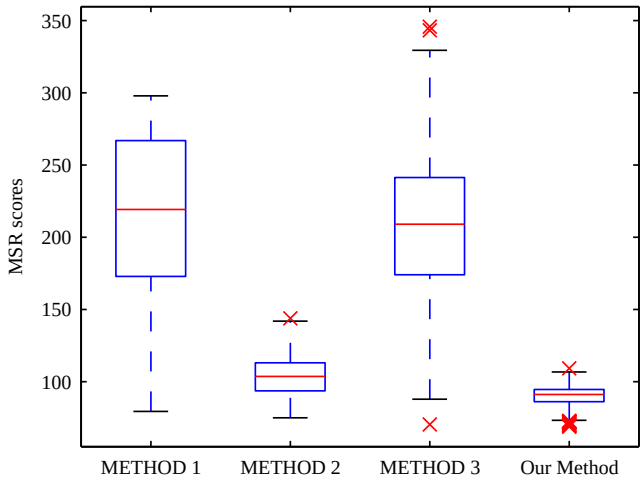


(a) Yeast cell cycle data [22, 99]

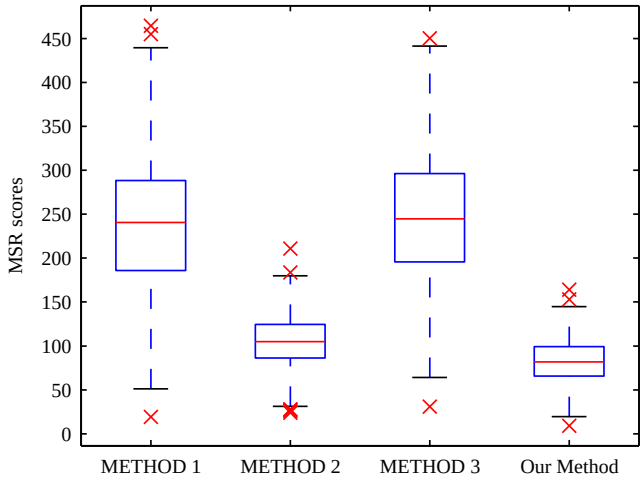


(b) Renal cell carcinoma data [42]

Figure 5.4: Performance comparison using biological data sets. The missing points on the plots mean that the corresponding experiment could not be finished in reasonable time.



(a) Yeast cell cycle [99]



(b) Renal cell carcinoma [42]

Figure 5.5: Box plots for MSR comparison. The line in the middle of a box indicates the position of the median. The upper and lower boundaries of the box represent the location of the 75th and 25th percentiles, respectively. The symbol 'x' outside the ends of the tails corresponds to outliers.

5.2.2 Bicluster quality evaluation

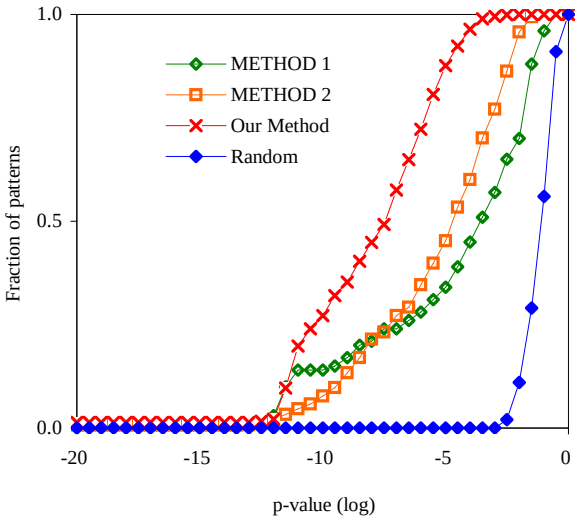
The experiments presented so far have demonstrated that our method outperforms the alternatives in terms of efficiency and the number of biclusters found. Here we present more experimental results to show that our method can produce statistically more significant and biologically more meaningful biclusters, thus suggesting that our method can be helpful to the researchers in biomedicine as well.

To this end, we let each algorithm find biclusters from the biological data sets used and then calculated the p-value of each biclusters by the method explained in Section 5.1.2. Figure 5.6(a) presents the correspondence plot [98] for the biclusters generated by several different methods from the yeast data set. The plot presents the fraction of biclusters whose p-value is at most P out of the n best biclusters discovered, for each p-value P on the plot. In this plot, early departure of a curve from the x-axis indicates the existence of biclusters with low p-values. Consequently, the area under a curve approximately shows the degree of statistical significance of the biclusters used to draw the curve. Figure 5.6(a) includes randomly generated biclusters. This plot indicates that the biclusters shown are all far from the random noise. It also demonstrates that the biclusters generated by our algorithm tend to be more statistically significant than the others, meaning that our biclusters conform to the known classification more accurately.

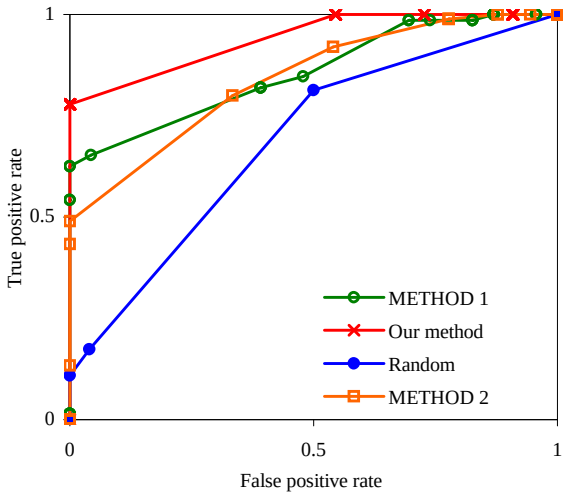
For additional bicluster quality evaluation, we produced ROC curves based upon the biclustering results from the B-cell lymphoma data set. Figure 5.6(b) presents the results.

Since any clustering algorithm corresponds to the unsupervised learning method, I associated each bicluster with known classes in order to draw an ROC curve. Suppose prior knowledge classifies M samples into two classes, P and N. Let B be a bicluster in which m_P samples belong to the class P and m_N samples belong to the class N. The class of bicluster B is set to P if $\frac{m_N}{m_P+m_N} < t$ for a given threshold t. Otherwise, the class of B is set to N. Determining the class of a bicluster thus corresponds to a test. Each sample is classified into one of (TP, TN, FP, FN) as usual, and the ROC curve is drawn varying the parameter t.

This result shows that our algorithm has a better characteristic in the ROC space



(a) Correspondence plot



(b) ROC curve

Figure 5.6: (a) The correspondence plot depicts the distribution of p-values of the produced biclusters with respect to a known classification of experimental conditions or gene annotations. The plot presents the fraction of biclusters whose p-value is at most P out of the n best biclusters discovered, for each p-value P on the plot. We used the yeast cell cycle data [99]. (b) In a receiver operating characteristic (ROC) curve, the abscissa is the false positive rate, $FP/(FP+TN)$, and the ordinate is the true positive rate, $TP/(TP+FN)$. We used the B-cell lymphoma data [4].

than the alternative methods tested. Consequently, the biclusters produced by our method tend to represent a more accurate classification of the genes or samples involved.

5.3 Summary

We performed experimental studies to analyze the performance of our algorithm on several benchmarks, including both synthetic and real expression data sets. It was observed that the proposed method outperforms some alternative methods not only in terms of efficiency but also with respect to the total number of biclusters discovered. In particular, it was confirmed that the use of ZBDDs can greatly enhance the scalability of our approach and enable the users to apply it to large-scale data sets. In addition, the biclusters discovered by the proposed technique tend to conform with the prior biological knowledge more closely.

Microarray analysis is improving our understanding of biomedical diagnosis and prognosis. For instance, transcription profiling using DNA microarrays has great potential as a systematic approach for discovering new classes of diseases and for assigning known diseases to classes in order to predict response to therapy. Thus, it is perceived that gene expression monitoring could provide new insights into many aspects of disease pathology, and the method described in this dissertation can be an invaluable tool for this.

Chapter 6

Linking Gene Expression and Clinical Traits

This chapter introduces a computational method to link clinical traits with genes that are potentially responsible for these traits. This method is based upon the biclustering algorithm explained in Chapter 4.

6.1 Introduction

The invention of DNA microarray technologies has enabled researchers to simultaneously monitor the expression level of virtually all known genes [29, 52]. Thus, for the purpose of finding genes related to a certain clinical trait (or parameter) of interest, it has become feasible to examine all the genes available and then select only those whose expression is consistently correlated with the trait over many samples. Although correlation does not always imply causality, this approach has been successful in many studies as an attempt to understand genetic mechanisms underlying clinical observations [17, 102, 109, 111].

To measure correlation between a gene and a clinical trait, existing approaches obtain a vector of the expression level of the gene over a number of samples and another vector of the value of the clinical trait over the same samples and then calculate statistical correlation between two vectors. By applying this procedure to

many genes, we can identify some genes correlated to the clinical trait of interest.

Proceeding one step further from prior methods that can reveal one-to-many relationships between a single trait and multiple genes (or vice versa), this chapter introduces a method that can find many-to-many relationships between multiple genes and traits using the co-clustering method proposed in Chapter 4.

Given gene expression data and clinical parameter values, we first create a matrix called *correlation matrix* that can collectively represent the degree of correlation between genes and clinical traits. Each row and column of this matrix corresponds to a gene and a clinical trait, respectively. Then, our method searches *co-clusters* or submatrices (with some semantics to be defined) covering the correlation matrix. As will be clear shortly, these co-clusters can be found by the algorithm to find Type 3 nested biclusters explained in Chapter 4.

Example 6.1. *Figure 6.1 shows an example of co-clustering genes and clinical traits. Figure 6.1(a) is a matrix of imaging traits (*e.g.*, tumor size) derived from the brain image shown in Figure 6.1(b). An example of a gene expression matrix is shown in Figure 6.1(c). In this matrix, the arrangement of the columns (patients) should be identical to the arrangement of the columns in Figure 6.1(a). Then, it is possible to calculate the Pearson correlation coefficient [83] between two row vectors, one from the trait matrix and the other from the gene expression matrix. Figure 6.1(d) shows a plot of the coefficients for every pair of rows. This plot can be represented by a matrix shown in Figure 6.1(e). This matrix is an example of the correlation matrix, and co-clusters of genes and clinical traits can be found from this matrix. In this particular example, Genes 2, 3, and 4 were found to be linked to Traits 1 and 2.*

We tested our method with the *acute myelogenous leukemia* (AML) data set [17], which consists of a DNA microarray data matrix and a parameter matrix for 119 patients, 15 parameters, and 6283 genes. We identified 43 co-clusters using the proposed method. To justify the grouping of certain genes and clinical traits by the co-clusters found from the AML data, we present some supporting evidence of co-clustered genes and traits from the literature. In addition, we show that certain *gene*

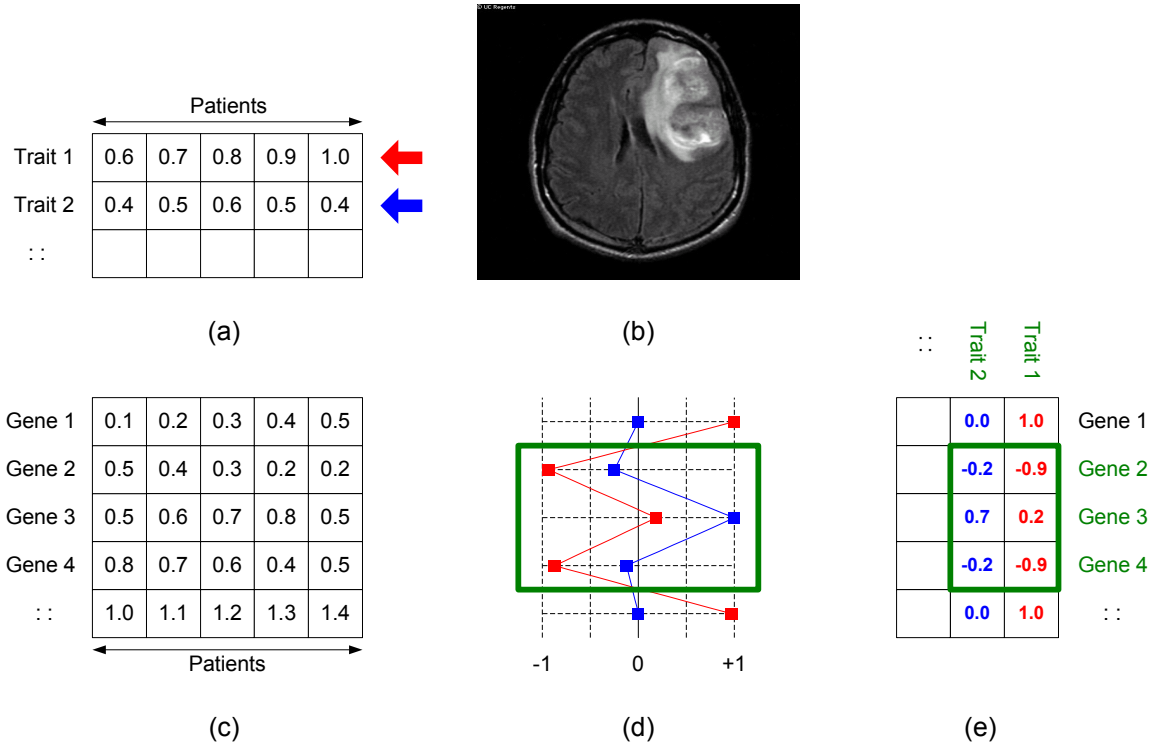


Figure 6.1: An example of co-clustering genes and clinical traits. (a) A matrix of clinical traits over a set of patients. This matrix was derived from the brain image shown in (b). Trait 1 can be, for instance, the size of a tumor. (b) The brain image. (c) A gene expression matrix. Here the columns are arranged in the same order as in (a). (d) A plot showing the correlation coefficient between two rows in the matrices in (a) and (b). (e) A correlation matrix from which co-clusters are found.

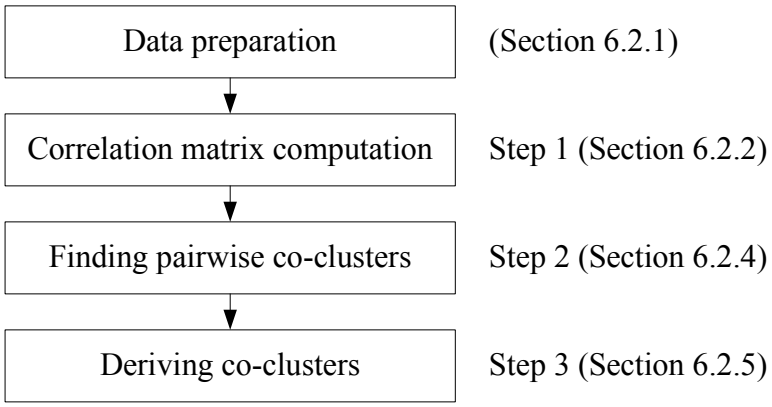


Figure 6.2: A flowchart of the method to find co-clusters of genes and clinical traits.

ontology (GO) [100] terms annotating genes in some co-clusters are significantly over-represented. Taken together, these experimental studies suggest that our method can reveal biologically meaningful connections between gene expression and clinical traits.

The remainder of this chapter is organized as follows. Section 6.2 explains at length our method to find co-clusters of genes and clinical traits. Experimental results and discussions are presented in Section 6.3, followed by a summary in Section 6.4.

6.2 Method

The input of our method consists of two data matrices. One is a gene expression data matrix, and the other is a matrix of clinical parameter values. The output is a set of co-clusters. A co-cluster is composed of a subset of the genes in the gene expression data and a subset of the clinical parameters in the parameter matrix. Informally, a co-cluster is a group of genes and traits that are closely related to each other, given the input matrices. A formal definition will be presented in Section 6.2.3.

As seen in Figure 6.2, the proposed method performs three major steps excluding data preparation.

Step 1 (Section 6.2.2) An intermediate data matrix called *correlation matrix* is constructed from the input matrices.

Step 2 (Section 6.2.4) A special type of co-clusters called *pairwise co-clusters* are found in the correlation matrix.

Step 3 (Section 6.2.5) Co-clusters are derived from the pairwise co-clusters previously discovered.

6.2.1 Data preparation

Let S represent a set of clinical samples. For each sample in S, gene expression levels are measured by the DNA microarray technology of choice, and we let G be the set of genes in the measurement. In addition, certain clinical traits are recorded for each sample, and we let T be the set of the recorded traits.

The input of our method consists of two data matrices constructed from the above experiments. One is a gene expression data matrix denoted by pair $A = (G, S)$. That is, $A \in \mathbb{R}^{|G| \times |S|}$, and the element a_{ik} of the matrix A represents the expression level of gene i for sample k. The other matrix is denoted by pair $B = (T, S)$, where the element b_{jk} of the matrix B corresponds to the value of trait j for sample k. Depending upon the type of trait j, b_{jk} may be quantitative, categorical, or others, as listed in Table 6.1. We arrange the columns of A and B in the same order.

