Discovering Coherent Biclusters from Gene Expression Data Using Zero-Suppressed Binary Decision Diagrams

Sungroh Yoon, Christine Nardini, Luca Benini, and Giovanni De Micheli

Abstract—The biclustering method can be a very useful analysis tool when some genes have multiple functions and experimental conditions are diverse in gene expression measurement. This is because the biclustering approach, in contrast to the conventional clustering techniques, focuses on finding a subset of the genes and a subset of the experimental conditions that together exhibit coherent behavior. However, the biclustering problem is inherently intractable, and it is often computationally costly to find biclusters with high levels of coherence. In this work, we propose a novel biclustering algorithm that exploits the *zero-suppressed binary decision diagrams* (ZBDDs) data structure to cope with the computational challenges. Our method can find all biclusters that satisfy specific input conditions, and it is scalable to practical gene expression data. We also present experimental results confirming the effectiveness of our approach.

Index Terms—Clustering, life and medical sciences, bioinformatics (genome or protein) databases, logic design.

1 Introduction

Cluster analysis, or *clustering*, is an unsupervised learning technique 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 [14]. Although there is mature statistical literature on clustering, DNA microarray data have sparked the development of multiple new methods [22]. In particular, the *biclustering* technique [7], [21], [8], [4], [13], [15], [16], [26], [29], [31], [33], [34] is one of the most promising innovations in this area [3]. Given a gene expression data matrix, this technique seeks to find a *bicluster*, or a subset of genes displaying similar behavior under a subset of conditions.

The biclustering technique is more suitable for cases in which genes have multiple functions and experimental conditions are diverse. Consequently, this method may provide additional biological insight that has been overlooked by traditional clustering approaches. For example, biclustering is more compatible with our understanding of cellular processes: We expect subsets of genes to be coregulated and coexpressed under certain experimental conditions, but to behave almost independently under other conditions [4]. Thus, the biclustering method may be useful in recognizing reusable genetic "modules" that are mixed and matched in order to create more complex genetic responses [3].

- *S. Yoon is with the Computer Systems Laboratory, Stanford University, Room 334, William Gates Computer Science Hall, Stanford, CA 94305. E-mail: sryoon@stanford.edu.*
- *C. Nardini and L. Benini are with DEIS, University of Bologna, Viale Risorgimento 2, Bologna 40136 Italy. E-mail: {cnardini, lbenini}@deis.unibo.it.*
- *G. De Micheli is with the Integrated Systems Center, EPFL, INF 341 CH-1015 Lausanne, Switzerland. E-mail: giovanni.demicheli@epfl.ch.*

Manuscript received 25 May 2004; revised 17 Oct. 2004; accepted 11 Mar. 2005; published online 31 Aug. 2005.
For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBB-0059-0504.

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. In addition, it may be helpful to characterize biclusters by the degree of *fluctuation* in gene expression levels, which cannot be captured by coherence measurement. In Fig. 1, we present a plot in which biclusters can be placed according to their degree of coherence and fluctuation.

In some applications, such as gene coregulation analysis, the biclusters in area A would be the most interesting because similar behavior between highly expressed genes is much more important than that between two poorly expressed genes [13]. On the other hand, the "flat" biclusters in area 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 subphenotypes. 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 B are less interesting because they have a lower level of coherence than those in areas A or C.

Many techniques have been proposed to find biclusters with a high level of coherence, particularly those that can be placed in areas A or C on the characterization plot. Some methods define a bicluster in a way such that any subbicluster of the bicluster is yet another bicluster under the same definition and input parameters. Examples include the δ-valid kj-patterns [7], OPSMs [4], xMOTIFs [21], δ-pClusters [31], and GEMS [32]. By measuring coherence with fine granularity, these biclusters can potentially exhibit high degrees of coherence [31]. Our approach also takes advantage of this property to find coherent biclusters.

The cluster search problem is in general NP-hard [28], and the biclustering problem is no exception [8], [31], [29]. To cope with this computational challenge, our method exploits a compact data structure called *zero-suppressed binary decision diagrams* (ZBDDs) [19], [20], [18] to implicitly

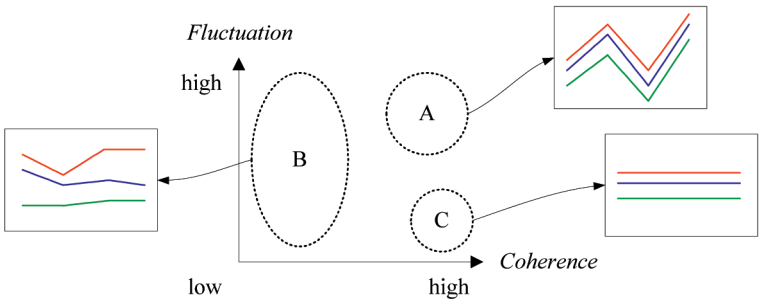


Fig. 1. Characterization of biclusters. In some applications, such as gene coregulation analysis, the biclusters in area A are most interesting. On the other hand, the biclusters in area C are important in other applications, such as marker gene identification.

represent and manipulate massive data. The ZBDDs have been used widespread in other domains, namely, the computer-aided design of *very large-scale integration* (VLSI) digital circuits, and can be useful in solving many practical instances of intractable problems. We emphasize that our method exploits this property of ZBDDs, and can find all the biclusters that satisfy specific input conditions without exhaustive enumeration.

In Section 2, we brief the reader on the relevant biclustering techniques as well as the fundamentals of ZBDDs. We also provide a formal problem statement. Section 3 introduces a special kind of bicluster that plays a crucial role in helping the reader to understand our method. In Section 4, we present the essential properties of biclusters in our definitions. We propose our biclustering algorithm in Section 5 and, in Section 6, we present the results from our experimental studies.

2 Preliminaries

2.1 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) [15]. Our method is applicable to both.

A bicluster is defined to be a subset of genes that exhibit similar behavior under a 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 1. An example of data matrix $D = (U_G, U_E)$ and biclusters on D are shown in Figs. 2a and 2b, respectively. Throughout the paper, we are going to explain how to find these two biclusters from matrix D.

2.2 Characterization of Biclusters

2.2.1 Definition of Similarity

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 [17], 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. [7] 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. [32] 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 [8] employed the concept of a *residual* to find a bicluster with coherent values. In the 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 [23], [27]. Thus, a low value of residual can show a high degree of coherence, while a high value

	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)

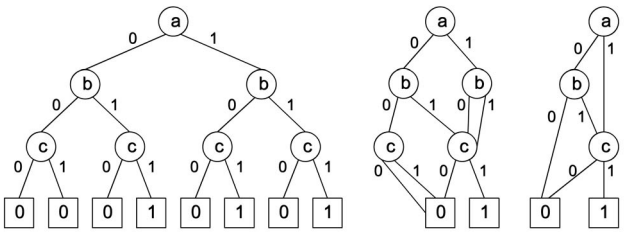
#	G	E
0	$\{g_0, g_2, g_3, g_4\}$	$\{e_0, e_1, e_3\}$
1	$\{g_0, g_2, g_3\}$	$\{e_1, e_3, e_5\}$

(b)

Fig. 2. Example to be referred to throughout the paper. (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 used are $\delta = 1$, $M_G = M_E = 3$, as will be explained in Section 2.4.

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)

Fig. 3. 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).

reveals the opposite. The residual of element a_{ij} in the matrix A denoted by a pair of sets (I, J) is

$$r_{ij} = a_{ij} - \overline{a}_{i\bullet} - \overline{a}_{\bullet j} + \overline{a}_{\bullet\bullet}, \tag{1}$$

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:

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

The pClustering technique by Wang et al. [31] 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. [4] 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 [21] proposed a representation for gene expression data called conserved gene expression motifs (xMOTIFs). An 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.