6.2.2 Correlation matrix computation

The first step of our method is to construct a correlation matrix by combining the input matrices A and B. As illustrated in Figure 6.3, the row set and the column set of the correlation matrix C are G and T, respectively. Each element c_{ij} of C indicates the degree of correlation between gene i and trait j. To calculate c_{ij}, we take advantage of an existing statistical method. More precisely, the element c_{ij} is the statistic defined in *significance analysis of microarrays* (SAM) [102], namely,

$$c_{ij} = \frac{r_{ij}}{s_{ij} + s_0}. \tag{6.1}$$

In this statistic, r_{ij} is a score that measures the degree of correlation between the expression level of gene i and the value of clinical trait j, s_{ij} is the "gene-specific

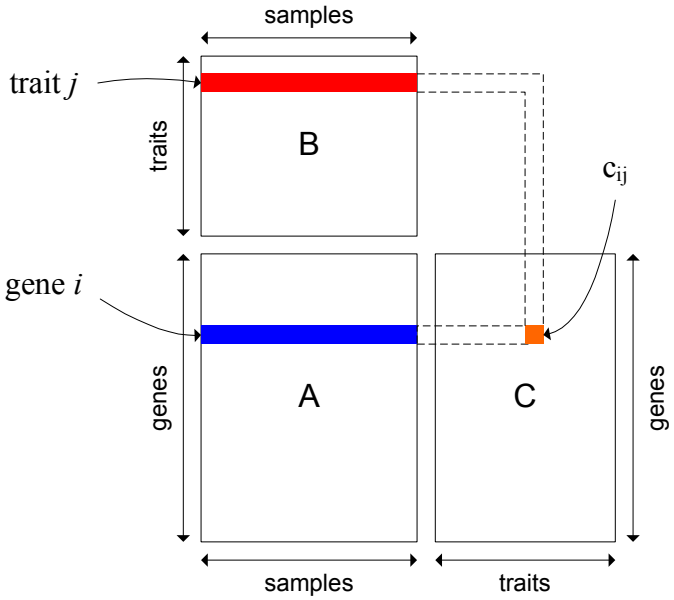


Figure 6.3: Construction of the correlation matrix. A co-cluster appears as a submatrix of the correlation matrix C.

scatter" or the standard deviation of repeated expression measurements, and s_0 is a "fudge" factor to prevent the computed statistic from becoming too large when s_{ij} is close to zero [29]. Intuitively, c_{ij} is a correlation coefficient over a standard deviation and is very similar to the t-statistic [83] for hypothesis testing.

This particular statistic was chosen over more conventional ones because c_{ij} is defined for various data types such as quantitative, categorical, and others. Table 6.1 lists the types of clinical traits that can be handled by the SAM procedure and high-level descriptions of how r_{ij} is defined for each type.

Although the details on computing the statistic c_{ij} is beyond the scope of this dissertation and is available elsewhere (*e.g.,* [29, 102]), we provide in this section an outline of computing c_{ij} and assessing its statistical significance, for completeness and clarity.

Computing c_{ij} The specific definition of r_{ij} varies depending upon the type of clinical trait j. For example, if clinical trait j has quantitative values then r_{ij} is defined in terms of the Pearson correlation coefficient [83] between the i-th row vector

Table 6.1: Types of clinical traits and the corresponding definitions of the score r_{ij} [102]. In the last line, the value 1 represents a death whereas 0 means censored data.

Type of trait	Example	Concepts used to define r_{ij}
Quantitative	$-1.2, 3.0, \ldots$	The Pearson correlation coefficient
Binary class	{death,survival}	The relative difference of two means
Multiclass	{state 0,1,2,3}	The Fisher's linear discriminant
Survival data	(time,[1|0])	Cox's proportional hazards function

of the matrix A and the j-th row vector of the matrix B, and s_{ij} is the standard error of r_{ij}.

Example 6.2. *Assume that the instances of trait j in the trait matrix $B = (T, S)$ have quantitative values. Then, the statistic r_{ij} for gene i in the gene expression matrix A and trait j is defined as follows:*

$$r_{ij} = \frac{\sum_k b_{jk}(a_{ik} - \overline{a}_i)}{\sum_k (b_{jk} - \overline{b}_j)^2} \tag{6.2}$$

where $\overline{a}_i = \sum_k a_{ik}/|S|$ and $\overline{b}_j = \sum_k b_{jk}/|S|$. s_{ij} is the standard error of r_{ij}:

$$s_{ij} = \frac{\hat{\sigma}_{ij}}{\sqrt{\sum_k (b_{jk} - \overline{b}_j)^2}} \tag{6.3}$$

where $\hat{\sigma}_{ij}$ is the square root of residual error:

$$\hat{\sigma}_{ij} = \sqrt{\frac{\sum_k (a_{ik} - \hat{a}_{ijk})^2}{n-2}} \tag{6.4}$$

$$\hat{a}_{ijk} = \hat{\beta}_{ij0} + r_{ij} b_{jk} \tag{6.5}$$

$$\hat{\beta}_{ij0} = \overline{a}_k - r_{ij}\overline{b}_k \tag{6.6}$$

For the multiclass type of traits (*e.g.,* trait $\in \{state_1, state_2, state_3, \ldots, state_n\}$), r_{ij} and s_{ij} are defined using the Fisher's linear discriminant [83] as follows.

Example 6.3. *Suppose that trait j in the trait matrix $B = (T, S)$ represent multiclass*

responses and can have values chosen from $\{1, 2, \ldots, H\}$*, a set of* H *classes. Let* C_h *denote the indices of observations in class* h*,* $n_h = |C_h|$*,* $\overline{a}_{ih} = \sum_{c \in C_h} a_{ic}/n_h$*, and* $\overline{a}_i = \sum_{k=1}^{|S|} a_{ik}/|S|$*. Then,* r_{ij} *and* s_{ij} *are defined as follows:*

$$r_{ij} = \sqrt{\frac{\sum n_h}{\prod n_h} \sum_{h=1}^{H} n_h (\overline{a}_{ih} - \overline{a}_i)^2} \tag{6.7}$$

$$s_{ij} = \sqrt{\frac{1}{\sum (n_h - 1)} \cdot \left(\sum \frac{1}{n_h} \right) \sum_{h=1}^{H} \sum_{c \in C_h} (a_{ic} - \overline{a}_{ih})^2} \tag{6.8}$$

Details on computing r_{ij} and s_{ij} for other types of traits can be found in [102]. The fudge factor s_0 is set to the γ percentile of the s_{ij} values, where γ can be determined adaptively according to the distribution of s_{ij} or can be fixed to a constant value such as $\gamma = 5\%$.

The problem of multiple comparisons When calculating c_{ij}, we must follow a procedure for multiple comparisons, thus ensuring that too many falsely significant ones are not declared [29,83]. In statistical hypothesis testing, a *Type I error* consists of rejecting a null hypothesis that is true [29, 83]. It is the equivalent of a "false positive" or "false alarm." The probability of Type I error is often denoted by symbol α. Statistical analysis typically involves not only a single hypothesis testing but multiple hypotheses testing. The problem is that using a value of, for example, $\alpha = 0.05$ means that approximately one out of every twenty such tests will produce a false positive. Consequently, if we perform a series of hypothesis testing for 10,000 genes, we can expect as many as 500 genes to be declared as significant if we perform a single test per gene with the significance level of $\alpha = 0.05$.

Conventional correction methods We need to correct the problem of multiple comparisons and control the false positive rate not only for any single test but also for the entire set of tests involved in our experiment. Let α_e denote the experiment-wise false positive rate. We can also consider symbol α, the Type I error rate, as each comparison-wise false positive rate. Then, the probability of at least one false positive

is

$$\alpha_e = 1 - (1 - \alpha)^n \tag{6.9}$$

assuming that the number of independent tests is n, and each test has α as its Type I error rate. If we want to determine α, an experiment-wise error rate that can guarantee α_e, a specific experiment-wise error rate, we can solve α for α_e:

$$\alpha = 1 - (1 - \alpha_e)^{\frac{1}{n}} \tag{6.10}$$

which is often referred to as the *Dunn-Šidák correction method* [105]. Since $(1-\alpha_e)^{\frac{1}{n}} \simeq 1 - \frac{\alpha_e}{n}$ for small values of α_e, we can also approximate α as

$$\alpha = \frac{\alpha_e}{n} \tag{6.11}$$

which is called the *Bonferroni correction method* [83].

Adaptive correction methods The correction methods described previously use the same values of α for every test. In addition, when n is quite large, as is typical in most genomics problems, the value of α can be too tiny and no test can find significant results in any circumstances. In contrast, adaptive correction methods generate the distribution of p-values observed from the entire tests and use different values of α for each test, depending upon the location of its p-value in the distribution. For instance, the *Holm's step-wise correction method* [43] orders the tests in the increasing order of the p-values of the individual tests and uses $\frac{\alpha_e}{n-i+1}$ as the value of α for the test located in the i-th order. Thus, the test with the smallest p-value uses $\alpha = \frac{\alpha_e}{n}$, the test with the second smallest p-value uses $\alpha = \frac{\alpha_e}{n-1}$, and so on. Benjamini and Hochberg [11] proposed a similar method, where the test with i-th largest p-value employs $\alpha = \left(\frac{i}{n}\right)\alpha_e$. Westfall and Young [110] proposed a more general method that is based upon the sample-label permutation, which is similar to the bootstrapping method [83] conducted by sampling without replacement [29].

The SAM procedure SAM uses the same permutation idea to estimate the percentage of the genes called significant by chance. This rate is referred to as the *false discovery rate* (FDR) and is calculated as follows [102]:

1. For given j, compute statistic c_{ij} for $i = 1, 2, \ldots, |G|$, where $|G|$ is the number of genes in the gene expression matrix.

2. Compute order statistics $c_{(1)} \leq c_{(2)} \cdots \leq c_{(|G|)}$.

3. Take M sets of permutations of the vector associated with trait j. For each permutation m, compute statistics c_{ij}^{*m} and corresponding order statistics.

4. From the set of M permutations, estimate the expected order statistics by $\overline{c}_{(i)} = (1/M) \sum_m c_{(i)}^{*m}$ for $i = 1, 2, \ldots, |G|$.

5. Plot the values of $c_{(i)}$ versus the values of $\overline{c}_{(i)}$.

6. For Δ, a fixed threshold, find the first $i = i_1$ such that $c_{(i)} - \overline{c}_{(i)} > \Delta$, starting at the origin and moving up to the right. All genes past i_1 are called *significant positive*. Similarly, find *significant negative* genes. For each Δ, define the upper cut-point $cut_{up}(\Delta)$ as the smallest c_{ij} among the significant positive genes, and similarly define the lower cut-point $cut_{low}(\Delta)$.

7. For a grid of Δ values, compute the total number of significant genes (from the previous step), and the median number of falsely called genes, by computing the median number of values among each of the M sets of $c_{(i)}^{*m}$ (for $i = 1, 2, \ldots, |G|$) that fall above $cut_{up}(\Delta)$ or below $cut_{low}(\Delta)$.

8. Estimate P_0, the proportion of true null (unaffected) genes in the data set (see [102] for details).

9. The median of the number of falsely called genes from Step 6 is scaled appropriately, according to the value of P_0 (see [102] for details).

10. A value of Δ can be specified by the user and the significant genes are listed.

11. The FDR is computed as the median of the number of falsely called genes divided by the number of genes called significant.

Using the notion of FDR, the p-value of the statistic c_{ij} is defined as follows.

Definition 6.1. *The p-value for score c_{ij}, denoted by p-value(c_{ij}), is the lowest FDR at which gene i is called significantly correlated with trait j.*

In summary, to construct the correlation matrix, we use the SAM statistic, which is a modified t-test statistic (or F-test statistic for multiclass analysis), with sample-label permutations to evaluate statistical significance [17].

6.2.3 Defining co-clusters

Co-clusters are defined on the correlation matrix. Informally, we are interested in finding a submatrix (of the correlation matrix) in which the values on all columns exhibit some common behavior. Co-clusters in this definition are similar to Type 3 nested biclusters in Chapter 4. An example is presented in Figures 6.5(a) and 6.5(b). In particular, we focus on searching submatrices where every pair of column vectors show positive or negative correlation as seen in Figures 6.5(c) and 6.5(d).

To assess the degree of correlation, we introduce a metric called *linear deviation*, which resembles a conventional statistic such as the Pearson correlation coefficient but can be computed more efficiently, especially in the current setup where we want to measure correlation between many sub-vectors of two vectors. The introduction of this metric is not to deny the effectiveness of a conventional statistic but to transform it to a computation-efficient form, minimizing loss in the detection power.

Definition 6.2. *For V, a vector on $\mathbb{R}$, the range of V, denoted by* $\textsc{range}(V)$*, is the absolute difference between the largest and the smallest elements of V.*

Note that this range operator is similar to that in descriptive statistics, where the range is the length of the smallest interval which contains all the data [93]. There, the range is also calculated by subtracting the smallest observations from the greatest and provides an indication of statistical dispersion.

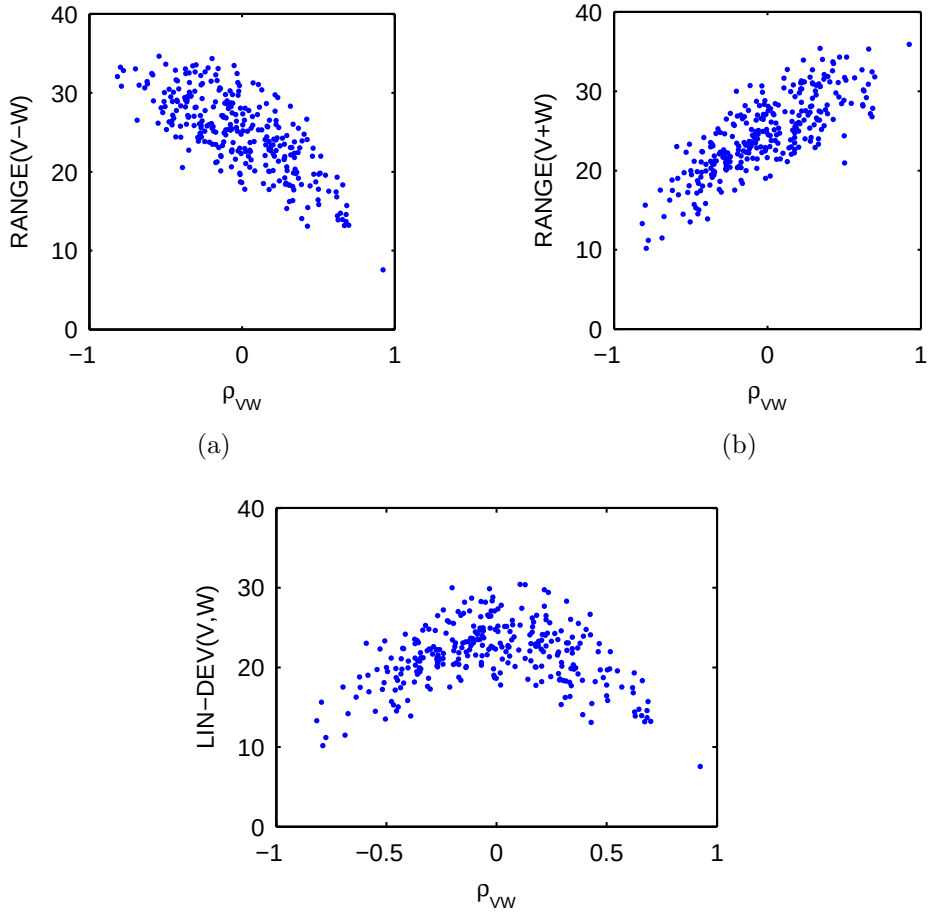


Figure 6.4: An empirical study to show the relationship between LIN-DEV and the Pearson correlation coefficient. (a) A pair of 10-dimensional vectors (V, W) were generated in such a way that the Pearson correlation coefficient ρ_{VW} between the two vectors randomly lies in $[-1, 1]$. The elements of the two vectors were ranged in $[-10, +10]$. We also calculated $\text{RANGE}(V - W)$ and then place point $(\rho_{VW}, \text{RANGE}(V - W))$ on a plot. We repeated this procedure 300 times. Two 300-dimensional vectors X and Y were created from the 300 points (x, y) on the plot. We calculated the Pearson correlation coefficient ρ_{XY} between vectors X and Y to obtain $\rho_{XY} = -0.7433$ $(P < 10^{-54})$, indicating significant negative correlation between $\text{RANGE}(V - W)$ and ρ_{VW}. (b) The same procedure repeated for a plot of ρ_{VW} versus $\text{RANGE}(V + W)$: $\rho_{XY} = 0.7934$ $(P < 10^{-66})$. (c) Focusing on a low value of the metric LIN-DEV thus enables us to detect either strongly positive or highly negative correlation between V and W.

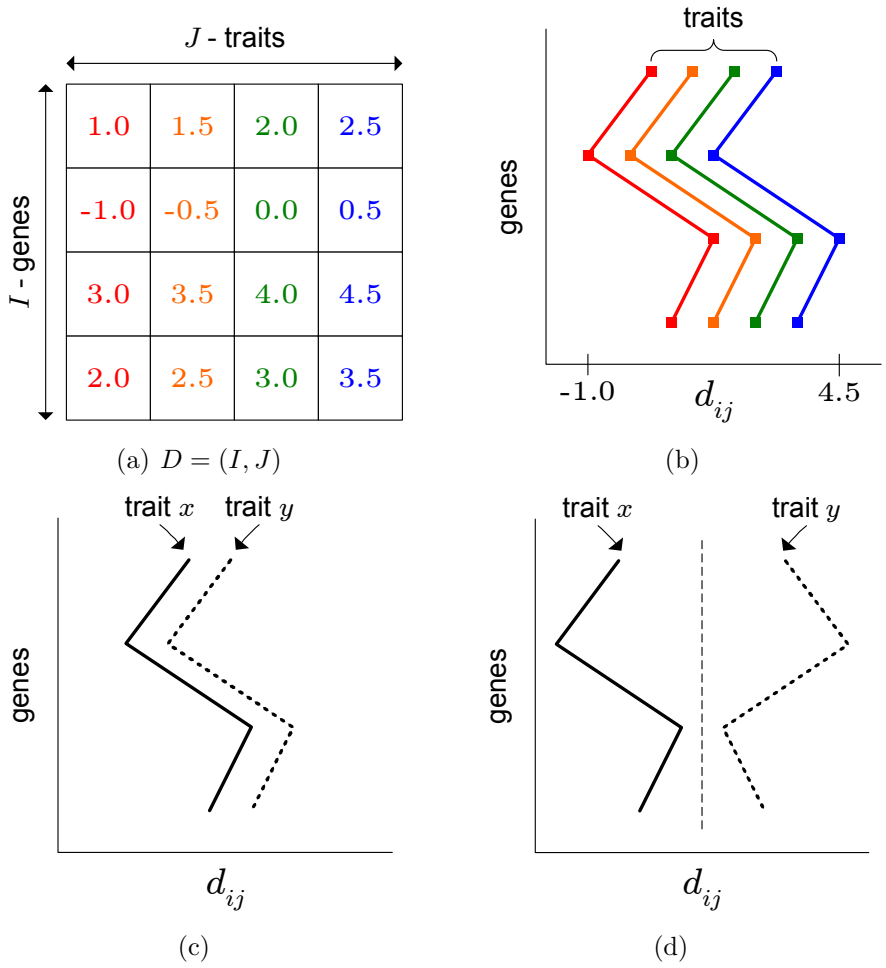


Figure 6.5: Defining co-clusters. (a) $D = (I, J)$, an example co-cluster with the gene set I and the trait set J. (b) The column vectors of D show the same trend. (c) Positive correlation between traits x and y. (c) Negative correlation between traits x and y.