2.2.2 Degree of Fluctuation in Expression Levels

Depending upon the situation encountered, it may be helpful to characterize biclusters by the degree of fluctuation in gene expression levels as well as by the similarity in behavior. Considering similarity alone is insufficient to capture widely fluctuating gene expression patterns. For instance, the lowest MSR value (zero) indicates that the gene expression levels fluctuate in unison. However, flat biclusters with no fluctuation can also have an MSR value of zero [8].

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

$$ARV(I, J) = \frac{1}{|I||J|} \sum_{i \in I} \sum_{j \in J} (a_{ij} - a_{i\bullet})^2. \tag{3}$$

2.3 Implicit Representation of Boolean Functions

In Sections 3.3 and 5.2.2, we will present a method to implicitly represent and manipulate massive data. This method is based upon an efficient data structure called the *zero-suppressed binary decision diagram* (ZBDD) [20]. To facilitate our explanation in a later section, here we explain the fundamentals of ZBDDs and related concepts. For a more extensive treatment of ZBDDs, the reader can refer to [18], [19], [25], [20].

2.3.1 Binary Decision Diagrams (BDDs)

Boolean logic functions can be represented in several ways. For example, Figs. 3a and 3b 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 [5]. Reduction rules are 1) merge equivalent subgraphs and 2) remove vertices with identical subgraphs. For example, we can apply the first rule to the BDD in Fig. 3b and obtain the BDD in Fig. 3c. Applying the second rule to the BDD in Fig. 3c finally gives the *reduced ordered* BDD (ROBDD) representation in Fig. 3d.

ROBDDs have found widespread use in the optimization and verification of VLSI design [6]. 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 [5], [10].

2.3.2 Zero-Suppressed BDDs (ZBDDs)

Zero-suppressed BDDs [19], [20] are a variant of ROBDDs that represent a set of combinations. 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 ith bit reports whether the ith 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, the sets of combinations are *sparse*, which is defined as the following:

- The sets contain only a small fraction of the 2^n possible bit vectors.
- Each bit vector in the sets has many zeroes.

By exploiting both types of sparsity, ZBDDs provide an efficient representation for manipulating large-scale sets of combinations [19]. Minato [19], [20] proposed two reduction

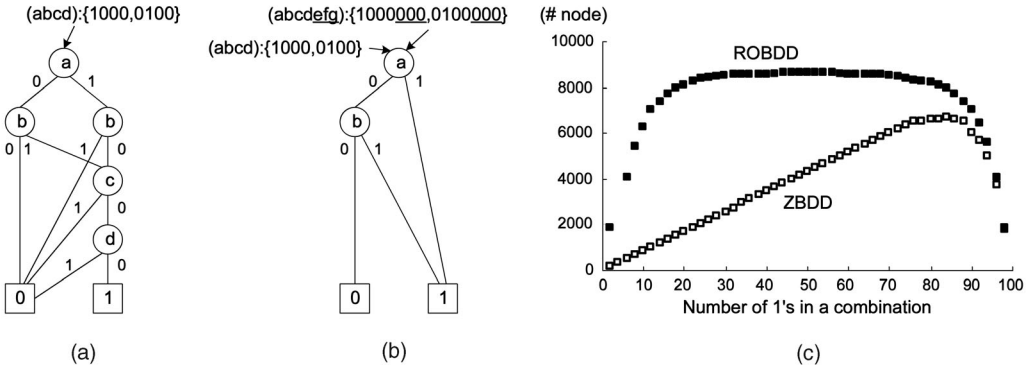


Fig. 4. Representation of a set of combinations. (a) ROBDD representation. (b) ZBDD representation. (c) Comparison of ROBDD and ZBDD [20].

rules to reduce ordinary BDDs to ZBDDs: 1) merge equivalent subgraphs 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.

Although other types of BDDs can represent a set of combinations, ZBDDs provide the most compact representations. For example, the ROBDD in Fig. 4a represents a set of combinations $\{1000, 0100\}$ for four input variables $(abcd)$. By applying the ZBDD reduction rules, we can reduce the BDD in Fig. 4a to the ZBDD in Fig. 4b, which is more compact. In addition, 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. For example, a set of combinations $\{1000000, 0100000\}$ for seven variables $(abcdefg)$ is represented by the same ZBDD in Fig. 4b. This is not the case if we use other types of BDDs. Minato [20] compared the size of a ZBDD with that of an ROBDD for a large set of combinations, as shown in Fig. 4c.

2.4 Formal Definition of a Bicluster and Problem Statement

Definition 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 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 2. The two maximal biclusters in Fig. 2b are found in the data matrix in Fig. 2a 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 [31]. The rationale of choosing this particular bicluster model is that δ-pClusters can have multiple desirable properties. According to Wang et al. [31], 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 Section 6 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 Fig. 1), whereas the biclusters found by pClustering tend to be located mainly in area C [34]. The next paragraph explains this reasoning.

The threshold parameter δ affects many aspects of the biclustering problem, as shown in Fig. 5. First, the difficulty of a biclustering problem depends to some extent on the value of δ since the amount of intermediate data is

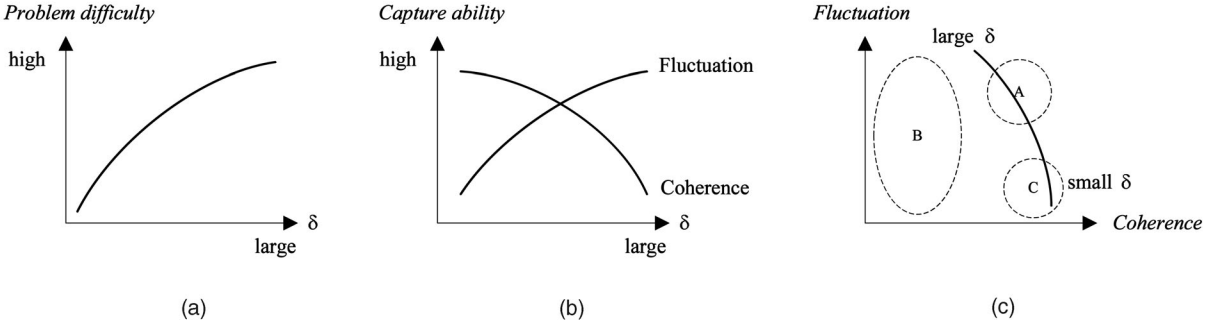


Fig. 5. Qualitative analysis of dependency on δ. (a) A larger value of δ means a more difficult problem. (b) The ability to capture fluctuation is roughly proportional to the value of δ, whereas the ability to capture coherence decreases as the parameter δ becomes larger. (c) A small value of δ tends to find biclusters in area C, while a large value of δ typically finds biclusters in areas A or B.

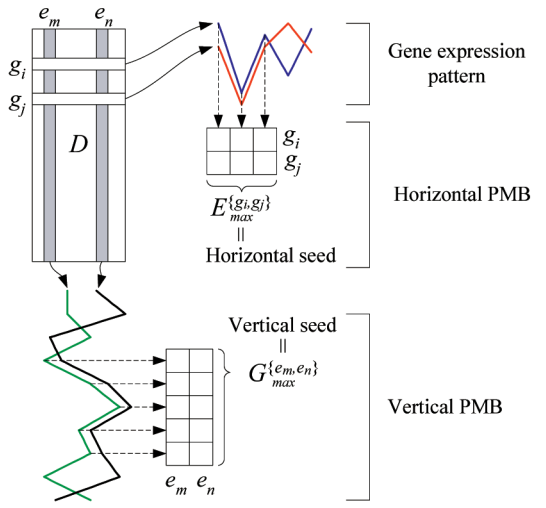


Fig. 6. Pairwise maximal biclusters (PMBs).

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. Figs. 5a and 5b depict these observations. Consequently, as shown in Fig. 5c, 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 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 general, intractable [8], [31], these special biclusters can be discovered in polynomial time.