Using the notion of this RANGE$(\cdot)$ operator, we define the *linear deviation* of two vectors as follows.

Definition 6.3. *Given V and W, two real vectors of the same dimension, the* linear deviation *of V and W, denoted by* LIN-DEV(V, W)*, is defined as*[1]

$$min\{\text{RANGE}(V - W), \text{RANGE}(V + W)\}. \tag{6.12}$$

The example in Figure 6.4 reveals the relationship between LIN-DEV and the Pearson correlation coefficient: a lower value of LIN-DEV typically corresponds to a higher level of either positive or negative correlation. Using the metric LIN-DEV, a co-cluster is formally defined as follows.

Definition 6.4. *Given the correlation matrix $C = (G, T)$ and thresholds $\tau \geq 0$ and $\pi > 0$, a* co-cluster *is a matrix, denoted by $D = (I, J)$, satisfying the following conditions:* (1) *$I \subseteq G$ and $J \subseteq T$;* (2) *for any two column vectors V and W of size $|I|$ in D,* LIN-DEV$(V, W) \leq \tau$*;* (3) *$\forall i \in I, \forall j \in J$, p-value$(c_{ij}) < \pi$.*

Condition (1) indicates that D is a submatrix of the correlation matrix C. Condition (2) is to require that every pair of $|I|$-dimensional column vectors from D exhibit correlation with respect to the metric LIN-DEV. The last condition is to find co-clusters with statistically significant elements.

In this study, we search only *maximal* co-clusters or those that are not contained by others, because non-maximal co-clusters are redundant.

6.2.4 Discovering pairwise co-clusters

Since co-clusters in the present definition are nested biclusters, we can follow an approach similar to the nested bicluster mining described in Chapter 4. Thus, after having computed the correlation matrix, the next step of our method is to find a special type of co-cluster called *pairwise co-cluster*, which corresponds to the atomic nested biclusters defined in Chapter 4. That is, a pairwise co-cluster is a co-cluster

[1]Since RANGE$(-V)$ = RANGE(V) by definition, we do not need to consider RANGE$(-V + W)$ or RANGE$(-V - W)$.

Algorithm 6.1: Find positively correlated dimensions for two vectors

```
   input  : V and W, two n-dimensional vectors
   input  : τ, a threshold
   output: I ⊆ {1, 2, ..., n}, a set of dimensions
 1 for i = 1 to n do
 2     S[i].score := V_i − W_i;
 3     S[i].dim := i;
 4 sort S in ascending order with respect to the field score;
 5 begin := 1, end := 2;
 6 while (end ≤ n) do
 7     if (S[end].score − S[begin].score ≤ τ) then
 8         end := end + 1;
 9         if (end > n) then
10             Report {S[begin].dim, ..., S[end − 1].dim};
11     else
12         Report {S[begin].dim, ..., S[end − 1].dim};
13         repeat
14             begin := begin + 1;
15         until (begin = end) or (S[end].score − S[begin].score ≤ τ);
```

Figure 6.6: Algorithm to find positively correlated dimensions.

with only two traits and can therefore be represented by a submatrix (of the correlation matrix) with two columns. Pairwise co-clusters are used later in Section 6.2.5 as seeds to find (non-pairwise) co-clusters.

To find a pairwise co-cluster in the correlation matrix $C = (G, T)$, we first select two distinct columns $v, w \in T$ and construct from them two $|G|$-dimensional column vectors $V = (c_{1v}, c_{2v}, \ldots, c_{|G|v})$ and $W = (c_{1w}, c_{2w}, \ldots, c_{|G|w})$. Then, we compare V and W to identify I, a set of dimensions over which V and W are correlated ($I \subseteq G$). Finally, we remove all $i \in I$ such that p-value of c_{iv} or c_{iw} is greater a given threshold. By Definition 6.4, the matrix denoted by pair $(I, \{v, w\})$ represents a co-cluster, and this co-cluster with only a pair of traits is called *pairwise co-cluster*.

Here we further explain the procedure to compare two vectors V and W and identify the dimension set I. The other details on finding pairwise co-clusters are straightforward and thus omitted.

Algorithm 6.1 presents the procedure to find I, a set of dimensions over which two vectors V and W are *positively* correlated. Invoking this algorithm with $-V, W$ or

$V, -W$ provides a set of negatively correlated dimensions. This algorithm resembles Algorithm 4.3 in Chapter 4.

The key idea of Algorithm 6.1 is simple: when the elements of a vector V are arranged in an ascending or descending order, $range(V)$ is simply the absolute difference between the first and the last elements of V, and no other elements need to be examined. Thus, in Lines 1–3, the vector $S = V - W$ is rearranged in ascending order. Then, in Lines 6–15, the algorithm examines sub-vectors of S and reports those whose range is not greater than the threshold τ. The boundary of a sub-vector under consideration is indicated by two pointers *begin* and *end*. The algorithm relies on these pointers to find only maximal subsets and to handle multiple (and possibly overlapping) instances of I.

The worst-case complexity of Algorithm 6.1 is polynomial in n, the number of dimensions in two vectors.

6.2.5 Deriving co-clusters

In the last step of our method, co-clusters are derived from pairwise co-clusters. The approach taken here is the same as the depth-first pattern mining method explained in Section 4.3.3 but is focused on the case for Type 3 nested biclusters.

For the sake of explanation, let pair (I, J) represent a pairwise co-cluster with $J = \{x, y\}$ and assume that we want to expand this co-cluster "horizontally" by adding z, a third column index, to the set J. Let (I', J') denote this new co-cluster. Since we are interested in finding only maximal co-clusters, assume that we are to find the instance of I' with maximal cardinality.

Clearly, the set J' is a superset of J, namely, $J' = J \cup \{z\} = \{x, y, z\}$. In contrast, the set I' is a subset of I by construction[2]. In what follows, we explain more precisely what the set I' should be.

First, $I' \subseteq I$ as previously stated. Second, if the pair $(I', \{x, y, z\})$ represents a co-cluster, then by definition, $(I', \{x, z\})$, $(I', \{y, z\})$, and $(I', \{x, y\})$ should be pairwise co-clusters. Now let I_{xz} and I_{yz} be the row sets of pairwise co-clusters obtained

[2]If any two trait vectors show a common trend over $|G|$ dimensions in the correlation matrix, then three traits including the two traits cannot show a common trend over more than $|G|$ dimensions.

by Algorithm 1 for column pairs $\{x, z\}$ and $\{y, z\}$, respectively. Then, $I' \subseteq I_{xz}$ and $I' \subseteq I_{yz}$, since Algorithm 1 finds only maximal pairwise co-clusters. Therefore, $I' \subseteq I \cap I_{xz} \cap I_{yz}$, and we can obtain the instance of I' with the largest cardinality by setting $I' = I \cap I_{xz} \cap I_{yz}$.

In general, given a co-cluster (I, J), we can add element z to the set J and produce a new maximal co-cluster $(I', J \cup \{z\})$ with

$$I' = I \cap \left(\bigcap_{\forall j \in J} I_{jz} \right), \tag{6.13}$$

where I_{jz} is a maximal pairwise co-cluster for columns $\{j, z\}$. Our approach to deriving co-clusters from pairwise co-clusters is based upon this idea.

Algorithm 6.2 provides the details. This algorithm is in essence equivalent to Algorithm 4.5 in Chapter 4. Recall that T is the set of clinical traits or the set of column indices in the correlation matrix $C = (G, T)$. Algorithm 6.2 examines elements $J \in 2^T$ in such an order that Equation 6.13 can be exploited to find a co-cluster (I, J). To this end, a data structure called *prefix tree* or *trie* [2] is employed to systematically represent the elements of the power set 2^T. For the sake of explanation, assume that $T = \{1, 2, 3, 4\}$. The prefix tree representing the power set 2^T is depicted in Figure 6.8(a). Each vertex v of the prefix tree is associated with two sets $v.I$ and $v.J$ such that $v.I \subseteq G$ and $v.J \subseteq T$. Indicated inside a vertex in Figure 6.8(a) is $v.J$.

The prefix tree is traversed in preorder by Algorithm 6.2. In the worst case, the algorithm needs to visit every vertex of the prefix tree. Thus, the worst-case complexity of Algorithm 2 is exponential in $|T|$. However, in most cases, this exhaustive enumeration is avoided, and the running time of our algorithm on typical benchmarks is practical. To see the reason, observe that the subtree rooted at a vertex v with $v.I = \emptyset$ needs not be visited and removed from the prefix tree. The condition $v.I = \emptyset$ means that the matrix represented by the pair $(v.I, v.J)$ cannot be a co-cluster. Thus, any pair (I', J') with $J' \supseteq v.J$ cannot represent a co-cluster either, regardless of the set I'. For instance, assume that $v.I = \emptyset$ for the top left vertex v with $v.J = \{1, 2\}$. Then, as shown in Figure 6.8(b), the vertex v and the subtree rooted at v are removed from the tree, producing the reduced tree in Figure 6.8(c).

Algorithm 6.2: Mining co-clusters

input : $C = (G, T)$, a correlation matrix
output: co-clusters

Generate pairwise co-clusters by Algorithm 1;
foreach $\{x, y\} \subseteq T$ **do**
    Create vertex v;
    $v.J := \{x, y\}$;
    $ConstructTrieInPreorder(v)$;
    delete v;
Remove redundancy and return remaining co-clusters;

procedure $ConstructTrieInPreorder$ (vertex v)
begin
    if $|v.J| = |\{x, y\}| = 2$ **then**
        $v.I := I_{xy}$;
    else
        vertex $p := v.parent$;
        $k :=$ the element in $v.J - p.J$;
        $v.I := p.I \cap (\cap_{\forall j \in p.J} I_{jk})$;
    if $v.I = \emptyset$ **then** return;
    Collect pattern $(v.I, v.J)$;
    $l :=$ the "largest" element in $v.J$ wrt a total order $\prec$;
    $J := \{j | j \in T$ and $l \prec j\}$;
    foreach $j \in J$ **do**
        create vertex w;
        $w.J := v.J \cup \{j\}$;
        $w.parent := v$;
        $ConstructTrieInPreorder(w)$;
        delete w;
end

Figure 6.7: Algorithm to find co-clusters.

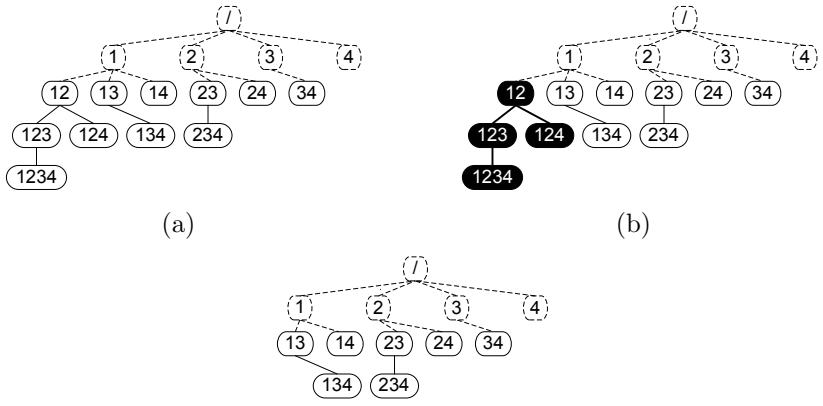


Figure 6.8: Prefix tree example. Each vertex v is associated with two sets $v.I$ and $v.J$. Indicated inside each vertex v is $v.J$. Algorithm 2 examines only those vertices with $|v.J| >= 2$. (a) A prefix tree representing the power set 2^T, where $T = \{1, 2, 3, 4\}$. (b) Assuming $v.I = \emptyset$ for the top most vertex with $v.J = \{1, 2\}$, the subtree rooted at the vertex v can be removed. (c) Reduced prefix tree.

Several remarks on Algorithm 6.2 are in order. First, the algorithm does not maintain the prefix tree in its entirety. Only a part of the subtree is constructed at a time and removed after its use. To emphasize this, the procedure used in Algorithm 6.2 was termed "*Construct*Trie" rather than "*Traverse*Trie." Second, multiple instances of $v.I$ can be produced in Line 15, since $p.I$ and I_{jk} in this line are not necessarily unique.

6.2.6 Remarks

As already stated in previous chapters, the problem of co-clustering is inherently intractable, and the worst-case complexity of our method is exponential in $|T|$, the total number of samples. However, the response time of our method was practical in all the cases we tested. Furthermore, given input matrices A and B, our method can find all co-clusters that satisfy specific input parameters. If desired, the users can also define a criterion to further select co-clusters of specific interest.

Our method provides both a list of co-clusters found in the correlation matrix and

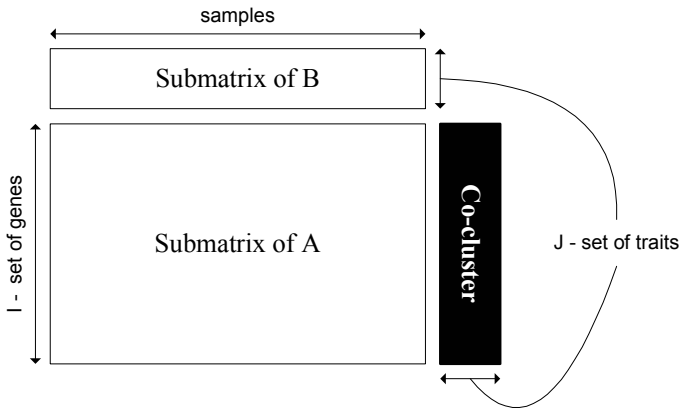


Figure 6.9: Composition of each images in Figure 6.10(d). Each figure is composed of three panels (heat maps). The panel in the middle corresponds to the submatrix (I, S) of the microarray matrix A, where $I \subseteq G$. The panel at the top corresponds to the submatrix (J, S) of the trait matrix B, where $J \subseteq T$. The right panel represents a co-cluster or a submatrix (I, J) of the correlation matrix C. The colored bars at the left of the middle panel indicate those genes from the gene groups C–H labeled in Figure 6.10(b).

the graphical images of these co-clusters. Figure 6.9 explains how these graphical images should be read. Section 6.3 will present some examples of co-clusters and their images. For example, Figure 6.10(d) shows some co-clusters discovered from the correlation matrix in Figure 6.10(c), which was constructed from the data in Figures 6.10(a) and 6.10(b).

6.3 Experimental results

6.3.1 Experiment procedure

We remind the reader of some notation that has been introduced so far. The input of our method are (1) $A = (G, S)$, a gene expression matrix with gene set G and sample set S; (2) $B = (T, S)$, a clinical parameter matrix with trait set S and sample set S; (3) τ and π, algorithm parameters.

We tested our method with data from an *acute myelogenous leukemia* (AML) study [17]. The AML data set used includes two matrices. One is a gene expression

Table 6.2: The input parameters used for the experiment and some statistics obtained from the output co-clusters.

Parameters/statistic	Value/reference
A: gene expression matrix	[17]
B: trait matrix	[17]
τ: parameter for Algorithm 1	2.5
π: p-value cutoff	0.05
Total number of co-clusters found	43
Average size of co-clusters $(\#genes, \#traits)$	$(143, 3)$

data matrix with 6283 genes and 119 samples as shown in Figure 6.10(b). The other is a matrix of 15 clinical parameters measured from the same subjects as seen in Figure 6.10(a). The description of these clinical parameters can be found in the caption of Figure 6.11.

We then used the procedure described in Section 6.2.2 to produce the correlation matrix[3] presented in Figure 6.10(c). In particular, we performed the *significance analysis of microarrays* (SAM) [102] on each clinical parameter over all the genes. Figure 6.11 shows the plots obtained by the SAM package[4]. The observed scores in the k-th SAM plot corresponds to the values on the k-th column of the correlation matrix. More details can be found in the caption of Figure 6.11.

We implemented our algorithm to find co-clusters in the correlation matrix in ANSI C++ on a 3.02 GHz Linux machine with 4 GB RAM. The implementation was invoked with the parameters listed in Table 6.2. The running time was in the order of minutes.

6.3.2 Results and discussion

We identified 43 co-clusters from the AML data set. Figure 6.10(d) shows some of the co-clusters found. Refer to Figure 6.9 for how to read the images in Figure 6.10(d).

[3]This correlation matrix has 14 columns instead of 15, because the traits "Status" and "Overall survival" were merged into one for the convenience in survival analysis.