3.1 Definition of PMBs

We refer to $2 \times |E|$ or $|G| \times 2$ maximal biclusters as *pairwise maximal biclusters* (PMBs). As shown in Fig. 6, a *horizontal PMB* is a bicluster composed 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 (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* $E^{\{g_i,g_j\}}_{max}$. *In this case, we call* B *a* horizontal *PMB for two genes* $\{g_i, g_j\}$, *and denote it by* $B = \left(\{g_i, g_j\}, E^{\{g_i,g_j\}}_{max}\right)$.

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

TABLE 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

By switching the roles of genes and experiments, *vertical PMBs* and *vertical seeds* are similarly defined. Table 1 summarizes the notations defined in this section.

3.2 Generation of PMBs

Wang et al. [31] 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 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 1 is $O(nlogn)$ [31], 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 1 is efficient, and the number of seeds discovered by this algorithm does not grow exponentially.

Algorithm 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: $G^{\{e_i,e_j\}}_{max}$, 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** ($|v| \leq \delta$) **then**
10 **if** ($end - start \geq M_G$ AND new = TRUE) **then**
11 Report $\{s[start].i, s[start+1].i, \ldots, s[end-1].i\}$;
12 $end := end + 1$;
13 $new :=$ TRUE;
14 **else**
15 $start := start + 1$;
16 $new :=$ FALSE;
17 **until** (end $\geq |X|$);
18 **if** ($end - start \geq M_G$ AND new = TRUE) **then**
19 Report $\{s[start].i, s[start+1].i, \ldots, s[end-1].i\}$;

Example 3. Tables 2b and 2c show the vertical seeds and the horizontal seeds generated from the data set in Fig. 2a, which is repeated in Table 2a for convenience.

TABLE 2
Example ($\delta = 1, M_G = M_E = 3$)

	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)

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

(b)

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

(c)

(a) Data matrix, (b) horizontal PMBs, and (c) vertical PMBs.

3.3 Representation of Vertical Seeds

Details on the representation of horizontal seeds will be provided in Section 5.1. Here, we explain how to represent vertical seeds. In particular, we utilize *zero-suppressed binary decision diagrams* (ZBDDs) [19], 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 $\{G_{max}^{\{e_m,e_n\}}\}$ 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 2.3.2 hold true. Hence, the symbolic representation using ZBDDs is more compact than the traditional data structures for sets. Furthermore, as shown in Section 5.2.2, the manipulation of vertical seeds, such as union and intersection, is implicitly performed on ZBDDs, thus resulting in high efficiency. Refer to Section 2.3.2 for details on how to construct ZBDDs.

Example 4. In Table 2c, we showed the vertical seeds for our running example. The ZBDD in Fig. 7a represents the set of vertical seeds $\{G_{max}^{\{e_0,e_3\}}\} = \{\{g_0, g_2, g_3, g_4\}, \{g_1, g_3, g_4\}\}$.

Example 5. The ZBDD representation of $\{G_{max}^{\{e_3,e_5\}}\} = \{\{g_0, g_1, g_2, g_3\}, \{g_0, g_1, g_4\}\}$ is shown in Fig. 7b, along with that of $\{G_{max}^{\{e_2,e_5\}}\} = \{\{g_1, g_3, g_4\}\}$.

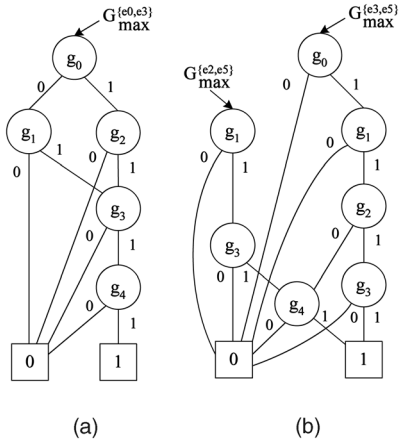


Fig. 7. ZBDDs for vertical seeds.

4 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.

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

Proof. Assume $G \supset G_s$ for all $G_s \in \{G_{max}^{\{e_m,e_n\}}\}$. Since (G, E) is a bicluster and $E \supseteq \{e_m, e_n\}$, its subbicluster $(G, \{e_m, e_n\})$ is also a bicluster for a given value of δ. By definition, if $G_s \in \{G_{max}^{\{e_m,e_n\}}\}$, 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 $G_s \in \{G_{max}^{\{e_m,e_n\}}\}$ must be false. Therefore, there must be at least one instance of $G_s \in \{G_{max}^{\{e_m,e_n\}}\}$ such that $G \subseteq G_s$. □

Example 6. Consider bicluster #1 in Fig. 2b, in which $G = \{g_0, g_2, g_3\}$ and $E = \{e_1, e_3, e_5\}$. From Table 2c, $\{G_{max}^{\{e_3,e_5\}}\} = \{\{g_0, g_1, g_4\}, \{g_0, g_1, g_2, g_3\}\}$. There exists $G_s \in \{G_{max}^{\{e_3,e_5\}}\}$ 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 above and is not presented here.)

Proposition 2. *Let (G, E) be a bicluster. If $G \supseteq \{g_i, g_j\}$, then there exists $E_s \in \{E_{max}^{\{g_i,g_j\}}\}$ such that $E \subseteq E_s$.*

Example 7. Consider bicluster #0 in Fig. 2b, in which $G = \{g_0, g_2, g_3, g_4\}$ and $E = \{e_0, e_1, e_3\}$. From Table 2b, $\{E_{max}^{\{g_0,g_3\}}\} = \{\{e_0, e_1, e_3\}, \{e_1, e_3, e_5\}\}$. There exists $E_s \in \{E_{max}^{\{g_0,g_3\}}\}$ such that $E \subseteq E_s$. That is, $E \subseteq E_s = \{e_0, e_1, e_3\}$.

4.2 Relationship between G and E

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

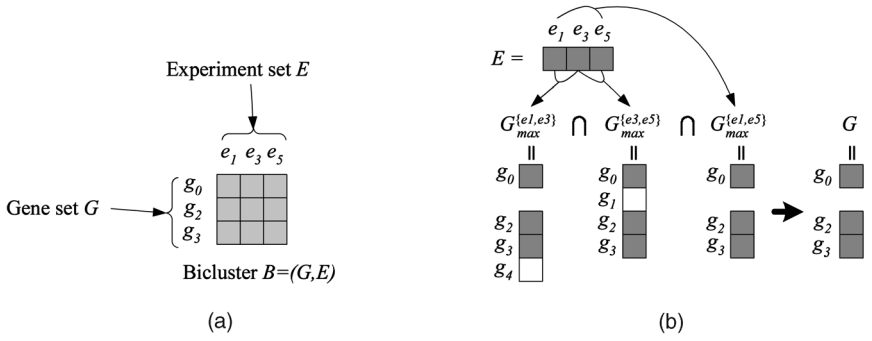


Fig. 8. Relationship between G and E in a Q-bicluster $B = (G, E)$. (a) Definitions. (b) Deriving G from E.

$$\mathcal{A} \otimes \mathcal{B} = \{I | I = A \cap B,\ \forall A \in \mathcal{A} \text{ and } \forall B \in \mathcal{B}\}. \quad (4)$$

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 1. We use bicluster #1 in Fig. 2b, which is depicted in Fig. 8a. By Proposition 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 Fig. 8b, $G = G_1 \cap G_2 \cap G_3$, 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:

$$\mathcal{G} = \{\text{all } G \text{ derivable from } E\} \quad (5)$$

$$= \bigotimes_{\forall (e_m,e_n) \in E} \left\{G_{max}^{\{e_m,e_n\}}\right\}, \quad (6)$$

and by symmetry,

$$\mathcal{E} = \{\text{all } E \text{ derivable from } G\} \quad (7)$$

$$= \bigotimes_{\forall (g_i,g_j) \in G} \left\{E_{max}^{\{g_i,g_j\}}\right\}. \quad (8)$$

In this work, we use (6) but not (8) because, in most gene expression data, $|U_E| \ll |U_G|$, which makes the evaluation of (6) much faster.