[4]`http://www-stat.stanford.edu/~tibs/SAM/`

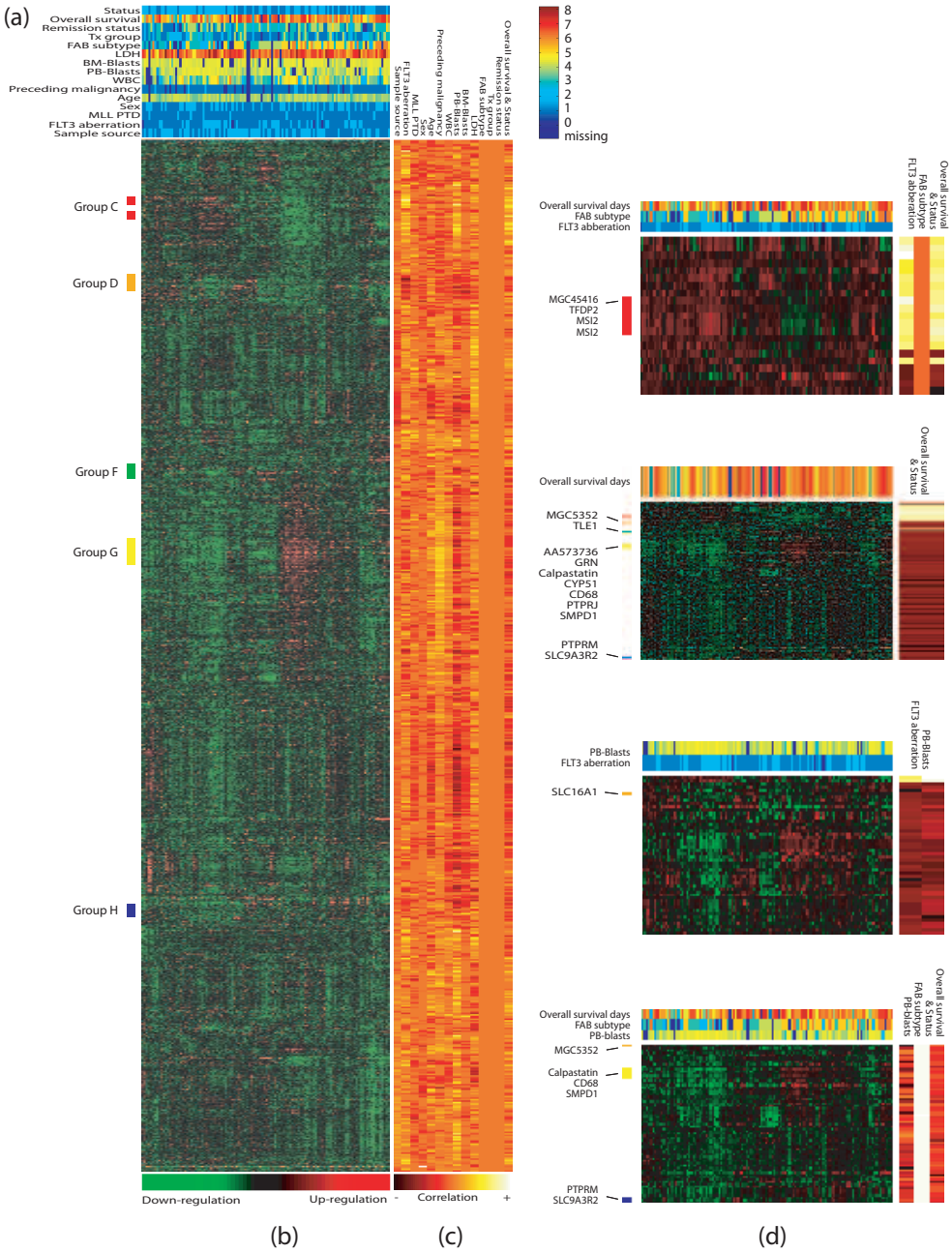


Figure 6.10: Data from an adult acute myeloid leukemia (AML) study [17]. (a) The heat map of the clinical trait matrix, in which each row corresponds to a trait and each column a sample. The legend of the heat map can also be found. (b) The heat map of the gene expression matrix with 6283 genes (rows) and 119 samples (columns). The vertical colored bars are to indicate the gene groups C–H used in the original study [17]. (c) The heat map of the correlation matrix. (d) Some co-clusters found by the proposed method. Refer to Figure 6.9 for further details.

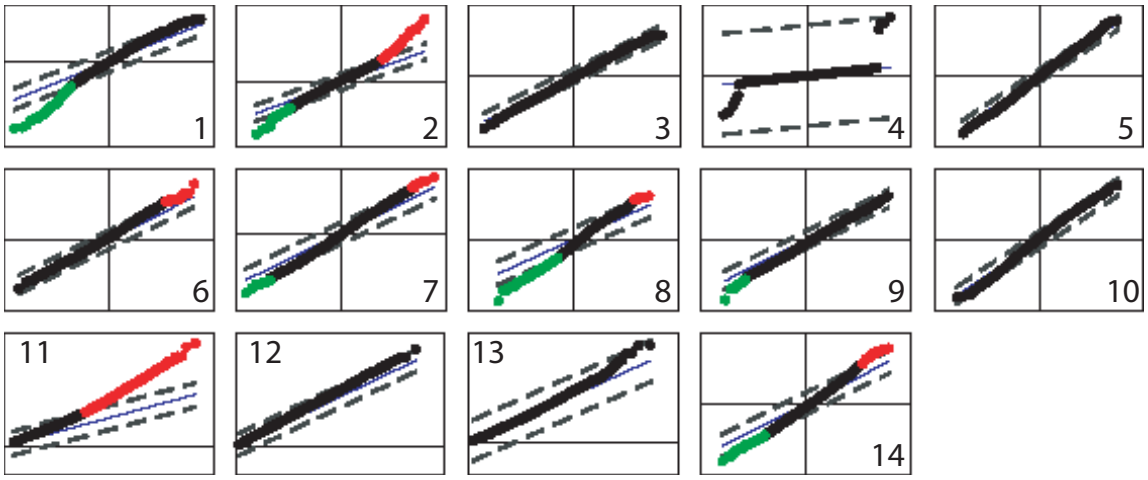


Figure 6.11: SAM plots obtained from the AML data set. The x and y axes of each plot represent the expected and the observed values of the SAM score (Equation 6.1), respectively. Further details on reading SAM plots can be found in [102]. The observed scores in the k-th SAM plot corresponds to the values on the k-th column of the correlation matrix. The number in each plot is the identifiers of the clinical traits used in [17]. [1: sample source, 2: FLT3 aberration, 3: MLL PTD, 4: sex, 5: age, 6: preceding malignancy, 7: WBC, 8: PB-Blasts, 9: BM-Blasts, 10: LDH, 11: FAB subtype, 12: Tx group, 13: remission status, 14: overall survival time and status.]

For each co-cluster (I, J) discovered ($I \subseteq G, J \subseteq T$), it is possible to pose a hypothesis of the form "genes $g \in I$ are correlated with traits $t \in J$", which can then be tested by further experimental studies.

Supporting evidence from the literature

Our data showed that trait "survival" is clustered with genes *TGFB1* or *TGFB2* and *CD1a* multiple times in co-clusters #37, #38, #42 and #43.

TGF-β (transforming growth factor-β) is a multifunctional peptide that has both growth-inhibitory and growth-stimulating properties [48]. Its combined effects with other growth factors or inhibitors have been shown to play a central role in the control of growth, differentiation, and morphogenesis of normal and malignant cells. For example, TGF-β is required for efficient *in vitro* generation of *dendritic cells* (DCs) from CD34+ progenitor cells [81]. However, it also inhibits cell proliferation and survival mediated by the Flt3 (Fms-like tyrosine kinase-3) signaling pathway [44,103]. Given that a mutated and constitutively active form of Flt3 is detected in 30-35 % of AML cases and the patients with Flt3 mutations tend to have a poor prognosis [108], it is interesting to note that "survival" trait is positively correlated with the expression of *TGFB1* and *TGFB2* which abrogate the effects of Flt3.

In addition, our data indicate that "survival" is associated with the expression of *CD1a*, a cell surface marker for mature DC. Previous studies reported that, when cultured in the presence of GM-CSF (granulocyte-macrophage colony-stimulating factor), TNF-α (tumor necrosis factor-α), and IL-4 (interleukin-4), AML cells were induced to differentiate into DCs and up-regulated the expression of *CD1a* and co-stimulatory molecules such as CD80 and CD86 [23,38]. Since DCs are the most potent *antigen-presenting cells* (APCs) in the immune system, the *CD1a*-positive leukemic DCs might function as APCs bearing leukemic antigens, hence priming cytotoxic T cells and generating a strong anti-leukemic immune response. This may explain why *CD1a* is often clustered with the trait "survival" in our data.

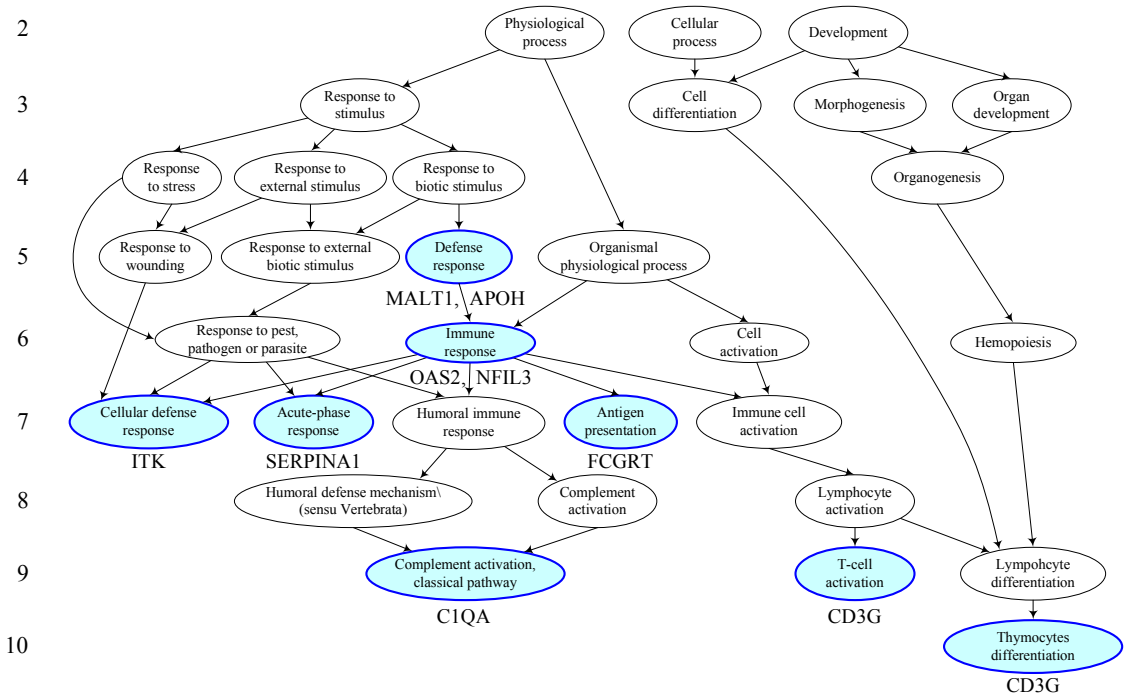


Figure 6.12: Terms from *Process Ontology* that annotate the genes in co-cluster #15 with p-values less than 0.05. The tool GO::TermFinder [13] was used to calculate p-values with multiple comparison correction as well as false discovery rate (FDR) calculation. The descriptions of the genes included in co-cluster #15 are listed in Table 6.3. Further analyses of the enriched terms are possible. For example, see Table 6.4 for more statistics for the term *defense response* (GO:0006952).

Validation with GO

The *gene ontology* (GO) is a collaborative effort to address the need for consistent descriptions of gene products in different databases [100]. GO consists of a trio of controlled vocabularies that are being developed to aid the description of the molecular functions of gene products, their placement in and as cellular components, and their participation in biological processes. Terms in each of the vocabularies are related to one another within a vocabulary in a poly-hierarchical (or directed acyclic graph) manner. Terms are mutually exclusive across the three vocabularies.

To determine whether any GO terms annotate genes in a specified co-cluster at a frequency greater than that would be expected by chance, a p-value is calculated in this particular setting using the hypergeometric distribution:

$$p\text{-value} = 1 - \sum_{i=0}^{k-1} \frac{\binom{M}{i}\binom{N-M}{n-i}}{\binom{N}{i}}, \tag{6.14}$$

where N is the total number of genes in the background distribution, M is the number of genes (within that distribution) that are annotated to the node of interest, n is the size of the list of genes in a co-cluster of interest, and k is the number of genes within that list which are annotated to the node. The background distribution is all the genes within a given GO annotation file. We used the tool GO::TermFinder [13] for the calculation of p-values as well as the multiple hypothesis correction [29] of the calculated p-values.

For example, Figure 6.12 shows a subgraph of Process Ontology, which includes the terms that annotate genes in co-cluster #15 with p-values less than a threshold of 0.05. The descriptions of the genes included in co-cluster #15 are listed in Table 6.3. Further analyses of the enriched terms are also possible, and as an example, Table 6.4 shows more statistics for the term *defense response* (GO:0006952) obtained from the tool used.

Table 6.3: Some genes contained in co-cluster #15 with descriptions.

Gene	Description
MALT1	MAL tissue lymphoma translocation gene 1
NFIL3	Nuclear factor, interleukin 3 regulated
APOH	Apolipoprotein H (beta-2-glycoprotein I)
FCGRT	Fc fragment of IgG, receptor, transporter, alpha
SERPINA1	Serine (or cysteine) proteinase inhibitor
C1QA	Complement component 1, q subcomponent, alpha
OAS2	2'-5'-oligoadenylate synthetase 2
ITK	IL2-inducible T-cell kinase
CD3G	CD3G antigen, gamma polypeptide (TiT3 complex)

Table 6.4: Further details on an enriched GO term in Figure 6.12, obtained by the tool GO::TermFinder [13]. The total number of genes used to calculate the background distribution of GO terms is 23531.

Item	Value
GO term	Defense response
Cluster frequency	9 out of 54 genes (16.7%)
Genome frequency of use	1209 out of 23531 genes (5.1%)
Corrected p-value	0.0359
False Discovery Rate (FDR)	3.00%
False Positives	0.06

6.4 Summary

We investigated the problem of finding co-clusters of genes and clinical traits using microarray data and clinical parameter information. An intermediate data matrix called correlation matrix was computed by means of a statistical method. We then modeled a co-cluster by a submatrix of the correlation matrix with coherent elements and aimed at finding statistically significant co-clusters. We describe a co-clustering algorithm, which is a special case of the nested bicluster mining algorithm in Chapter 4. We tested this co-clustering algorithm with the AML data set and discovered a number of co-clusters of clinical traits and possibly related genes. The validation with GO as well as the literature suggests that some of these co-clusters are biologically meaningful. Extension of this technique may provide an effective methodology for finding genes responsible for any set of clinical parameters of interest, which can lead to a major impact on clinical diagnosis. Since the proposed technique allows us to trace the relationship between genes with respect to some clinical traits, this technique can also provide useful information for reconstructing gene regulatory networks.

Chapter 7

Prediction of MicroRNA Regulatory Modules

This chapter covers an application of the biclustering method described in Chapters 3 and 4 to the problem of predicting microRNA regulatory modules.

7.1 Introduction

MicroRNAs (miRNAs) are endogenous 21-22-nucleotide RNAs that can play crucial regulatory roles in animals and plants by targeting transcripts for cleavage or translational repression [8]. Hundreds of different miRNAs have now been identified in complex eukaryotes, implying that they mediate a vast network of unappreciated regulatory interactions [54].

Computational methods have been applied to the studies of miRNAs largely in two ways. First, techniques to identify miRNA host genes have been proposed [55,60, 75,82]. These methods rely upon the observation that miRNAs generally derive from phylogenetically conserved stem-loop precursor RNAs with characteristic features. Second, given that miRNA target gene selection is guided by the sequence, algorithms have been suggested to systematically identify miRNA targets *in silico* [31,45,49,58, 77,79,92,94].

Typically, multiple miRNAs regulate one message, reflecting cooperative translational control. Conversely, one miRNA may have several target genes, indicative of target multiplicity [31]. This multiplicity of targets and cooperative signal integration on target genes are key features of the control of translation by miRNAs [45]. However, this many-to-many relationship between miRNAs and target genes is often complicated (e.g., see Figure 7.6(c)), and we thus need an automated analysis tool.

In this chapter, we mathematically formulate the biological observations of the interactions of miRNAs and their targets and present a way to identify important patterns hidden in the complex interactions. In particular, we propose a computational method to predict *miRNA regulatory modules* (MRMs) or groups of miRNAs and their targets that are believed to participate cooperatively in post-transcriptional gene regulation. MRMs are similar to Type 1 patterns introduced in Chapter 4 and can thus be found by the biclustering algorithm described in that chapter.

We apply our method to the prediction of human miRNA regulatory modules and here report a predicted module that contains the genes: *BTG2*, *WT1*, *PPM1D*, *PAK7* and *RAB9B*, and the miRNAs: *miR-15a* and *miR-16*. As will be detailed later, it has been reported that these genes are mostly regulators and their anomaly can be found in breast, renal, and prostate cancers [34, 47, 96]. Interestingly, *BTG2*, *WT1*, and *PPM1D* have been shown to be directly associated with the function of *p53*, a tumor suppressor gene [104]. Furthermore, the human miRNAs *miR-15a* and *miR-16* are clustered on chromosome 13q14, and this region has been shown to be deleted altogether in several types of cancer [59, 86]. The annotation of this module with the terms in the database Gene Ontology (GO) [100] also suggests that the genes in this module indeed share some common roles in biological processes.

The miRNA regulatory modules predicted can further be useful in some important tasks including the reconstruction of gene regulatory networks as well as the biological validation of miRNA-target duplexes. Specifically, the regulatory interactions newly revealed by MRMs may provide a missing piece in the puzzle of gene regulation mechanisms, enabling us to reverse-engineer more accurate gene regulatory networks. In addition, the genes included in MRMs can be reasonable candidates for the experimental validation of miRNA targets, since these genes are detected multiple times by

distinct miRNAs. Focusing on the genes included in MRMs may be an effective way to design an experiment for target validation.

The remainder of this chapter is organized as follows. Section 7.2 formally defines miRNA regulatory modules and presents our approach to predict them. Section 7.3 provides the details of our analysis of a predicted module through a literature review and annotation with GO.

7.2 Method

Our method consists of five major steps, each of which will be detailed in this section.

1. Target identification: given a set of miRNAs, their target genes are identified (Section 7.2.1).

2. Relation graph representation: the relation between miRNAs and their targets are represented by a weighted bipartite graph called *relation graph* (Section 7.2.2).

3. Seed finding: a *seed* or a set of miRNAs that bind a common target with similar binding strength is identified (Section 7.2.3). A seed corresponds to the Type 1 atomic nested bicluster described in Chapter 4 and can be found by an approach similar to Algorithm 4.1.

4. Merging seeds to find candidate modules: the seeds found in the previous step are collected and merged to produce candidates for miRNA regulatory modules (Section 7.2.4). This step is a simplified version of Algorithm 3.2 in Chapter 3.

5. Post-processing: statistically significant miRNA regulatory modules are selected by computing the p-value or the probability of finding a module by chance [93] (Section 7.2.5).

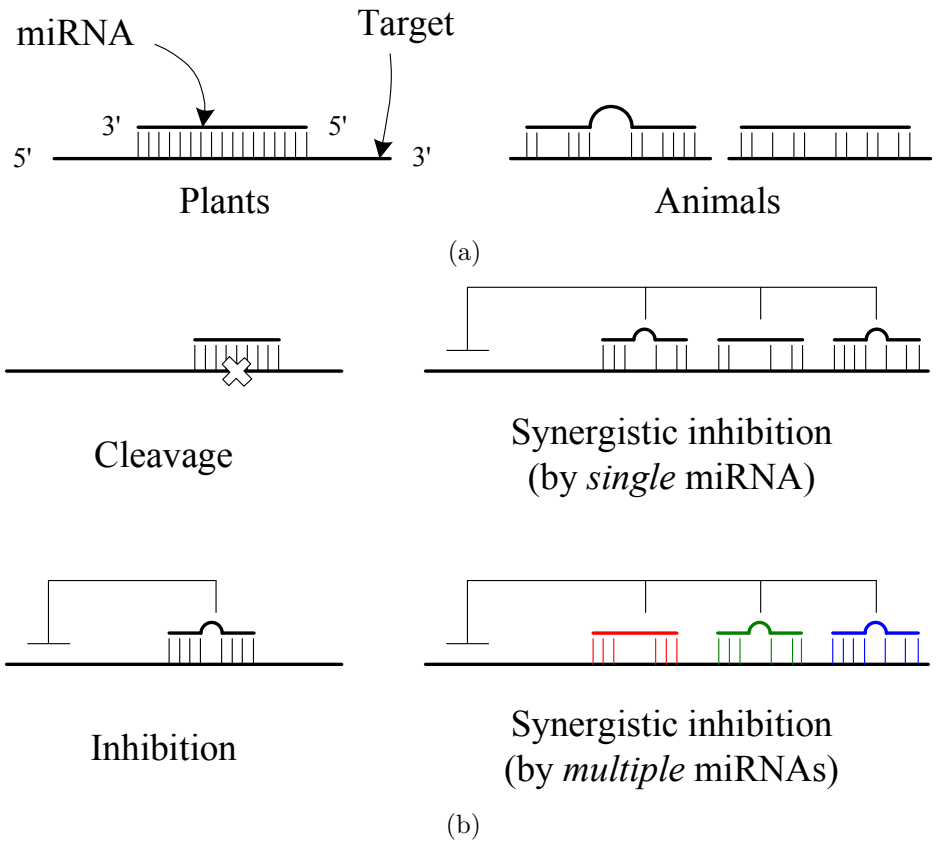


Figure 7.1: MicroRNAs and targets [54]. (a) Plant miRNAs exhibit extensive complementarity to their targets, but animal miRNAs generally do not. (b) Various configurations for miRNA-target duplexes: one near-perfect binding site for one miRNA (upper left), one strong site for one miRNA (lower left), multiple 'modest' sites for one miRNA (upper right), and multiple 'modest' sites for multiple miRNAs (lower right).

7.2.1 Identification of miRNA target sites

Target selection is guided by the miRNA sequence, as informally shown in Figure 7.1 [54]. In *plants*, probable targets of most miRNAs can be found simply by searching for highly complementary sequences in mRNA coding sequences or *untranslated regions* (UTRs). In contrast, *animal* miRNAs do not generally exhibit extensive complementarity to any endogenous transcripts (Figure 7.1(a)). Various configurations for miRNA-target duplexes are possible, as presented in Figure 7.1(b). In particular, when multiple binding sites exist on a target, the strength of each binding is not too strong or weak but modest and similar, according to Lai (2004, page 115.2). This observation will be reflected in our mathematical formulation in Section 7.2.2.

We first brief the reader on the existing target identification techniques, upon which the first step of our method depends. One stream of the algorithms to identify animal miRNA targets relies on three properties: (i) sequence complementarity using a position-weighted local alignment algorithm, (ii) free energies of miRNA-target duplexes, and (iii) evolutionary conservation of target sites in homologous genes. In particular, the *conservation filter* tends to be the most predictive criterion for accurate target detection [31]. The complementarity displayed by a miRNA and its binding site is usually not enough to be statistically significant, since a miRNA is only 21-22 nucleotides long. Thus, this conservation filter plays a crucial role in reducing the number of false positives.

Recently, another type of target identification algorithm [57] was proposed. This algorithm works by identifying mRNA targets with conserved complementarity to the nucleotides 2–7 of the miRNA. This method does not consider other criteria. What was found by this approach was that an overrepresentation of conserved adenosines (flanking the mRNA sites that are complementary to the nucleotides 2–7 of the miRNA) indicates that primary sequence determinants can supplement base pairing to specify miRNA target recognition. This method uses simplified detection rules and potentially predicts more miRNA targets than previous methods.

In the first step of our method, we identify miRNA-mRNA duplexes by the method

described in [58][1] and [45]. (Other methods can also be used as long as they can quantify the strength of miRNA-target binding, as is usually the case.) We refer to the local alignment score and the free energy of a miRNA-target duplex as s_A and s_E, respectively. The scores s_A and s_E are (negatively) correlated in most cases, because a duplex with a high local alignment score tends to have a low free energy and vice versa.

7.2.2 Relation graph representation

In the second step of our method, we represent the many-to-many relation between miRNAs and target genes by a weighted bipartite graph termed *relation graph*.

Definition 7.1. *Let M denote a set of miRNAs and T a set of targets (typically $|M| \ll |T|$). The* relation graph *is a weighted bipartite graph $G = (V, E, w)$ with the vertex set $V = M \cup T$, the edge set $E = \{\{m, t\} | \text{miRNA } m \in M \text{ binds target } t \in T\}$, and the weight function $w : E \to \mathbb{R}$.*

We determine the weight function w by performing *principal component analysis* (PCA) [46] on the space spanned by s_A and s_E. After making the populations of s_A and s_E have a zero mean, we find the unit vector u so that when the data is projected onto the direction corresponding to u the variance of the projected data is maximized. This unit vector u is equivalent to the principal eigenvector of Σ, the empirical covariance matrix of the data, defined as

$$\Sigma = \frac{1}{N} \sum_{i=1}^{N} \left(\begin{bmatrix} s_A \\ s_E \end{bmatrix}_i \cdot \begin{bmatrix} s_A \\ s_E \end{bmatrix}_i^{\mathsf{T}} \right), \tag{7.1}$$

where N represents the number of edges, $\begin{bmatrix} s_A \\ s_E \end{bmatrix}_i$ is a score vector for the i-th edge, and $^{\mathsf{T}}$ means the transpose operator. Finally, for each $e \in E$, its weight $w(e)$ is calculated

[1] According to our experiments, the set of target genes identified by the method described in [57] is typically a superset of the set of target genes predicted by the earlier method [58] from the same authors.

as the projection of a score vector onto u, namely,

$$w(e) = \begin{bmatrix} s_A \\ s_E \end{bmatrix}^{\mathsf{T}} u. \tag{7.2}$$

Modeling MRMs

We model the miRNA regulatory module by a *biclique* or a complete subgraph in a bipartite graph [2]. In particular, we search only those bicliques in which, for each target vertex t, the edges incident on t have similar weights, following the biological observation explained in Section 7.2.1. To avoid redundancy, we find only *maximal* bicliques that are not contained by other bicliques as a proper subgraph.

Definition 7.2. *For set A on $\mathbb{R}$, $range(A)$ denotes the difference between the largest and the smallest elements of A.*

Definition 7.3. *Let $G = (M \cup T, E, w)$ be the relation graph and $\delta \geq 0$ be given as a parameter. Graph $G' = (M' \cup T', E', w)$ is called a* miRNA regulatory module *(MRM), if G' is a maximal biclique in G, and for each $t \in T'$, $range(\{w | w = w(\{m, t\}),\ \forall m \in M'\}) \leq \delta$.*

Example 7.1. *Figure 7.2 shows an example of the relation graph and an MRM found in this relation graph.*

7.2.3 Finding seeds

The third step of our method is to find *seeds* for each predicted target gene. A seed is similar to the Type 1 atomic nested bicluster described in Chapter 4. Thus, an approach similar to Algorithm 4.1 can be taken here by switching the role of rows and columns in Algorithm 4.1.

Definition 7.4. *Let t be a target gene and M_t be a set of miRNAs that binds the target gene t. A* seed *for t, denoted by $S(t)$, is a subset of M_t such that (i) $range(S(t)) \leq \delta$, and (ii) there is no $M' \supset S(t)$ such that $M' \subseteq M_t$ and $range(M') \leq \delta$.*

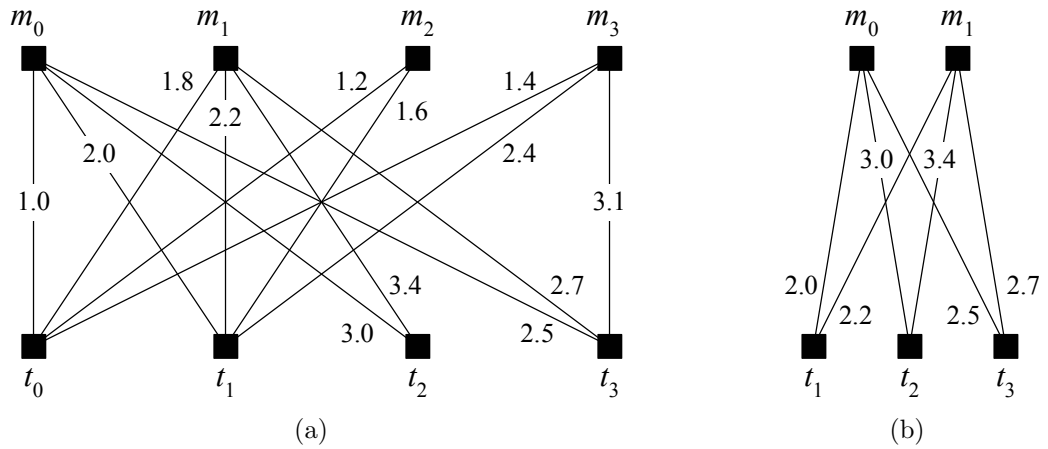


Figure 7.2: (a) Example relation graph $G = (M \cup T, E, w)$, where $M = \{m_0, m_1, m_2, m_3\}$, and $T = \{t_0, t_1, t_2, t_3\}$ with some hypothetical weights. (b) An MRM found in G with the parameter $\delta = 0.5$.

Algorithm 7.1 presents our approach to generate a seed set for a given target transcript. This algorithm closely resembles Algorithm 4.1. Algorithm 7.1 takes as input a target gene and a set of all the miRNAs binding the target gene regardless of binding strength. The output is a seed for the target gene or a maximal set of miRNAs whose binding strength to the target gene is similar in the sense that the difference between the maximum and the minimum strength is less than given δ.

The key idea of Algorithm 7.1 is simple: when the elements of set A are sorted and arranged in the corresponding order, $range(A)$ is simply the absolute difference between the first and the last elements of A.

In **Lines 1–6**, miRNAs are sorted in ascending order by their binding strength. The variables *begin* and *end* in **Lines 7–8** are to point to the first and the last elements of the sub-array under consideration at some point. The set of seeds $\mathcal{S}$, which is to be returned as output, is initialized in **Line 9**. Inside the while loop in **Lines 10–21**, seeds are generated as the variables *begin* and *end* are incremented. Since the miRNAs are sorted, we only need to compare the first element ($s[begin]$) and the last element ($s[end]$), as is done in Line 11, in order to see if all the elements in the sub-array are similar. **In Lines 11–12**, the variable *end* is extended as long as $s[end].w - s[begin].w \leq \delta$. A seed is found and collected in **Lines 14–15** and **Lines**

Algorithm 7.1: Find a seed for each target gene

input : t, a target transcript
input : M_t, a set of all miRNAs binding t
input : δ, a threshold
output: $\{S(t)\}$, a set of seeds

```
1  i := 0;
2  foreach m ∈ M_t do
3      s[i].w := w(t, m);
4      s[i].id := m;
5      i := i + 1;
6  sort array s in ascending order with respect to the w field;
7  begin := 0;
8  end := 1;
9  S = ∅;
10 while (end < |M_t|) do
11     if (s[end].w − s[begin].w ≤ δ) then
12         end := end + 1;
13         if (end = |M_t|) then
14             S := GetOneSeed(begin, end, s);
15             S := S ∪ {S};
16     else
17         S := GetOneSeed(begin, end, s);
18         S := S ∪ {S};
19         repeat
20             begin := begin + 1;
21         until (begin = end) or (s[end].w − s[begin].w ≤ δ);
22 return S;
23 procedure GetOneSeed(begin, end, s)
24 begin
25     S := ∅;
26     for i = begin to end − 1 do
27         S := S ∪ {s[i].id};
28     return S;
29 end
```

Figure 7.3: Algorithm to find seeds for each target

Table 7.1: The seeds generated by Algorithm 1 from the relation graph in Figure 7.2(a) with the parameter $\delta = 0.5$.

t: target gene	$S(t)$: seed for target gene t	# seeds
t_0	$\{m_0, m_2, m_3\}$, $\{m_1, m_3\}$	2
t_1	$\{m_0, m_1, m_3\}$, $\{m_0, m_2\}$	2
t_2	$\{m_0, m_1\}$	1
t_3	$\{m_0, m_1\}$, $\{m_1, m_3\}$	2

17–18. **Lines 19–21** are to adjust the variable *begin* after a seed is found.

Note that multiple seeds can exist for a single target gene, and thus a set of all the seeds for the given target gene is returned as output. Also notice that two distinct seeds for the same target gene can overlap. The worst-case complexity of the algorithm is polynomial in $|M_t|$.

Example 7.2. *Table 7.1 shows all the seeds generated by Algorithm 7.1 from the relation graph in Figure 7.2(a) with the parameter* $\delta = 0.5$.

Related data mining tasks

Before describing the next step of our method, we show how the process of finding miRNA regulatory modules is related to several data mining techniques, in order to put the description in proper context.

First, the problem of *frequent itemset mining* [1] is to find a group of items that occur together frequently in a database. Formally, let I be a set of all items in database D. A set, I', is called an *itemset* if $I' \subseteq I$. A *transaction* is pair (tid, I'), where tid is the transaction identifier and I' is an itemset. The transaction (tid, I') is said to *support* itemset I_s if $I_s \subseteq I'$. The *cover* of an itemset is the set of the identifiers of transactions that support the itemset. That is, for itemset I_s,

$$cover(I_s) = \{tid | (tid, I') \in D, I_s \subseteq I'\}. \tag{7.3}$$

The itemset I_s is called *frequent* if $|cover(I_s)| \geq \beta$, where β is a given threshold.

In the current problem, the set of miRNAs and the set of targets forming an MRM are similar to a frequent itemset and its cover, respectively. One difference is that a target can have multiple seeds whereas a transaction has only one itemset in a typical setup.

The present problem is also related to the problem of biclustering. Any matrix can be converted to a weighted bipartite graph, and vice versa. Thus, the relation graph $G = (M \cup T, E, w)$ can be converted to a matrix of weights with the row set M and the column set T. Then, a miRNA regulatory module is similar to a bicluster with constant values on columns. In other words, miRNA regulatory modules can be represented by Type 1 nested biclusters defined in Chapter 4 by switching the role of rows and columns.

7.2.4 Deriving MRMs from seeds

The fourth step of our method is to collect all the seeds and derive MRMs from the seed collection. As previously stated, this step corresponds to the second step of the biclustering process described in Chapter 3. Thus, to collect seeds in a systematic and effective manner, we exploit a *trie*, a compact data structure used to represent sets of character strings [2]. Many overlaps often occur between the seeds, and a trie can provide compact representations. The seeds stored in the nodes of the trie are then merged to form MRMs as the trie is traversed.

Algorithm 7.2 details our approach. In addition to the seeds found by Algorithm 7.1, the algorithm takes as input two parameters, min_T and min_M, to specify the minimum size of MRMs to find. Note that this algorithm is similar to Algorithm 3.2. One difference is that the first step of Algorithm 3.2 is not needed in Algorithm 7.2, since here we search for Type 1 biclusters (with the role of rows and columns switched).

In **Lines 2–6**, each seed is inserted into a trie. To decide the location of the node into which a seed is inserted, we first assume a total order among the elements of M (the set of all miRNAs in Definition 7.1), of which every seed is a subset. For each seed $S(t)$ of target t, we then sort its elements with respect to the total order. The

Algorithm 7.2: Find miRNA regulatory modules from the seeds

input : All the seeds generated by Algorithm 7.1
input : min_T, the minimum number of target genes in MRMs
input : min_M, the minimum number of miRNAs in MRMs
output: miRNA regulatory modules

```
/* Represent seeds by a trie */
foreach seed S(t) do
    Sort the elements in S(t);
    n := the node whose path is specified by (sorted) S(t);
    n.T := n.T ∪ {t};
    n.S := S(t);
/* Merge the seeds */
foreach node n in the post-order traversal of the trie do
    foreach node n' s.t. |n'.S| = |n.S| − 1 ≥ min_M do
        n'.T := n'.T ∪ n.T;
/* Prune the trie and collect candidates */
foreach node n in the pre-order traversal of the trie do
    if |n.S| ≥ min_M then
        if |n.T| < min_T then
            Remove n and its subtrie rooted at n;
        else
            Collect (n.T, n.S) as a candidate MRM;
Return maximal candidates as MRMs;
```

Figure 7.4: Algorithm to find miRNA regulatory modules.