5 Our Biclustering Algorithm

We first repeat the problem statement presented in Section 2.4. 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 1) $G \subseteq U_G$ and $E \subseteq U_E$ and 2) 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 1) B is not too small, namely, $|G| \geq M_G$ and $|E| \geq M_E$ and 2) 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 Fig. 9. Sections 4.1 and 4.2 together suggest a way to derive biclusters from PMBs: We first determine E from the horizontal seeds and then compute G by (6) with reference to the vertical seeds. This idea is elaborated upon in this section.

Fig. 10 shows a flowchart of the proposed method. As described in Section 3.2, Algorithm 1 is used to find vertical and horizontal PMBs. Section 3.3 already presented how to represent vertical seeds by ZBDDs. The representation for horizontal seeds will be explained in Section 5.1. Algorithm 2, presented in Section 5.1, is used to predict E from horizontal seeds. Section 5.2 describes Algorithm 3, which can derive G from E by (6). For very large-scale gene expression data, we can optionally split input data by the method introduced in Section 5.3 before starting Algorithms 2 and 3.

5.1 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 2 in Section 4.1. 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 technique to avoid exhaustive enumeration of subsets in Algorithm 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 2 with the parameters $\delta = 1$ and $M_G = M_E = 3$.

5.1.1 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 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 $G_{max}^{\{e_m,e_n\}} \supseteq \{g_i, g_j\}$.*

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 1 does not generate $G_{max}^{\{e_m,e_n\}}$ at all if it has less than M_G elements.

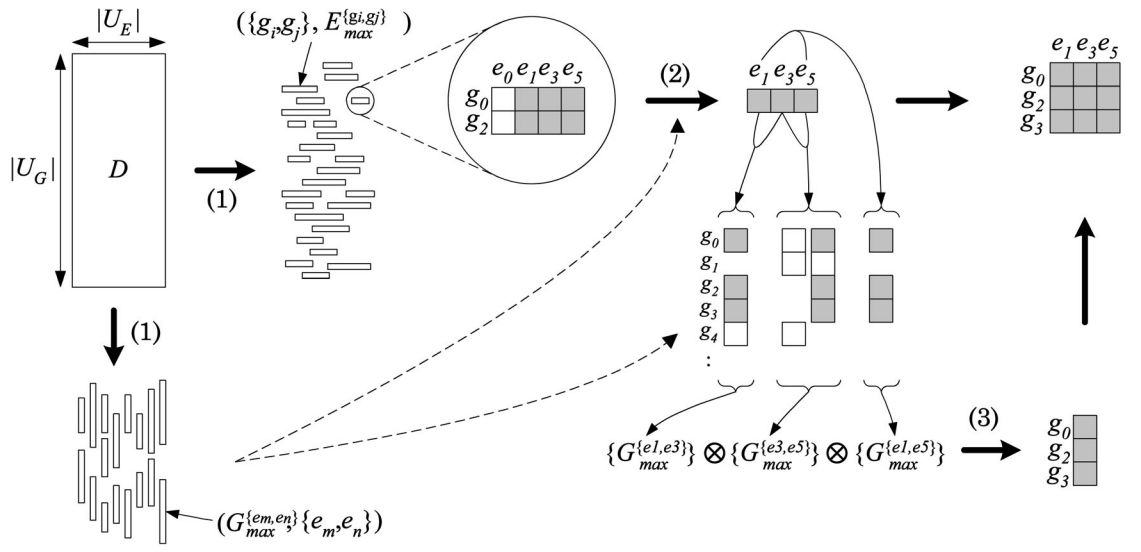


Fig. 9. Overview. (1) Generating horizontal and vertical PMBs: Section 3.2. (2) Predicting the experiment set E: Section 5.1. (3) Calculating the gene set G from E: Section 5.2.

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 (6) to them (Condition 2).

Example 8. Let E_1 and E_2 be the two instances of the seed $E^{\{g_2,g_4\}}_{max}$ in Table 2b. 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 because there is at least one instance of each $G^{\{e_0,e_1\}}_{max}$, $G^{\{e_1,e_3\}}_{max}$, and $G^{\{e_0,e_3\}}_{max}$ containing $\{g_2, g_4\}$. In contrast, E_2 is not valid because $G^{\{e_1,e_2\}}_{max}$ and $G^{\{e_2,e_3\}}_{max}$ 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 $E^{\{g_i,g_j\}}_{max}$ 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 $G^{\{e_m,e_n\}}_{max}$ such that $G^{\{e_m,e_n\}}_{max} \supseteq \{g_i, g_j\}$, as described in **Lines 3-7** of Algorithm 2. Then, a valid subset of $E^{\{g_i,g_j\}}_{max}$ corresponds to a clique (or complete subgraph) of at least M_E vertices.

Example 9. Figs. 11a and 11b 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 [9]. 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 removes these elements. If removing them from the set $E^{\{g_i,g_j\}}_{max}$ makes it contain fewer than M_E elements, then we eliminate the whole $E^{\{g_i,g_j\}}_{max}$ (**Lines 9-10**).

Algorithm 2: Predicting experiment set E

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

1 */* Step 1: removing extra elements */*
2 **foreach** $E^{\{g_i,g_j\}}_{max}$ **do**
3 $\quad \mathbf{V} = E^{\{g_i,g_j\}}_{max}$;
4 $\quad \mathbf{E} = \{\}$;
5 $\quad$ **foreach** (e_m, e_n) *in* $\mathbf{V}$ **do**
6 $\quad\quad$ **if** $\exists G^{\{e_m,e_n\}}_{max} \supseteq \{g_i, g_j\}$ **then** $\mathbf{E} = \mathbf{E} \cup \{(e_m, e_n)\}$;
7 $\quad$ Construct an undirected graph $(\mathbf{V}, \mathbf{E})$;
8 $\quad E^{\{g_i,g_j\}}_{max} = \mathbf{V} - \{$the vertices with the degree less than $M_E - 1\}$;
9 $\quad$ **if** $E^{\{g_i,g_j\}}_{max}$ *has less than* M_E *experiments* **then**
10 $\quad\quad$ Remove $E^{\{g_i,g_j\}}_{max}$;
11 */* Step 2: representing horizontal seeds by trie */*
12 **foreach** $E^{\{g_i,g_j\}}_{max}$ **do**
13 $\quad$ Sort the elements in $E^{\{g_i,g_j\}}_{max}$;
14 $\quad$ Locate node n whose path is specified by the ordered elements;
15 $\quad n.G = n.G \cup \{g_i, g_j\}$;
16 $\quad n.E = E^{\{g_i,g_j\}}_{max}$;
17 */* Step 3: predicting experiment set E from horizontal seeds */*
18 **foreach** *node* n *in the post-order traversal of the trie* **do**
19 $\quad$ Set $m.G = m.G \cup n.G$ for every node m in which $|m.E| = |n.E| - 1$ and $|m.E| \geq M_E$;
20 */* Step 4: eliminating invalid predictions */*
21 **foreach** *node* n *in the pre-order traversal of the trie* **do**
22 $\quad$ **if** $|n.G| < M_G$ **then** Remove n and its subtrie rooted at n;
23 */* Step 5: collecting exposed biclusters */*
24 **foreach** *node* n *in the pre-order traversal of the trie* **do**
25 $\quad$ **if** $(n.G, n.E)$ *is a bicluster* **then** Collect $(n.G, n.E)$;

Example 10. In Fig. 11b, 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 $E^{\{g_i,g_j\}}_{max}$ 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.

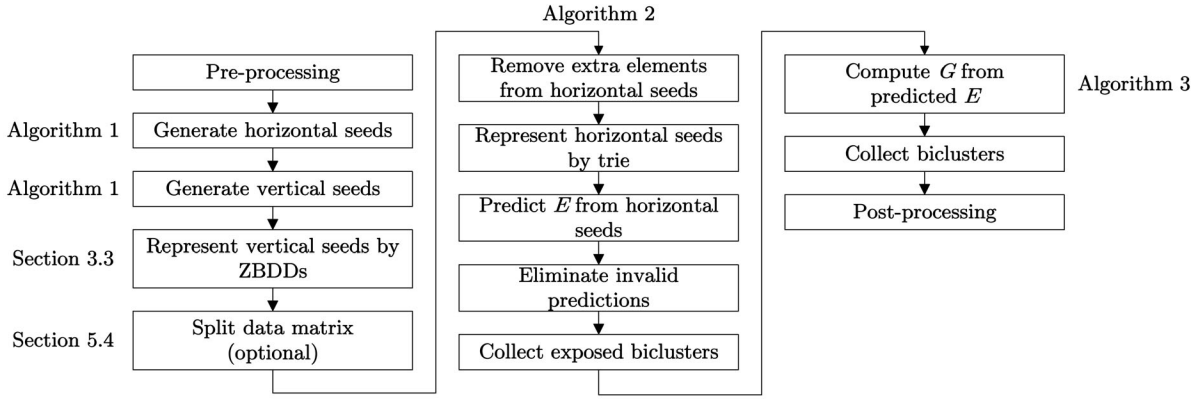


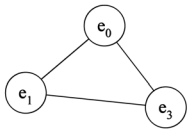
Fig. 10. Overall flow of the proposed method.

Example 11. Fig. 11c 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.

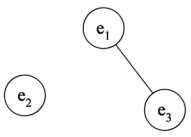
5.1.2 Step 2: Representation of Horizontal Seeds by a Trie

In **Lines 12-16** of Algorithm 2, the horizontal seeds are collectively represented by a *trie*, a data structure to represent sets of character strings [1]. 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.



(a)



(b)

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)

Fig. 11. Example for Step 1. (a) $E_{max}^{\{g_2,g_4\}} = \{e_0, e_1, e_3\}$. (b) $E_{max}^{\{g_2,g_4\}} = \{e_1, e_2, e_3\}$. (c) Horizontal seeds after Step 1.

Suppose that the seed $E_{max}^{\{g_i,g_j\}}$ is to be inserted into 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 12. Fig. 12a shows the trie representation of the horizontal seeds in Fig. 11c. For instance, $E_{max}^{\{g_0,g_2\}} = \{e_0, e_1, e_3, e_5\}$ and $E_{max}^{\{g_2,g_3\}} = \{e_0, e_1, e_3, e_5\}$ 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.

5.1.3 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 *postorder* traversal of the trie, the gene set $n.G$ is distributed to every node m in which $|m.E| = |n.E| - 1$ and $|m.E| \geq M_E$. Fig. 12b shows the trie representation after Step 3 is performed on the trie in Fig. 12a with $M_E = 3$.

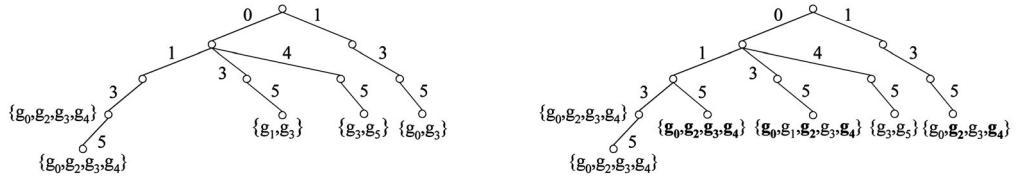
After Step 3, the set $n.G$ represents an upper bound of the gene set that can form a bicluster with the experiment set $n.E$.

5.1.4 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 *preorder* traversal of the trie. Genes were distributed in postorder 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. Fig. 13a shows the trie after this step for the running example.

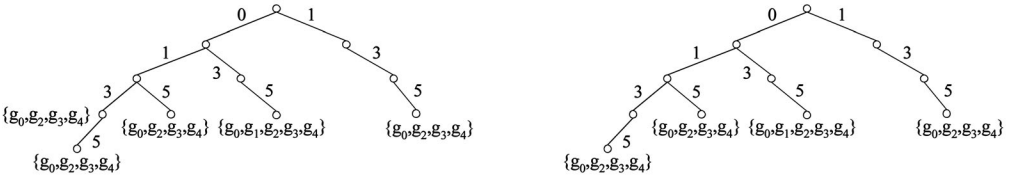
5.1.5 Step 5: Collecting Exposed Biclusters

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



(a) (b)

Fig. 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 Fig. 11c. (b) Predicted E sets. The trie has been expanded to examine possible experiment sets. $M_E = 3$.



(a) (b)

Fig. 13. Continuation of the trie example in Fig. 12. (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 Fig. 2b.

$$\begin{bmatrix} x & y \\ z & w \end{bmatrix}$$

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). Fig. 13b presents the trie after this step. Bicluster #0 in Fig. 2b is found here.

The biclusters found in this step are only a by-product of our algorithm to predict experiment sets. In Section 5.2, we describe our main approach that derives gene set G from experiment set E by (6).

5.2 Calculating the Gene Set G

After completing Steps 1-5 of Algorithm 2, we apply (6) 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 (6). 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.

5.2.1 Reducing the Number of $\otimes$ Operations

We use the following example to introduce this idea.

Example 13. Suppose that (6) 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 (6) to $E' = \{e_0, e_1, e_3\}$ and saved the result into $\mathcal{G}'$, then we only need the following three $\otimes$ operations:

$$\mathcal{G}' \otimes \left(\left\{G_{max}^{\{e_0,e_5\}}\right\} \otimes \left\{G_{max}^{\{e_1,e_5\}}\right\} \otimes \left\{G_{max}^{\{e_3,e_5\}}\right\}\right). \quad (9)$$

In general, when applying (6) 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 shows the outline of our approach.

Algorithm 3: Calculating gene set G

input : The trie from Algorithm 2; Vertical seeds
output: Maximal biclusters

```
1  foreach node n in the pre-order traversal of the trie do
2      if |n.E| == 2 then
3          n.G = {G_max^{n.E}};
4      else
5          E = n.E;
6          E' = n'.E, where n' is the parent node of n;
7          e_n = E − E';
8          foreach e in E' do
9              n.G = n.G ⊗ {G_max^{e,e_n}};
10         n.G = n.G ⊗ n'.G;
11     if n.G = ∅ then
12         Remove n and its subtrie rooted at n;
13     else if |E| ≥ M_E then
14         foreach G in n.G do
15             Collect (G, E);
16     Report maximal biclusters;
```

In the algorithm, each node n is associated with $n.\mathcal{G}$, a set of gene sets. For each node n visited in the *preorder* traversal of the trie, we perform the following procedure. If the set $n.E$ has

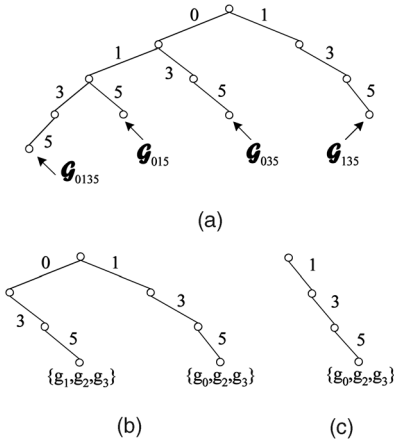


Fig. 14. Continuation of the trie example in Fig. 13. Bicluster #1 in Fig. 2b is found from the trie in (c).

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 (9) $\left(\left\{G_{max}^{\{e_0,e_5\}}\right\} \otimes \left\{G_{max}^{\{e_1,e_5\}}\right\} \otimes \left\{G_{max}^{\{e_3,e_5\}}\right\}\right)$ in Example 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 14. In Fig. 14a, the intermediate results for the leaves are indicated by $\mathcal{G}_{0135}$, $\mathcal{G}_{015}$, $\mathcal{G}_{035}$, and $\mathcal{G}_{135}$. Here,

$$\begin{aligned}\mathcal{G}_{0135} &= \left\{G_{max}^{\{e_0,e_5\}}\right\} \otimes \left\{G_{max}^{\{e_1,e_5\}}\right\} \otimes \left\{G_{max}^{\{e_3,e_5\}}\right\} \\ &= \{\{g_0\},\{g_0,g_2\},\{g_2,g_3\}\}.\end{aligned}$$

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). Fig. 14b finally shows the trie in which the two left leaves are removed from the trie in Fig. 14a.