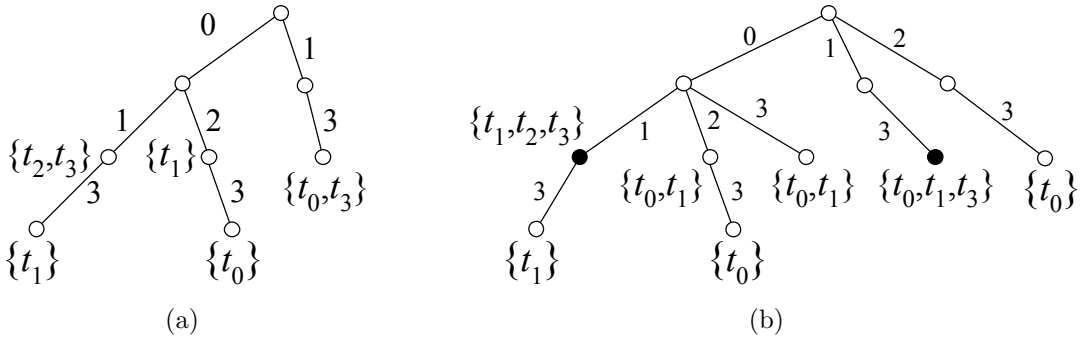


Figure 7.5: (a) The trie representation of the seeds in Table 7.1. The edge labeled i represents miRNA m_i. (b) The seeds merged by Algorithm 2 with the parameters $min_T = 3$, and $min_M = 2$. The solid-circled vertices represent a candidate for MRMs.

sorted seed can now be inserted into the node whose path is specified by the ordered elements.

To keep track of the seeds and associated target genes represented by the trie efficiently, two sets $n.S$ and $n.T$ are associated with each node n, as seen in **Lines 5–6**. Suppose that $S(t)$, a seed for target t, is inserted into node n. Then the set $n.S$ stores $S(t)$ proper, and the set $n.T$ contains the target gene t. Later in Line 10, the set $n.T$ will be expanded in such a way that $n.T = \{\tau \in T | n.S \subseteq S(\tau)\}$.

Example 7.3. *The trie in Figure 7.5(a) collectively represents the seeds in Table 7.1.*

In **Lines 8–10**, the algorithm expands the trie to systematically merge the seeds and find candidates for MRMs. For each node n encountered in the *post-order* traversal of the trie, the set $n.T$ is distributed to every node n' in which $|n'.M| = |n.M| - 1$ and $|n'.M| \geq min_M$. The node n' is a node such that the number of elements in $n'.S$ is one smaller than n but not less than min_M.

In **Lines 14–15**, every node n in which $|n.T| < min_T$ is deleted. This step can be performed efficiently by a *pre-order* traversal of the trie. Target genes were distributed in post-order in Lines 8–10. Consequently, node n in the trie always has a superset of the genes its children have. Thus, if the node n has less than min_T target genes, then none of its children can have more. For this reason, we can safely remove the entire subtree whose root is located at the node n without visiting its child nodes.

Table 7.2: miRNA regulatory modules predicted with the parameters $\delta = 0.5$, $min_T = 3$, and $min_M = 2$.

MRM #	Targets in the module	miRNAs in the module
1	$\{t_1, t_2, t_3\}$	$\{m_0, m_1\}$
2	$\{t_0, t_1, t_3\}$	$\{m_1, m_3\}$

In **Lines 17–18**, candidates for MRMs are collected, and the maximal ones are returned as MRMs.

Example 7.4. *Figure 7.5(b) shows the trie after the seeds have been merged. Table 7.2 lists two candidate MRMs predicted from our running example.*

The problem of enumerating maximal bicliques is inherently intractable [65], and the worst-case complexity of Algorithm 2 is exponential in the number of miRNAs in the relation graph. However, the execution time of the algorithm on typical benchmarks is practical (see Section 7.3). This is because a seed seldom contains all the miRNAs in the relation graph, and the trie-based representation of seeds helps to prevent unnecessary enumeration of intermediate results.

7.2.5 Post-processing

Out of the miRNA regulatory modules found by Algorithm 7.2, we select those with a low p-value. We estimate the p-value of an MRM, or the probability of finding it by chance, on top of the statistical framework by [18]. They calculated the probability that a random submatrix of a gene expression data matrix has near-constant values on rows. They also reported that the distribution of the number of such matrices can be well approximated by the Poisson distribution. As previously stated, an MRM can be viewed as a matrix in which the values of each row are similar. Thus, we take advantage of the result by [18] with minor modifications in order to approximate the p-values of MRMs. More precise assessment of their statistical significance will be possible as more exact mechanisms of miRNA-target interaction are revealed.

We assume that the number of miRNA regulatory modules with m miRNAs and t targets in the relation graph is a Poisson random variable denoted by $X_{m \times t}$. That

is,

$$P(X_{m\times t} = k) = \frac{\lambda^k e^{-\lambda}}{k!},\ k = 0, 1, 2, \ldots \tag{7.4}$$

The parameter λ corresponds to the average number of the MRMs with m miRNAs and t targets in the relation graph, namely,

$$\lambda = \binom{|M|}{m}\binom{|T|}{t} P_{m\times t}, \tag{7.5}$$

where $P_{m\times t}$ is the probability that a random $(m \times t)$ biclique in the relation graph satisfies the condition to be a $(m \times t)$ MRM. Based upon the result by [18], $P_{m\times t}$ can be approximated by

$$P_{m\times t} \simeq \zeta^t [1-\zeta]^{|T|-t} [1-(1+m^{-1})^t \delta^t]^{|M|-m}, \tag{7.6}$$

where

$$\zeta = m\delta^{m-1} - (m-1)\delta^m. \tag{7.7}$$

The p-value of the MRM with m miRNAs and t targets is then defined to be the probability that one or more such MRMs occur by chance in the relation graph, namely,

$$P(X_{m\times t} \geq 1) = 1 - P(X_{m\times t} = 0) = 1 - e^{-\lambda}. \tag{7.8}$$

Finally, we choose those MRMs whose p-value computed by Equation (7.8) is less than a certain threshold, highlighting statistically significant modules.

7.3 Experimental results

We tested our method with the miRNAs and genes in *Homo sapiens* and predicted 431 miRNA regulatory modules. On average, an MRM consists of 3.58 miRNAs and 6.74 target genes.

Table 7.3: The parameters used for the experiment and some statistics obtained. († The standard deviation of the weight distribution in Figure 7.6(b).)

Parameters/statistic	Value/reference
Parameters (s_A cutoff, s_E cutoff)	$(91, -17$ kcal/mol)
Parameters (min_T, min_M, δ)	$(3, 3, 2\sigma^\dagger = 2.40)$
Size of the relation graph $(\|T\|, \|M\|, \|E\|)$	$(2888, 156, 7886)$
Weights in the relation graph	Figure 7.6
Total number of modules found	431
Average size of modules (# targets, # miRNAs)	$(6.74, 3.58)$

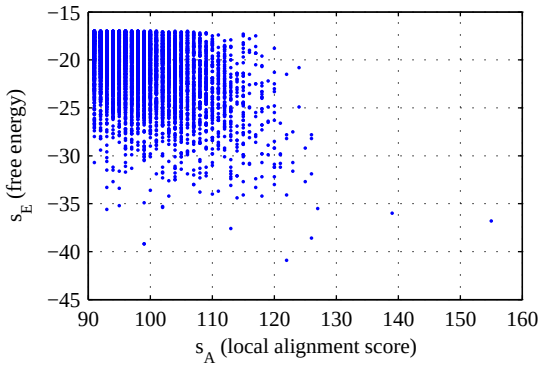
7.3.1 Experiment procedure

The input to our method was the human miRNA sequences[2] and the human gene sequences[3]. The output was a list of miRNA regulatory modules. The methods described in [58] and [45] were first used to identify 7,886 human miRNA-mRNA duplexes. 2,888 genes and 156 miRNAs were found to participate in forming a duplex (see Table 7.3). After scalar weights were calculated by Equation (7.2), the relation graph was constructed. Figure 7.6 shows the distributions of s_A and s_E and a matrix representation of the relation graph. Algorithms 1 and 2 were invoked with the parameters listed in Table 7.3. Statistically significant MRMs were selected with the p-value threshold of 0.01. The annotation of selected modules with the terms in Gene Ontology[4] was finally performed. The computation ran on a 3.06 GHz Linux machine with 4 GB RAM, and the response time for Algorithms 1 and 2 was in the order of minutes.

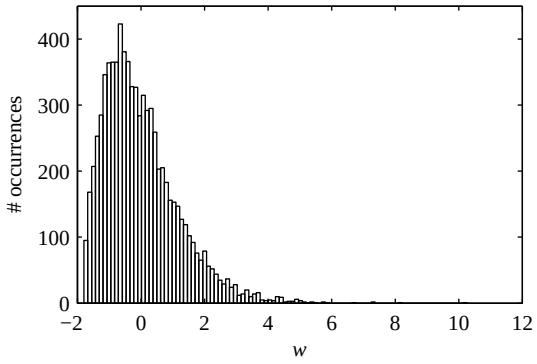
7.3.2 Prediction and analysis of an oncogenic module

From the 431 predicted modules, here we present a cancer-related module and analyze it at length in an attempt to reveal its biological implications.

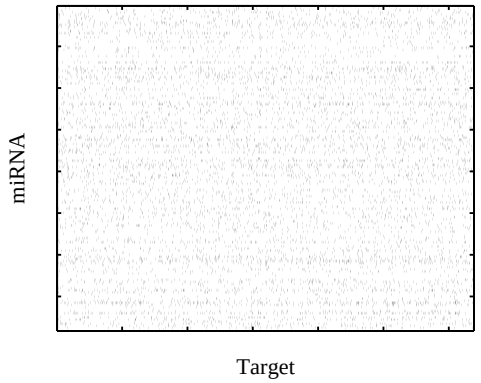
[2] `http://www.sanger.ac.uk/Software/Rfam/mirna`
[3] `http://www.ensembl.org/EnsMart`
[4] `http://www.geneontology.org`



(a)



(b)



(c)

Figure 7.6: (a) The distributions of the scores s_A and s_E. (b) The edge weight distribution with $\mu = 0$, $\sigma = 1.20$, $min = -1.79$ and $max = 10.26$. (c) The relation graph represented by a $156 \times 2,888$ matrix. A dot exists at row i and column j, if target j has a binding site for miRNA i. This plot visualizes the initial raw data set, clearly showing a need for an automated tool to identify important patterns underlying the complex interactions between miRNAs and targets.

Table 7.4: A predicted human MRM. The first column represents the genes in the module, and the last three columns show the miRNAs with their binding strength to each target in terms of the weight calculated by Equation (7.2). The parameters used are listed in Table 7.3. († Has not been experimentally verified in human.)

Target	Ensemble ID	Description	*miR-15a*	*miR-16*	*miR-195*†
PAK7	ENSG00000101349	p21-activated kinase 7	1.609	-0.789	0.676
RAB9B	ENSG00000123570	Ras-associated oncogenic protein 9b	1.303	-0.746	-0.956
BTG2	ENSG00000159388	B cell translocation gene 2	-0.162	-0.816	-1.259
PPM1D	ENSG00000170836	Protein phosphatase 1D Mg-dependent	-0.487	-0.817	-1.143
WT1	ENSG00000184937	Wilms' tumor	0.275	1.019	-0.514

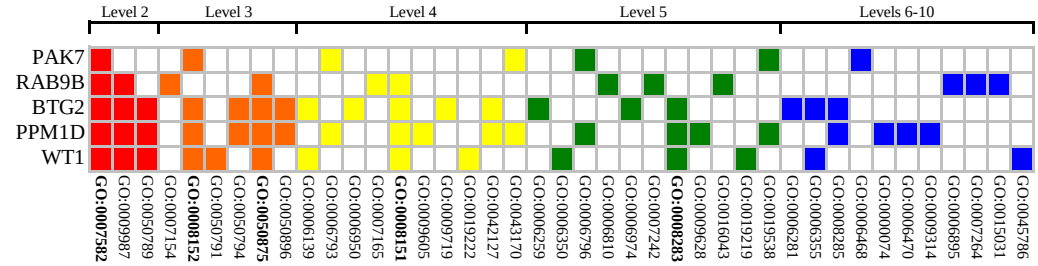
Our data shows that a set of genes *PAK7*, *BTG2*, *WT1*, *PPM1D*, and *RAB9B* are candidate targets for human *miR-15a*, *miR-16*, and *miR-195*. Table 7.4 lists more details of this module. In what follows, we consider *miR-15a* and *miR-16* only, since *miR-195* is a predicted miRNA based on homology to a verified miRNA from mouse [53], and the expression of this miRNA has not been verified in human.

Using Gene Ontology (GO) [100] has become a standard way to validate the functional coherence of genes in a list. Typically, this type of validation is accompanied by a statistical significance analysis.

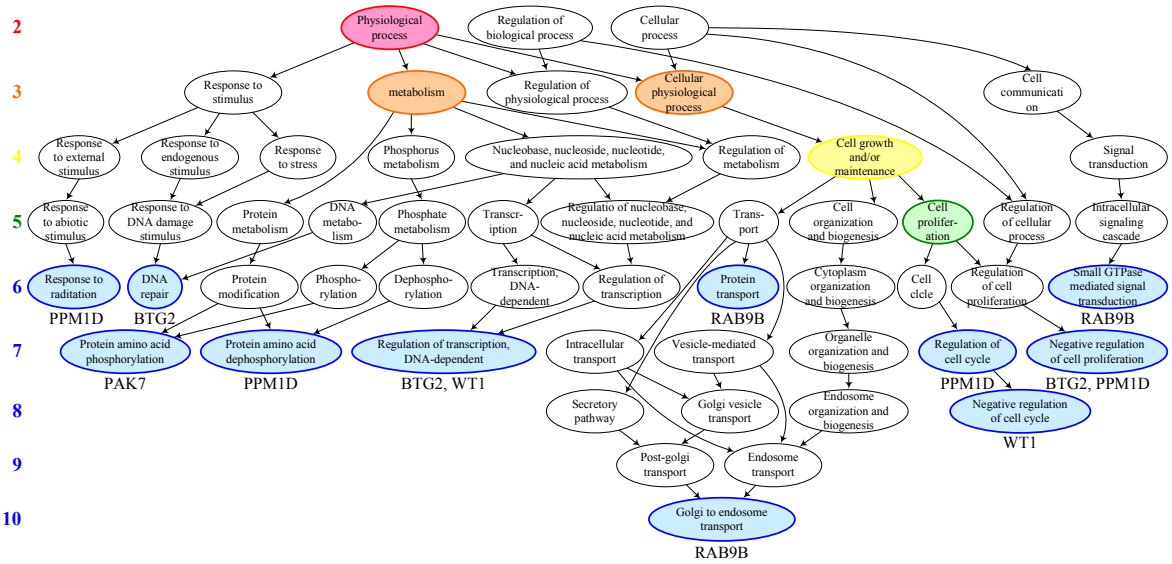
Figure 7.7 shows the annotation of the genes in this module with the terms in the *Biological Process* category of GO. In particular, Figure 7.7(a) shows the distribution of the GO terms over the genes, and Figure 7.7(b) presents how these terms are related in the GO dag[5]. We observe that the abundant terms include GO:0007582 (physiological process), GO:0008152 (metabolism), GO:0050875 (cellular physiological process), GO:0008151 (cell growth and/or maintenance), and GO:0008283 (cell proliferation).

Furthermore, we used the tool GO::TermFinder [13] to find significantly over-represented GO terms. This tool calculates a p-value relative to the hypergeometric distribution and also performs the multiple comparison correction. For example, Table 7.5 presents some more details on one of the enriched GO term shown in Figure 7.7(b).

[5]Each of the three ontology databases is represented by a *directed acyclic graph* (dag).



(a) Distribution of the GO terms over the genes in the module in Table 7.4



(b) A subgraph of the GO dag to show the relationship among the GO terms used

Figure 7.7: Annotation with GO terms. (a) Each row represents a target gene, and each column a GO term in Biological Process. A colored box exists at row i and column j if target i has GO term j. The abundant terms are GO:0007582 (physiological process), GO:0008152 (metabolism), GO:0050875 (cellular physiological process), GO:0008151 (cell growth and/or maintenance) and GO:0008283 (cell proliferation). (b) The blue vertices are for the terms in levels 6–10 associated with the targets in the predicted MRM. The ancestor vertices are also included, where the most abundant terms in each level are colored in red (level 2), orange (level 3), yellow (level 4), and green (level 5). Further analysis of each enriched GO term is possible. For example, Table 7.5 presents some detailed analysis of the term *negative regulation of cell proliferation*.

Table 7.5: Further details on an enriched GO term in Figure 7.7(b), obtained by the tool GO::TermFinder [13].

Item	Value
GO ID	GO:0008285
Term	Negative regulation of cell proliferation
p-value	0.000259
Corrected p-value	0.0184
Annotated Genes	*BTG2*, *PPM1D*
Genome frequency of use	134 out of 23531 genes

7.3.3 Supporting evidence from the literature

BTG2 is a negative regulator of cell cycles, and impaired expression of *BTG2* has been found in breast, renal, and prostate cancers in human [34,47,96]. *WT1* is a gene encoding zinc-finger transcription factor, and defects in *WT1* are a cause of *Wilms' tumor* (WT), an embryonal malignancy of the kidney [64]. *PPM1D* is a *p53*-inducible protein phosphatase and its overexpression has been reported to cause breast cancer and neuroblastoma in human [59, 86].