In **Line 10**, the resulting $\mathcal{G}_n$, corresponding to (9) 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**). Fig. 14c presents the final trie in which bicluster #1 in Fig. 2b is found at the leaf node.

5.2.2 Efficient Implementation of $\otimes$

The operator $\otimes$ is implemented using the basic set operators on ZBDDs, such as $\cap$ and $\cup$. These operators are recursively defined on ZBDDs with trivial terminal cases, such as $\mathcal{P} \cap \emptyset = \emptyset$ or $\mathcal{P} \cup \emptyset = \mathcal{P}$. For more details, we refer the reader to [19], [20].

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 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 2c. 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 Fig. 15a. The problem of $\mathcal{P} \cup \mathcal{Q}$ can now become two smaller problems, $(\mathcal{P}_0 \cup \mathcal{Q}_0)$ and $(\mathcal{P}_1 \cup \mathcal{Q}_1)$.

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 $(\mathcal{P}_1 \otimes \mathcal{Q}_1)$. Fig. 15b 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.

Example 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 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 Fig. 15b, 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.

5.3 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.

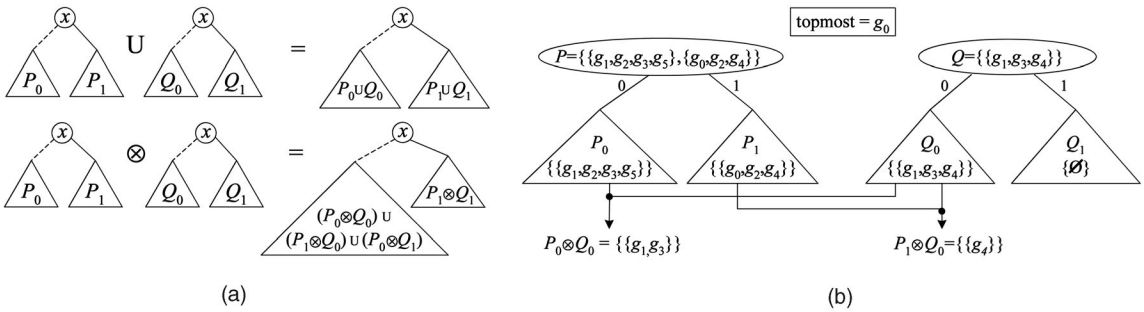


Fig. 15. The operators $\cup$ and $\otimes$ on ZBDDs. (a) The set operators are recursively defined on the ZBDDs. The operator $\cap$ is defined in the same way as the operator $\otimes$, but is not shown here. (b) An example.

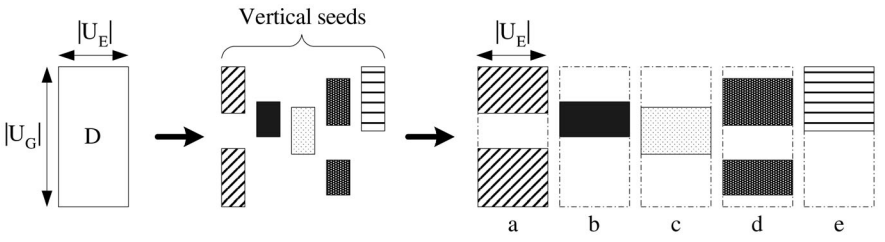


Fig. 16. Dividing a large data matrix into submatrices of manageable sizes.

TABLE 3
Algorithm Parameters for the Experiments

Experiments		Algorithms and parameters									
Fig.	Data	Our method			δ-biclustering [8]		pClustering [31]			GEMS [32]	
		δ	M_G	M_E	α	δ	δ	nr	nc	a	w
17(a)	Synthetic [31]	0	10	5	1.2	1000	0	30	5	0.08	200
17(b)	Yeast [30]	75–80	80	12	1.2	300	10–15	80	10–12	0.25	15–30
18(a)	Yeast [30]	90	150	11	1.2	300	2	40	6	-	
18(b)	B cell [2]	70	65	7	1.2	1200		-		-	
19	Yeast [30]	80	28	13	1.2	300		-		-	

(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 $(G_{max}^{\{e_m,e_n\}}, U_E)$ for all $\{e_m, e_n\}$ in U_E.

Example 17. In Fig. 16, 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 motivation 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. 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.

5.4 Algorithm Complexity

The problem of biclustering is inherently intractable [8], [29], [31], 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 Section 6. 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 [19], [20], [10], [18], [5], [6], [25].

Additionally, our algorithm works in polynomial time if the input data matrix and parameters satisfy a certain condition. We refer the interested reader to [34] for more details on this condition.

6 Experimental Results

We performed experimental studies to analyze the performance of our algorithm on several benchmarks, including both synthetic and real expression data sets. We implemented our method[2] in ANSI C++, and for the generation and manipulation of the ZBDDs, we utilized the CUDD[3] and EXTRA[4] packages. We conducted our experiments on a 3.02 GHz Linux machine with 4 GB RAM. We listed the specific algorithm parameters for each experiment in Table 3.

6.1 Evaluation of Algorithm Efficiency

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 created these synthetic data sets by the method introduced in [31]. Initially, a matrix is filled with random values ranging from 0 to 500, and then a fixed number of *perfect* pClusters are embedded. (*Perfect* pClusters are those that can be found with $\delta = 0$.) We created matrices of 100 columns and five different rows (1K, 3K, 6K, 9K, and 12K) to test the scalability of various algorithms, including an alternative implementation of our method that uses a classical data structure for sets instead of ZBDDs. The results are presented in Fig. 17a.

2. http://akebono.stanford.edu/users/sryoon/tcbb04.
3. http://vlsi.colorado.edu/~fabio/CUDD/cuddIntro.html.
4. http://www.ee.pdx.edu/~alanmi/research/extra.htm.

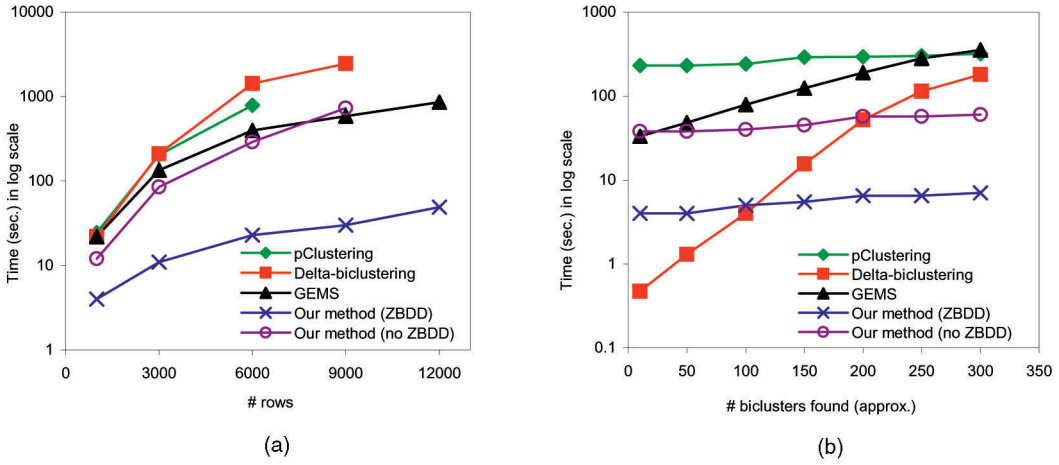


Fig. 17. Running time comparison of several biclustering algorithms: δ-biclustering [8], pClustering [31], GEMS [32], and our method. (a) Synthetic data [31]. (b) Yeast cell cycle data [30].

Next, we compared the time needed to produce the first n biclusters from the yeast data [30] by our method and the other techniques. This data set has 2,884 genes and 17 conditions obtained from the yeast *Saccharomyces cerevisiae* cell cycle expression levels. In Fig. 17b, the abscissa is the number of biclusters produced and the ordinate is the time required to find these biclusters. Our method and the pClustering method do not take as input the exact number of biclusters to generate. Thus, we run these algorithms multiple times with different values of δ to find approximately n biclusters.

Through the above experiments, we established that our method tends to outperform the others in terms of running time for both the synthetic and the real gene expression data.

6.2 Statistical and Biological Validation

In Section 6.2.1, we use *correspondence plots* [29] to assess the statistical significance of the biclusters produced by our algorithm. In addition, we utilize the *receiver operating characteristic* (ROC) curves [11], [24], [12] for two more validation studies. In Section 6.2.2, we first use the ROC curves to qualitatively evaluate the ability of our algorithm to produce biclusters conforming to prior biological knowledge. Section 6.2.3 then presents a statistical comparison of the biclusters produced by our method and the δ-biclusters [8] in terms of the ROC curves.

6.2.1 Correspondence Plot

To statistically validate the biclusters produced, we employed the technique proposed by Tanay et al. [29], which takes advantage of a known (putatively correct) classification of genes or experimental conditions. Suppose prior knowledge classifies N genes into K classes, $C_1, C_2, \ldots C_K$. Let B be a bicluster with g genes and assume that out of those g genes, g_j genes belong to class C_j. Assuming the most abundant class for B is C_i, the hypergeometric distribution is used to calculate the p-value of bicluster B:

$$p(B) = \Sigma_{k=g_i}^{g} \frac{\binom{|C_i|}{k}\binom{N-|C_i|}{g-k}}{\binom{N}{g}}.$$

That is, the p-value corresponds to the probability of obtaining at least g_i elements of the class C_i in a random set of size g.

In the correspondence plot, the early departure of a curve from the x-axis indicates the existence of biclusters with low p-values. Consequently, the area under a curve shows the approximate degree of statistical significance of the biclusters used to draw the curve.

Fig. 18a presents the correspondence plot for the biclusters generated by several different methods on the aforementioned yeast data set [30]. The plot also includes randomly generated biclusters. It 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.

6.2.2 ROC Curve for Algorithm Evaluation

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 [11], [24]. 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 [12].

The *area under the ROC curve* (AUC) is a reasonable summary of the overall diagnostic accuracy of the test [24]. 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.

Since our algorithm is unsupervised, we associate each bicluster with known classes as follows: Suppose prior knowledge classifies M samples into two classes, P and N.

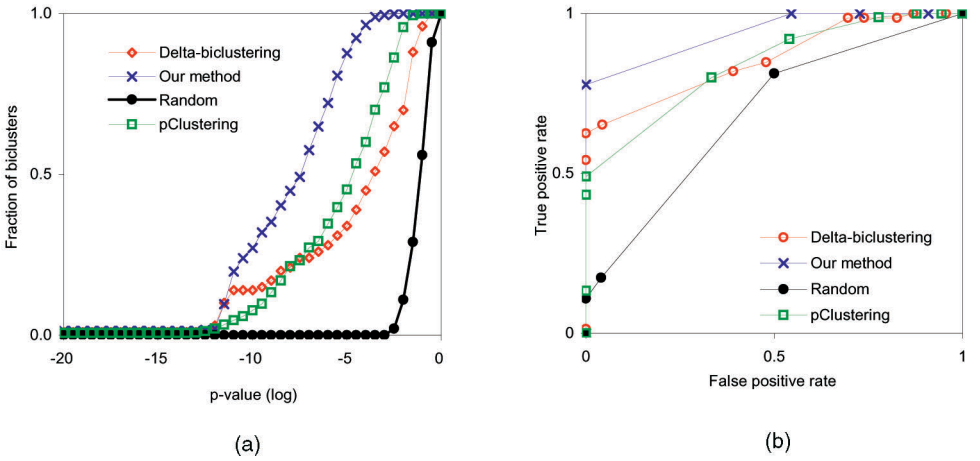


Fig. 18. (a) The correspondence plot depicts the distribution of p-values of the produced biclusters with respect to a known classification of experimental conditions or gene annotations. The plot presents the fraction of biclusters whose p-value is at most P out of the n best biclusters discovered, for each p-value P on the plot. We used the yeast cell cycle data [30]. (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 [2].

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.

Fig. 18b presents the ROC curves based upon the biclustering results from the B-cell lymphoma data set by Alizadeh et al. [2], which contains normal and cancerous cell-line samples for 4,026 genes. This result shows that our algorithm has a better characteristic in the ROC space than the alternative methods on this benchmark. Consequently, the biclusters produced by our method tend to represent more accurate classification of the genes or samples involved.

6.2.3 ROC Curve for Bicluster Evaluation

We next compared the biclusters found by our method from the yeast data set [30] with the δ-biclusters [8] found from the same data set. Using the ROC curve, it was possible to statistically distinguish these two sets of biclusters with respect to the 30 known categories (or groups) of yeast genes reported by Tavazoie et al. [30]. Moreover, our analysis suggests that the biclusters found by our method are in fact more compatible with the known categories of genes. The details are presented here.

ROC curves can be used to see if two populations are statistically different, and the departure of AUC from 0.5 indicates the degree of discrimination [11], [24], [12]. In our experiment, we constructed as follows one population (called *Population 1*) from the biclusters obtained by our algorithm and another population (called *Population 2*) from the δ-biclusters. We compared each bicluster found by our method with each of the 30 known gene groups; we used a χ^2 test for a 2×30 contingency table [24]. The contingency table lists 1) the distribution of the genes in a single bicluster found by our method over the known 30 gene categories by Tavazoie et al. and 2) the distribution of the genes in a single known gene group over the 30 known gene categories. A χ^2 test was performed on this table, and the corresponding χ^2 statistic was computed and added to *Population 1*. The lower the statistic, the more similar the two rows in the contingency table, and vice versa. *Population 2* was constructed similarly from the χ^2 scores obtained by the comparison of δ-biclusters with the known gene categories.

To focus more on the biclusters that best represented the known gene categories, we next selected, for each known gene category, one bicluster by our method and one δ-bicluster that had the lowest χ^2 scores. That is, we left only 30 χ^2 scores in each population. After this filtering, *Population 1* had a mean of 58.4 and a standard deviation of 44.0. The mean and standard deviation for *Population 2* were 96.5 and 34.4, respectively.

Finally, we produced an ROC curve out of these two populations as follows. We arbitrarily considered *Population 1* as the "control." We then merged the two populations into one and rank-ordered individual χ^2 scores. We examined each score in descending order, labeling it "False Positive" if it belonged to the control population and "True Positive" otherwise. Scanning 60 χ^2 scores in this way allowed us to produce points on the ROC plane, and by connecting these points, the ROC curve in Fig. 19 was obtained.

The AUC of this ROC curve is 0.75, indicating reasonable discrimination of the two populations. Combined with the fact that the 30 best biclusters discovered by our method tend to have lower χ^2 scores than the 30 best δ-biclusters (58.4 versus 96.5 on average), it appears that the biclusters obtained by our algorithm are more compatible with the known yeast gene groups reported by Tavazoie et al. [30].

7 Conclusion

In this study, we investigated the problem of finding coherent biclusters. We mathematically characterized this problem and proposed a suite of new algorithms to find biclusters with coherent values. Our 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 our method. We conducted experimental

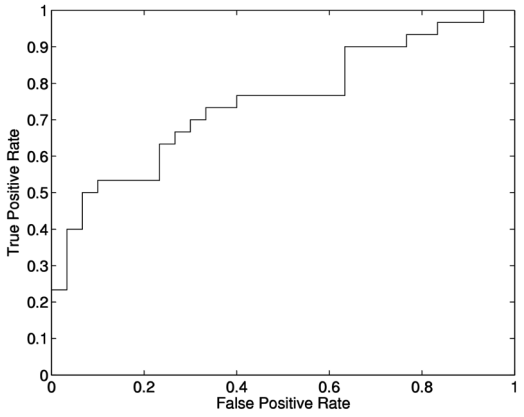


Fig. 19. ROC curve ($AUC = 0.75$) showing that our biclusters and the δ-biclusters [8] found from the yeast data set [30] can be discriminated against with respect to the known gene categories by Tavazoie et al. [30].

studies including statistical and biological validations of the biclusters produced by our method. The results demonstrated the effectiveness of our approach.