Interestingly, *BTG2*, *WT1*, and *PPM1D* have been shown to be directly associated with the function of *p53*, a tumor suppressor gene whose activation results in cell cycle arrest and apoptosis upon DNA damage, viral infection and oncogene activation [104]. Since inactivation of *p53* by deletion or mutation can cause tumor, it is also possible that the impaired function of *p53* by dysregulation of *BTG2*, *WT1*, or *PPM1D* mediated by *miR-15a* and *miR-16* might promote tumor development in an indirect way.

Several lines of evidence suggest that miRNAs may be related to leukemia and other cancers. For example, the human *miR-15a* and *miR-16* are clustered within 0.5 kb on chromosome 13q14, and this region has been shown to be deleted in *B cell chronic lymphocytic leukemia* (B-CLL), mantle cell lymphoma, multiple myeloma, and prostate cancer [19, 68, 95]. A recent study by [19] demonstrated that *miR-15a* and *miR-16* are located within a 30-kb region of loss in CLL, and both genes are

deleted or down-regulated in more than two thirds of CLL cases, strongly suggesting the involvement of miRNA genes in human cancers.

7.4 Discussions

7.4.1 A strategy for biological validation

Given that *miR-15a* and *miR-16* are detected together and found to regulate a set of genes that are actively involved in tumorigenesis, further studies should be focused on elucidating the role of *miR-15a* and *miR-16* in other types of cancers through dysregulation of their target gene expression.

For instance, whether *miR-15a* or *miR-16* is involved in the incidence of breast cancer through the regulation of *PPM1D* can be tested as follows. First of all, in order to determine whether or not those miRNAs participate in breast cancer pathogenesis, the level of miRNAs will be measured in breast cancer samples and breast cancer cell lines by Northern Blot. If a significant reduction or increase in the expression of *miR-15a* or *miR-16* is observed compared with normal tissues, gain-of-function and loss-of-function analyses will be performed to demonstrate that dysregulation of *miR-15a* or *miR-16* is directly associated with breast cancer. Since the expression of *PPM1D*, a predicted target for *miR-15a* and *miR-16*, has been found to increase in breast cancer, it is highly likely that those miRNAs are down-regulated in breast cancer patients.

For gain of function analysis, breast cancer cells will be infected with retroviral vectors containing *miR-15a* or *miR-16*, and whether ectopic expression of the miRNAs would revert or alleviate tumorigenesis will be examined. In addition, cancer cells infected with the miRNAs will be implanted into nude mice, and tumor growth and malignant progression will be compared to that of unmanipulated cancer cells.

Since the over-expression of *PPM1D* has been shown to lead to breast cancer, ectopic expression of *miR-15a* or *miR-16* is expected to repress *PPM1D* protein expression and reduce tumor burden, if *PPM1D* is a functional target. Therefore, whether *PPM1D* is down-regulated by the ectopic expression of *miR-15a* or *miR-16*

at the protein level will be examined by Western Blot. At the same time, in order to verify that the level of mRNA is not changed despite the repression of protein expression by miRNAs, *reverse transcription polymerase chain reaction* (RT-PCR) or Northern Blot will be performed.

Furthermore, loss of function experiments will be performed by introducing o-methyl-oligos specific for *miR-15a* or *miR-16*, thus inhibiting the function of these miRNAs. The use of o-methyl oligos will reduce the level of endogenous or ectopically expressed miRNAs and may convert normal cells into tumor cells by increasing *PPM1D* expression.

7.4.2 Extension of our computational method

As the understanding of *in vivo* miRNA target selection mechanisms deepens, more advanced methods to computationally identify miRNA targets will emerge. New findings on miRNA-target interactions will need to be represented by a new relation graph. Our method can then be applied to the augmented relation graph without further modifications. In fact, identifying animal miRNA targets is computationally difficult and there is much room for improvement. This is because animal miRNAs are short and only partially complementary to their targets. Enhanced methods may consider interactions involving RNA binding proteins, conservation filtering through sophisticated phylogenetic profiling techniques, and handling for some unusual structures in targets. For example, a very long loop structure in the target sequence cannot easily be detected without adversely affecting the rate of false positive detection [31].

The miRNA regulatory module defined in this work consists of miRNAs and their targets. Since computational methods have been proposed to identify the genes that encode miRNAs [55, 60, 75, 82], it is possible to redefine the MRM as a group of host genes, the miRNAs encoded by the host genes, and the target genes bound by the miRNAs. This will complete the regulatory chain of 'host gene $\rightarrow$ miRNA $\rightarrow$ target gene.' This new piece of information can be incorporated into the modeling of gene regulatory networks.

7.5 Summary

MicroRNAs are small endogenous RNAs that can play important regulatory roles via the RNA-interference pathway by targeting mRNAs for cleavage or translational repression. Using the biclustering algorithm described in Chapter 4, this chapter proposes a computational method to predict *miRNA regulatory modules* (MRMs) or groups of miRNAs and target genes that are believed to participate cooperatively in post-transcriptional gene regulation. MRMs are similar to Type 1 nested biclusters and can thus be found by the approaches covered in Chapters 3 and 4.

The method to predict MRMs was tested with the human genes and miRNAs, predicting 431 miRNA regulatory modules. We analyze a module that includes genes *BTG2*, *WT1*, *PPM1D*, *PAK7* and *RAB9B*; and miRNAs *miR-15a* and *miR-16*. Review of the literature and annotation with *Gene Ontology* terms reveal that the roles of these genes are indeed closely related in specific biological processes, such as the gene regulation involved in breast, renal, and prostate cancers. Furthermore, it has been reported that *miR-15a* and *miR-16* are deleted together in certain types of cancers, suggesting a possible connection between these miRNAs and cancer. Given that most known functionalities of miRNAs are related to negative gene regulation, extending our approach and exploiting the insight thus obtained may provide clues to achieving practical accuracy in the reverse-engineering of gene regulatory networks.

Chapter 8

Conclusions

Previous chapters described the development of a biclustering algorithm and the application of this algorithm to several problems in computational genomics. This final chapter summarizes the contributions of this dissertation and presents future research directions.

8.1 Dissertation summary

While Chapters 1 and 2 are dedicated to a general introduction and background information, respectively, the other chapters contain the original contributions: Chapters 3 and 4 describe the ZBDD-based biclustering algorithm whereas Chapters 5, 6, and 7 present the application of this algorithm.

- Chapter 3 described a novel biclustering algorithm, which exploits a compact data structure called *zero-suppressed binary decision diagrams* (ZBDDs). ZBDDs have been extensively studied in the field of design and verification of VLSI digital circuits. It has been reported that ZBDDs are useful in solving many practical instances of intractable problems. This ZBDD-based biclustering approach has the important advantages over alternative methods: The proposed algorithm is exact, and it can find all the biclusters satisfying specific input conditions. In addition, this algorithm is scalable to very large data sets due to the use of dynamic programming, a divide-and-conquer strategy, and ZBDDs.

In this chapter, the ZBDD-based algorithm was described at length, focusing on handling gene expression data sets.

- Chapter 4 presented a method to find nested biclusters. This method is a generalized and extended version of the ZBDD-based biclustering algorithm described in Chapter 3. In fact, the algorithm described in this chapter is a unifying method that can be applied to finding any type of biclusters defined in the same formalism. Moreover, our experimental studies confirmed that this method is far more efficient than alternative algorithms. This chapter introduced the notion of nested biclusters: A bicluster is *nested* if any sub-bicluster of this bicluster is yet another bicluster under the same input condition. Nested biclusters can have several desirable properties. For instance, we can apply the dynamic programming paradigm to devise an efficient method to find nested biclusters. In addition, nested biclusters can model more coherent patterns than non-nested biclusters. Many bicluster definitions appearing in the literature describe nested biclusters, and the proposed biclustering technique is applicable with minor modifications. In this chapter, three examples of nested bicluster definitions were introduced as case studies.

- Chapter 5 showed the first application of the proposed biclustering algorithm to gene expression data analysis. DNA microarray technology allows us to monitor transcription levels of thousands of genes simultaneously. This fascinating technology helps us to annotate gene functions, reconstruct gene regulatory networks, diagnose disease conditions, and characterize effects of medical treatments. In this chapter, several gene expression data sets obtained from DNA microarray experiments were analyzed by the proposed biclustering method. First, the proposed algorithm and alternative algorithms were compared in terms of response time, the number of biclusters that can be found, and the size of data sets that can be handled. In addition, the biclusters discovered by different methods were evaluated in terms of compatibility to prior biological knowledge. It was confirmed that the proposed algorithm consistently outperforms the alternatives.

- Chapter 6 presented a method to link gene expression with clinical traits by biclustering. It can have clinical impact if we can determine which genes are responsible for a given set of clinical traits. Now that we can monitor expression levels of many genes at the same time due to DNA microarray technology, it has become feasible to correlate each gene expression level with the observed instances of clinical traits. In the experiment explained in this chapter, from one matrix of gene expression levels and patients and another matrix of patients and their clinical traits, an intermediate matrix of gene expression levels and clinical traits was constructed by averaging over the set of patients. Statistically significant biclusters were found from this intermediate matrix in an unsupervised fashion by the proposed method. Some biclusters discovered were validated with existing biological knowledge, and they are promising in that they can potentially provide new hypotheses for further clinical studies.

- Chapter 7 covered the prediction of microRNA regulatory modules. MicroRNAs are small endogenous RNAs that play important regulatory roles via the RNA-interference pathway by targeting mRNAs for cleavage or translational repression. Given that miRNA target gene selection is guided by the sequence, algorithms have been suggested to systematically identify miRNA target genes. In this chapter, interactions between miRNAs and their target genes were modeled using a weighted bipartite graph. In this graph, bicliques with some edge constraints were named *miRNA regulatory modules* (MRMs) and discovered using the proposed biclustering algorithm. MRMs correspond to groups of miRNAs and their target genes that are believed to participate cooperatively in post-transcriptional gene regulation. This method was tested with the human genome, and a number of statistically significant MRMs were discovered. Some MRMs were thoroughly validated with the literature as well as Gene Ontology, and the functional coherence of the genes in these MRMs was confirmed. Extending this approach can be very useful in reconstructing gene regulatory networks, since the regulation patterns by miRNAs has not been appreciated before in the process of gene network reconstruction.

In summary, this dissertation proposed a series of novel data mining techniques that can find a variety of local structures appearing in large-scale genomic data sets in an unsupervised fashion. The effectiveness of this method was confirmed with a number of experimental studies.

8.2 Future work

An interesting research direction could lie in enhancing the computation speed involved in *Bayesian networks* using the ZBDD-based data management framework established through this dissertation. Bayesian networks, also termed *belief networks* or *probabilistic networks*, are graphical models for visually representing the interaction between variables [5]. In essence, Bayesian networks provide a compact representation of joint probability distributions. However, a Bayesian network is a "representational" factorization of a probability distribution, not a "computational" one [25]. That is, although the network allows us to compactly represent the distribution, it needs to be processed further to obtain answers to arbitrary probabilistic queries. This *inference problem* in Bayesian networks is NP-hard, and in theory, even approximate inference of probabilities in Bayesian networks can be NP-hard [71]. Thus, any Bayesian network approach is typically very computationally intensive, although the use of Bayesian networks is recently widespread in many disciplines in which uncertainty needs to be handled. *Binary decision diagrams* (BDDs) are a special kind of Bayesian networks [101], and it may be possible to devise a fast algorithm for the inference problem in Bayesian networks using a BDD-based approach.

In addition to the applications described in this dissertation, biclustering can be useful in many other contexts. For example, one of the earliest developments of biclustering was in the field of natural language text processing, or text mining [66]. The rapid growth of digitally stored scientific literature provides attractive opportunities for biomedical text mining as well. A starting point for applying any data mining algorithm to a corpus is to create a *vector space model*, also known as *a bag-of-words model* [87]. The basic concept in this model is to extract unique content-bearing words from the corpus treating these words as features and to represent each document in

the corpus as a vector of certain weighted word frequencies in this feature space [28]. Thus, the entire document collection may be treated as a *word-by-document* matrix, in which rows corresponds to words and columns to documents. A subset of the MEDLINE repository[1] can be used to build a word-by-document matrix, and additional information extracted from other biomedical resources can be linked to this matrix using the methodology introduced in Chapter 6. Finding biclusters from this combined information then validating them may provide an interesting scheme for biomedical text mining.

Another interesting application may be genome-wide association studies to obtain information on the association of *single nucleotide polymorphism* (SNP) to phenotypes across the entire genome. SNPs are the most common form of genetic variation in humans comprising almost 0.1% of the average human genome. Predicting and understanding the downstream effects of genetic variation using computational methods are becoming increasingly important for SNP selection in genetic studies [72]. The biclustering method investigated in this study can be applied as follows. We can produce a *subject-by-genotype* matrix in which rows correspond to subjects under study and columns to SNP genotypes for each subject. We can also record phenotypes for each subject and construct a *subject-by-phenotype* matrix. Then, it is possible to find a *genotype-by-phenotype* matrix following the technique described in Chapter 6. Finding biclusters from this matrix corresponds to associating genotypes with phenotypes, and evaluation of these biclusters may provide valuable biological insight.

[1] `http://www.ncbi.nlm.nih.gov/entrez/`

Bibliography

[1] R Agrawal, T Imielinski, and A N Swami. Mining association rules between sets of items in large databases. In *Proc. ACM SIGMOD '93,*, pages 207–216, 1993.

[2] Alfred V Aho, John E Hopcroft, and Jeffrey D Ullman. *Data Structures and Algorithms*. Addison-Wesley, Reading, Massachusetts, 1983.

[3] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. *Molecular Biology of the Cell*. Garland Science, New York, fourth edition, 2002.

[4] Ash Alizadeh et al. Distinct types of diffuse large B-cell lymphoma identified by gene-expression profiling. *Nature*, 4051:503–511, 2000.

[5] Ethem Alpaydin. *Introduction to Machine Learning*. MIT Press, Massachusetts, 2004.

[6] Russ B Altman and Soumya Raychaudhuri. Whole-genome expression analysis: challenges beyond clustering. *Current Opinion in Structural Biology*, 11:340–347, 2001.

[7] Pierre Baldi and Søren Brunak. *Bioinformatics: The Machine Learning Approach*. MIT Press, Massachusetts, 2nd edition, 2001.

[8] David P Bartel. MicroRNAs: Genomics, biogenesis, mechanism, and function. *Cell*, 116(2):281–297, 2004.

[9] R E Bellman. *Adaptive Control Processes*. Princeton University Press, New Jersey, 1961.

[10] Amir Ben-Dor, Benny Chor, Richard Karp, and Zohar Yakhini. Discovering local structure in gene expression data: The order-preserving submatrix problem. *Journal of Computational Biology*, 10(3-4):373–384, 2003.

[11] Y Benjamini and Y Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society*, 57(1):289–300, 1995.

[12] Jeremy M. Berg, John L. Tymoczko, and Lubert Stryer. *Biochemistry*. W.H. Freeman and Company, New York, 5th edition, 2002.

[13] E I Boyle, S Weng, J Gollub, H Jin, D Botstein, J M Cherry, and G Sherlock. GO::TermFinder. *Bioinformatics*, 20(18):3710–3715, December 2004.

[14] K S Brace, R L Rudell, and R E Bryant. Efficient implementation of a BDD package. In *Proceedings of the 27th Design Automation Conference*, pages 40–45, 1990.

[15] Randal E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

[16] Randal E. Bryant. Binary decision diagrams and beyond: Enabling technologies for formal verification. In *IEEE/ACM International Conference on Computer Aided Design, ICCAD, San Jose/CA*, pages 236–243. IEEE CS Press, Los Alamitos, 1995.

[17] L Bullinger, K Dohner, E Bair, S Frohling, R F Schlenk, R Tibshirani, H Dohner, and J R Pollack. Use of gene-expression profiling to identify prognostic subclasses in adult acute myeloid leukemia. *N Engl J Med.*, 350(16):1605–1616, April 2004.

[18] Andrea Califano, Gustavo Stolovitzky, and Yuhai Tu. Analysis of gene expression microarrays for phenotype classification. In *Proc. Int Conf Intell Syst Mol Biol*, pages 75–85, 2000.

[19] G A Calin, C D Dumitru, M Shimizu, R Bichi, S Zupo, E Noch, H Aldler, S Rattana, M Keating, K Rai, L Rassenti, T Kipps, M Negrini, F Bullrich, and C M Croce. Frequent deletions and down-regulation of micro-RNA genes miR15 and miR16 at 13q14 in chronic lymphocytic leukemia. *Proc Natl Acad Sci USA*, 99(24):15524–15529, 2002.

[20] W Chen, M Reiss, and D J Foran. A prototype for unsupervised analysis of tissue microarrays for cancer research and diagnostics. *IEEE Trans Inf Technol Biomed*, 8(2):89–96, 2004.

[21] Yizong Cheng and George M. Church. Biclustering of expression data. In *Proceedings of ISMB*, pages 93–103, 2000.

[22] R J Cho, M J Campbell, E A Winzeler, L Steinmetz, A Conway, L Wodicka, T G Wolfsberg, A E Gabrielian, D Landsman, D J Lockhart, and R W Davis. A genome-wide transcriptional analysis of the mitotic cell cycle. *Mol. Cell*, 2(1):65–73, July 1998.

[23] A Cignetti, E Bryant, B Allione, A Vitale, R Foa, and M A Cheever. CD34(+) acute myeloid and lymphoid leukemic blasts can be induced to differentiate into dendritic cells. *Blood*, 94:2048–2055, 1999.

[24] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, Massachusetts, 2001.

[25] Adnan Darwiche. A logical approach to factoring belief networks. In *Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning*, pages 409–420, April 2002.

[26] Giovanni De Micheli. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, New York, 1994.

[27] J DeRisi, L Penland, P O Brown, M L Bittner, P S Meltzer, M Ray, Y Chen, Y A Su, and J M Trent. Use of a cDNA microarray to analyse gene expression patterns in human cancer. *Nature Genetics*, 14(4):457–460, 1996.

[28] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Knowledge Discovery and Data Mining*, pages 269–274, 2001.

[29] Sorin Drăghici. *Data Analysis Tools for DNA Microarrays*. Chapman & Hall/CRC, Florida, 2003.

[30] Richard O Duda, Peter E Hart, and David G Stork. *Pattern Classification*. Wiley, New York, 2nd edition, 2001.

[31] Anton J Enright, Bino John, Ulrike Gaul, Thomas Tuschl, Chris Sander, and Debora S Marks. MicroRNA targets in *Drosophila*. *Genome Biology*, 5(1):R1, 2003.

[32] Tom Fawcett. ROC graphs: Notes and practical considerations for data mining researchers. *HP Laboratories Technical Report*, 2003.

[33] Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, pages 534–543, May 2002.

[34] M A Ficazzola, M Fraiman, J Gitlin, K Woo, J Melamed, M A Rubin, and P D Walden. Antiproliferative B cell translocation gene 2 protein is down-regulated post-transcriptionally as an early event in prostate carcinogenesis. *Carcinogenesis*, 22(8):1271–1279, 2001.

[35] G. Getz, E. Levine, and E. Domany. Coupled two-way clustering analysis of gene microarray data. *Proc. Natl. Acad. Sci USA*, 94:12079–12084, 2000.

[36] Dan Gusfield. *Algorithms on String, Trees and Sequences: Computer Science and Computational Biology*. Cambridge, New York, 1997.

[37] David Hand, Heikki Mannila, and Padhraic Smyth. *Principles of Data Mining*. MIT Press, Massachusetts, 2001.

[38] Beth D. Harrison, Julie A. Adams, Mark Briggs, Michelle L. Brereton, and John A. Liu Yin. Stimulation of autologous proliferative and cytotoxic T-cell responses by "leukemic dendritic cells" derived from blast cells in acute myeloid leukemia. *Blood*, 979:2764–2771, 2001.

[39] Leland H. Hartwell, Leroy Hood, Michael L. Goldberg, Ann E. Reynolds, Lee M. Silver, and Ruth C. Veres. *Genetics*. McGraw-Hill, New York, 2000.

[40] Arjang Hassibi and Thomas H Lee. A programmable electrochemical biosensor array in $0.18\mu m$ standard CMOS. In *Proceedings of IEEE International Solid-State Circuits Conference*, 2005.

[41] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, New York, 2001.

[42] John P T Higgins, Rajesh Shinghal, Harcharan Gill, Jeffrey H Reese, Martha Terris, Ronald J Cohen, Michael Fero, Jonathan R Pollack, Matt van de Rijn, and James D Brooks. Gene expression patterns in renal cell carcinoma assessed by complementary DNA microarray. *American Journal of Pathology*, 162(3):925–932, March 2003.

[43] S Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70, 1979.

[44] S E Jacobsen, O P Veiby, J Myklebust, C Okkenhaug, and S D Lyman. Ability of flt3 ligand to stimulate the in vitro growth of primitive murine hematopoietic progenitors is potently and directly inhibited by transforming growth factor-beta and tumor necrosis factor-alpha. *Blood*, 87:5016–5026, 1996.

[45] Bino John, Anton J Enright, Alexei Arain, Thomas Tuschl, Chris Sander, and Debora S Marks. Human microRNA targets. *PLoS Biology*, 2(11):e363, 2004.

[46] I T Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, second edition, 2002.

[47] H Kawakubo, J L Carey, E Brachtel, V Gupta, J E Green, P D Walden, and S Maheswaran. Expression of the NF-kappaB-responsive gene BTG2 is aberrantly regulated in breast cancer. *Oncogene*, 23(50):8310–8319, 2004.

[48] J Keski-Oja, E B Leof, R M Lyons, R J Coffey Jr, and H L Moses. Transforming growth factors and control of neoplastic cell growth. *J Cell Biochem*, 33:95–107, 1987.

[49] Marianthi Kiriakidou, P T Nelson, A Kouranov, P Fitziev, C Bouyioukos, Z Mourelatos, and A Hatzigeorgiou. A combined computational-experimental approach predicts human microRNA targets. *Genes and Development*, 18(10):1165–1178, 2004.

[50] S Kiyonaka, K Sada, I Yoshimura, S Shinkai, N Kato, and I Hamachi. Semi-wet peptide/protein array using supramolecular hydrogel. *Nature Mater.*, 3(1):58–64, 2004.

[51] Y Kluger, R Basri, J T Chang, and M Gerstein. Spectral biclustering of microarray data: Coclustering genes and conditions. *Genome Research*, 13(4):703–716, April 2003.

[52] I S Kohane, A T Kho, and A J Butte. *Microarrays for an Integrative Genomics*. The MIT Press, Cambridge, Massachusetts, 2003.

[53] M Lagos-Quintana, R Rauhut, J Meyer, A Borkhardt, and T Tuschl. New microRNAs from mouse and human. *RNA*, 9(2):175–179, 2003.

[54] Eric C Lai. Predicting and validating microRNA targets. *Genome Biology*, 5(9):115.1–6, 2004.

[55] Eric C Lai, Pavel Tomancak, Robert W Williams, and Gerald M Rubin. Computational identification of *Drosophila* microRNA genes. *Genome Biology*, 4(7):R42.1–20, 2003.

[56] L Lazzeroni and Art Owen. Plaid models for gene expression data. *Stanford University Technical Report*, 2000.

[57] Benjamin P Lewis, Christopher B Burge, and David P Bartel. Conserved seed pairing, often flanked by adenosines, indicates that thousands of human genes are microRNA targets. *Cell*, 120:15–20, 2005.

[58] Benjamin P Lewis, I-hung Shih, Matthew W Jones-Rhoades, David Partel, and Christopher B Burge. Prediction of mammalian microRNA targets. *Cell*, 115(7):787–798, 2003.

[59] J Li, Y Yang, Y Peng, R J Austin, W G van Eyndhoven, K C Nguyen, T Gabriele, M E McCurrach, J R Marks, T Hoey T, S W Lowe, and S Powers. Oncogenic properties of PPM1D located within a breast cancer amplification epicenter at 17q23. *Nature Genetics*, 31(2):133–134, 2002.

[60] Lee P Lim, Margaret E Glasner, Soraya Yekta, Christopher B Burge, and David P Bartel. Vertebrate microRNA genes. *Science*, 299(5612):1540, 2003.

[61] J Liu, J Yang, and Wei Wang. Biclustering in gene expression data by tendency. In *Proceedings of CSB*, pages 182–193, 2004.

[62] D Lockhart, H Dong, M Byrne, M Follettie, M Gallo, M Chee, M Mittmann, C Wang, M Kobayashi, H Horton, and E L Brown. Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nature Biotechnology*, 14(13):1675–1680, 1996.

[63] Harvey Lodish, Arnold Berk, Paul Matsudaira, Chris A. Kaiser, Monty Krieger, Matthew P. Scott, Lawrence Zipursky, and James Darnell. *Molecular Cell Biology*. W. H. Freeman, New York, fifth edition, 2003.

[64] D M Loeb and S Sukumar. The role of WT1 in oncogenesis: tumor suppressor or oncogene? *Int J Hematol.*, 76(2):117–126, 2002.

[65] Sara C Madeira and Arlindo L Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

[66] Christopher D Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Massachusetts, 1999.

[67] Christoph Meinel and Thorsten Theobald. *Algorithms and Data Structures in VLSI Design*. Springer, Berlin, 1998.

[68] A Migliazza, E Cayanis, F Bosch-Albareda, H Komatsu, S Martinotti, E Toniato, S Kalachikov, M F Bonaldo, P Jelene, X Ye, A Rzhetsky, X X Qu, M Chien, G Inghirami, G G Gaidano, U Vitolo, G Saglio, L L Resegotti, P P Zhang, M B Soares, J Russo, S G Fischer, I S Edelman, A Efstratiadis, and R Dalla-Favera. Molecular pathogenesis of B-cell chronic lymphocytic leukemia: analysis of 13q14 chromosomal deletions. *Curr Top Microbiol Immunol.*, 252:275–284, 2000.

[69] Shinichi Minato. Zero-suppressed BDDs for set manipulation in combinatorial problems. In *IEEE/ACM Design Automation Conference, DAC, Dallas/TX*, pages 272–277. ACM Press, New York, 1993.

[70] Shinichi Minato. *Binary Decision Diagrams and Applications for VLSI CAD*. Kluwer, 1996.

[71] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[72] Sean Mooney. Bioinformatics approaches and resources for single nucleotide polymorphism functional analysis. *Briefings in Bioinformatics*, 6(1):44–56, March 2005.

[73] Gordon E. Moore. Cramming more components onto integrated circuits. *Electronics Magazine*, April 1965.

[74] T M Murali and Simon Kasif. Extracting conserved gene expression motifs from gene expression data. In *Proceedings of Pacific Symposium on Biocomputing*, pages 77–88, 2003.

[75] Uwe Ohler, Soraya Yekta, Lee P Lim, David P Bartel, and Christopher B Burge. Patterns of flanking sequence conservation and a characteristic upsteam motif for microRNA gene identification. *RNA*, 10(9):1309–1322, 2004.

[76] R Peeters. The maximum edge biclique problem is NP-complete. *Discrete Applied Math.*, 131(3):651–654, 2003.

[77] Nikolaus Rajewsky and Nicholas D Socci. Computational identification of microRNA targets. *Developmental Biology*, 267(2):529–535, 2003.

[78] Soumya Raychaundhuri, Patric D Sutphin, Jeffrey T Chang, and Russ B Altman. Basic microarray analysis: grouping and feature reduction. *Trends in Biotechnology*, 19(5):189–193, May 2001.

[79] Marc Rehmsmeier, Peter Steffen, Matthias Hchsmann, and Robert Giegerich. Fast and effective prediction of microRNA/target duplexes. *RNA*, 10(10):1507–1517, 2004.

[80] John A. Rice. *Mathematical Statistics and Data Analysis*. Duxbury Press, 1994.

[81] E Riedl, H Strobl, O Majdic, and W Knapp. TGF-beta 1 promotes in vitro generation of dendritic cells by protecting progenitor cells from apoptosis. *J Immunol*, 158:1591–1597, 1997.

[82] Antony Rodriguez, Sam Griffiths-Jones, Jennifer L Ashurst, and Allan Bradley. Identification of mammalian microRNA host genes and transcription units. *Genome Research*, 14(10A):1902–1910, 2004.

[83] B Rosner. *Fundamentals of Biostatistics*. Duxbury, Pacific Grove, California, 5th edition, 2000.

[84] A Y Rubina, E I Dementieva, A A Stomakhin, E L Darii, S V Pan'kov, V E Barsky, S M Ivanov, E V Konovalova, and A D Mirzabekov. Hydrogel-based protein microchips: manufacturing, properties, and applications. *Biotechniques*, 34(5):1008–1014, 2003.

[85] Stuart Russell and Peter Norvig. *Artificial Intelligence: A modern Approach*. Prentice Hall, New Jersey, 2nd edition, 2003.

[86] F Saito-Ohara, I Imoto, J Inoue, H Hosoi, A Nakagawara, T Sugimoto, and J Inazawa. PPM1D is a potential target for 17q gain in neuroblastoma. *Cancer Research*, 63(8):1876–1883, 2003.

[87] G Salton and M J McGill. *Introduction to Modern Retrieval*. McGraw-Hill, 1983.

[88] Tsutomu Sasao and Masahiro Fujita. *Representations of Discrete Functions*. Kluwer, Massachusetts, 1996.

[89] M Schienle, C Paulus, A Frey, F Hofmann, B Holzapfl, P Schindler-Bauer, and R Thewes. A fully electronic DNA sensor with 128 positions and in-pixel A/D conversion. *IEEE Journal of Solid-State Circuits*, 39(12):2438–2445, 2004.

[90] E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller. Rich probabilistic models for gene expression. *Bioinformatics*, 17(Suppl 1):S243–52, 2001.

[91] I S Shergill, N K Shergill, M Arya, and H R Patel. Tissue microarrays: a current medical research tool. *Curr Med Res Opin.*, 20(5):707–712, 2004.

[92] Neil R Smalheiser and Vetle I Torvik. A population-based statistical approach identifies parameters characteristic of human microRNA-mRNA interactions. *BMC Bioinformatics*, 5(1):139, 2004.

[93] Robert R. Sokal and F. James Rohlf. *Biometry*. WH Freeman and Co., 1994.

[94] Alexander Stark, Julius Brennecke, Robert B Russel, and Stephen M Cohen. Identification of *Drosophila* microRNA targets. *PLoS Biology*, 1(3):397–409, 2003.

[95] S Stilgenbauer, J Nickolenko, J Wilhelm, S Wolf, S Weitz, K Dohner, T Boehm, H Dohner, and P Lichter. Expressed sequences as candidates for a novel tumor suppressor gene at band 13q14 in B-cell chronic lymphocytic leukemia and mantle cell lymphoma. *Oncogene*, 16(14):1891–1897, 1998.

[96] K Struckmann, P Schraml, R Simon, K Elmenhorst, M Mirlacher, J Kononen, and H Moch. Impaired expression of the cell cycle regulator BTG2 is common in clear cell renal cell carcinoma. *Cancer Research*, 64(5):1632–1638, 2004.

[97] M Sultan, D A Wigle, C A Cumbaa, M Maziarz, J Glasgow, M S Tsao, and I Jurisica. Binary tree-structured vector quantization approach to clustering and visualizing microarray data. *Bioinformatics*, 18:S111–S119, 2002.

[98] A Tanay, R Sharan, and R Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18:S136–S144, 2002.

[99] Saeed Tavazoie, Jason D. Hughes, Michael J. Campbell, Raymond J. Cho, and George M. Church. Systematic determination of genetic network architecture. *Nature Genetics*, 22:281–285, 1999.

[100] The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.

[101] Robert R. Tucci. Binary decision diagrams are a subset of bayesian nets. *eprint arXiv:quant-ph/0209009*, 2002.

[102] V G Tusher, R Tibshirani, and G Chu. Significance analysis of microarrays applied to the ionizing radiation response. *Proc. Natl. Acad. Sci USA*, 98(9):5116–5121, April 2001.

[103] O P Veiby, F W Jacobsen, L Cui, S D Lyman, and S E Jacobsen. The flt3 ligand promotes the survival of primitive hemopoietic progenitor cells with myeloid as well as b lymphoid potential. Suppression of apoptosis and counteraction by TNF-alpha and TGF-beta. *J Immunol*, 157:2953–2960, 1996.

[104] B Vogelstein, D Lane, and A J Levine. Surfing the p53 network. *Nature*, 408(6810):307–310, 2000.

[105] Z Šidák. Rectangular confidence regions for the means of multivariate normal distributions. *Journal of the American Statistical Association*, 62:626–633, 1967.

[106] C C Wang, R P Huang, M Sommer, H Lisoukov, R Huang, Y Lin, T Miller, and J Burke. Array-based multiplexed screening and quantitation of human cytokines and chemokines. *J Proteome Res.*, 1(4):337–343, 2002.

[107] Haixun Wang, Wei Wang, Jiong Yang, and Philip S. Yu. Clustering by pattern similarity in large data sets. In *Proceedings of ACM SIGMOD*, pages 394–405, 2002.

[108] E Weisberg, C Boulton, L M Kelly, P Manley, D Fabbro, T Meyer, D G Gilliland, and J D Griffin. Inhibition of mutant FLT3 receptors in leukemia cells by the small molecule tyrosine kinase inhibitor PKC412. *Cancer Cell*, 1:433–443, 2002.

[109] M West, C Blanchette, H Dressman, E Huang, S Ishida, R Spang, H Zuzan, J A Olson Jr, J R Marks, and J R Nevins. Predicting the clinical status of human breast cancer by using gene expression profiles. *Proc Natl Acad Sci USA*, 98(20):11462–11467, September 2001.

[110] P H Westfall and S S Young. *Resampling-based multiple testing: Examples and Methods for p-value adjustment*. Wiley, New York, 1993.

[111] A R Whitney, M Diehn, S J Popper, A A Alizadeh, J C Boldrick, D A Relman, and P O Brown. Individuality and variation in gene expression patterns in human blood. *Proc Natl Acad Sci USA*, 100(4):1896–1901, February 2003.

[112] Chang-Jiun Wu, Yutao Fu, T M Murali, and Simon Kasif. Gene expression module discovery using gibbs sampling. *Genome Informatics*, 15(1):239–248, 2004.

[113] Jiong Yang, Haixun Wang, Wei Wang, and Philip Yu. Enhanced biclustering on expression data. In *Proc. IEEE 3rd Symposium on Bioinformatics and Bioengineering*, pages 321–327, 2003.

[114] Sungroh Yoon, Luca Benini, and Giovanni De Micheli. Finding co-clusters of genes and clinical parameters. In *Proceedings of the 27th Annual International Conference of the IEEE EMBS*, September 2005.

[115] Sungroh Yoon, Luca Benini, and Giovanni De Micheli. A pattern mining mehtod for high-throughput lab-on-a-chip data analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(2), February 2006, in press.

[116] Sungroh Yoon and Giovanni De Micheli. An application of zero-suppressed binary decision diagrams to clustering analysis of DNA microarray data. In *Proceedings of the 26th Annual International Conference of the IEEE EMBS*, pages 2925–2928, September 2004.

[117] Sungroh Yoon and Giovanni De Micheli. Prediction and analysis of human microRNA regulatory modules. In *Proceedings of the 27th Annual International Conference of the IEEE EMBS*, September 2005.

[118] Sungroh Yoon and Giovanni De Micheli. Prediction of regulatory modules comprising microRNAs and target genes. *Bioinformatics*, 21:ii93–ii100, September 2005.

[119] Sungroh Yoon, Christine Nardini, Luca Benini, and Giovanni De Micheli. Enhanced pClustering and its applications to gene expression data. In *Proceedings of IEEE 4th Symposium on Bioinformatics and Bioengineering*, pages 275–282, May 2004.

[120] Sungroh Yoon, Christine Nardini, Luca Benini, and Giovanni De Micheli. Discovering coherent biclusters from gene expression data using zero-suppressed binary decision diagrams. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2(4), October-December 2005, in press.