ACKNOWLEDGMENTS

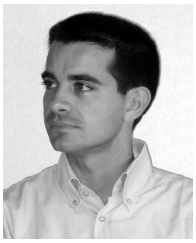
The authors would like to thank the anonymous reviewers of the manuscript for their valuable comments. S. Yoon would like to thank Giselle Sylvester for proofreading the manuscript. This work was supported by a grant of Jerry Yang and Akiko Yamazaki.

REFERENCES

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman, *Data Structures and Algorithms.* Reading, Mass.: Addison-Wesley, 1983.
[2] A. Alizadeh et al., "Distinct Types of Diffuse Large B-Cell Lymphoma Identified by Gene-Expression Profiling," *Nature,* vol. 4051, pp. 503-511, 2000.
[3] R.B. Altman and S. Raychaudhuri, "Whole-Genome Expression Analysis: Challenges beyond Clustering," *Current Opinion in Structural Biology,* vol. 11, pp. 340-347, 2001.
[4] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini, "Discovering Local Structure in Gene Expression Data: The Order-Preserving Submatrix Problem," *J. Computational Biology,* vol. 10, nos. 3-4, pp. 373-384, 2003.
[5] R.E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Trans. Computers,* vol. 35, no. 8, pp. 677-691, Aug. 1986.
[6] R.E. Bryant, "Binary Decision Diagrams and Beyond: Enabling Technologies for Formal Verification," *Proc. IEEE/ACM Int'l Conf. Computer Aided Design, (ICCAD),* pp. 236-243, 1995.
[7] A. Califano, G. Stolovitzky, and Y. Tu, "Analysis of Gene Expression Microarrays for Phenotype Classification," *Proc. Int'l Conf. Intelligent Systems for Molecular Biology,* pp. 75-85, 2000.
[8] Y. Cheng and G.M. Church, "Biclustering of Expression Data," *Proc. Conf. Intelligent Systems for Molecular Biology (ISMB),* pp. 93-103, 2000.
[9] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms.* Cambridge, Mass.: MIT Press, 2001.
[10] G. De Micheli, *Synthesis and Optimization of Digital Circuits.* New York: McGraw-Hill, 1994.
[11] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification.* New York: Wiley, second ed., 2001.
[12] T. Fawcett, "ROC Graphs: Notes and Practical Considerations for Data Mining Researchers," HP Laboratories technical report, 2003.
[13] G. Getz, E. Levine, and E. Domany, "Coupled Two-Way Clustering Analysis of Gene Microarray Data," *Proc. Nat'l Academy of Science,* vol. 94, pp. 12079-12084, 2000.
[14] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning.* New York: Springer-Verlag, 2001.
[15] Y. Kluger, R. Basri, J.T. Chang, and M. Gerstein, "Spectral Biclustering of Microarray Data: Coclustering Genes and Conditions," *Genome Research,* vol. 13, no. 4, pp. 703-716, Apr. 2003.
[16] L. Lazzeroni and A. Owen, "Plaid Models for Gene Expression Data," Stanford Univ. technical report, 2000.
[17] S.C. Madeira and A.L. Oliveira, "Biclustering Algorithms for Biological Data Analysis: A Survey," *IEEE/ACM Trans. Computational Biology and Bioinformatics,* vol. 1, no. 1, pp. 24-45, 2004.
[18] C. Meinel and T. Theobald, *Algorithms and Data Structures in VLSI Design.* Berlin: Springer, 1998.
[19] S. Minato, "Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems," *Proc. IEEE/ACM Design Automation Conf., (DAC),* pp. 272-277, 1993.
[20] S. Minato, *Binary Decision Diagrams and Applications for VLSI CAD.* Kluwer, 1996.
[21] T.M. Murali and S. Kasif, "Extracting Conserved Gene Expression Motifs from Gene Expression Data," *Proc. Pacific Symp. Biocomputing,* pp. 77-88, 2003.
[22] S. Raychaundhuri, P.D. Sutphin, J.T. Chang, and R.B. Altman, "Basic Microarray Analysis: Grouping and Feature Reduction," *Trends in Biotechnology,* vol. 19, no. 5, pp. 189-193, May 2001.
[23] J.A. Rice, *Mathematical Statistics and Data Analysis.* Duxbury Press, 1994.
[24] B. Rosner, *Fundamentals of Biostatistics.* fifth ed., Pacific Grove, Calif.: Duxbury, 2000.
[25] T. Sasao and M. Fujita, *Representations of Discrete Functions.* Mass.: Kluwer, 1996.
[26] E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller, "Rich Probabilistic Models for Gene Expression," *Bioinformatics,* vol. 17, pp. 243-52, 2001.
[27] R.R. Sokal and F.J. Rohlf, *Biometry.* WH Freeman and Co., 1994.
[28] 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,* vol. 18, pp. 111-119, 2002.
[29] A. Tanay, R. Sharan, and R. Shamir, "Discovering Statistically Significant Biclusters in Gene Expression Data," *Bioinformatics,* vol. 18, pp. 136-144, 2002.
[30] S. Tavazoie, J.D. Hughes, M.J. Campbell, R.J. Cho, and G.M. Church, "Systematic Determination of Genetic Network Architecture," *Nature Genetics,* vol. 22, pp. 281-285, 1999.
[31] H. Wang, W. Wang, J. Yang, and P.S. Yu, "Clustering by Pattern Similarity in Large Data Sets," *Proc. ACM SIGMOD Conf.,* pp. 394-405, 2002.
[32] C.-J. Wu, Y. Fu, T.M. Murali, and S. Kasif, "Gene Expression Module Discovery Using Gibbs Sampling," *Genome Informatics,* vol. 15, no. 1, pp. 239-248, 2004.
[33] J. Yang, H. Wang, W. Wang, and P. Yu, "Enhanced Biclustering on Expression Data," *Proc. IEEE Third Symp. Bioinformatics and Bioeng.,* pp. 321-327, 2003.
[34] S. Yoon, C. Nardini, L. Benini, and G. De Micheli, "Enhanced Pclustering and Its Applications to Gene Expression Data," *Proc. IEEE Fourth Symp. Bioinformatics and Bioeng.,* pp. 275-282, 2004.

Sungroh Yoon received the BS degree in electrical engineering from Seoul National Univ. in 1996 and the MS degree in electrical engineering from Stanford University in 2002. He is currently a PhD candidate in electrical engineering at Stanford University. His research interests include machine learning and its applications to computational biology and bioinformatics. He is a student member of the IEEE.

Christine Nardini is a PhD student in the Department of Electrical Engineering and Computer Science (DEIS) at the University of Bologna. Her research interests are in bioinformatics and, in particular, related to the processing of cDNA microarray data, from data mining to the applications oriented to clinical genomics.



Luca Benini is an associate professor in the Department of Electrical Engineering and Computer Science (DEIS) at the University of Bologna. He received the PhD degree in electrical engineering from Stanford University in 1997. Dr. Benini's research interests are in all aspects of computer-aided design of digital circuits, with special emphasis on low-power applications, and in the design of portable systems. On these topics, he has published more than 250 papers in international journals and conferences and three books. He has been program chair and vice-chair of Design Automation and Test in Europe Conference. He is a member of the technical program committee and organizing committee of several technical conferences, including the Design Automation Conference, International Symposium on Low Power Design, and the Symposium on Hardware-Software Codesign. He is a senior member of the IEEE and the IEEE Computer Society.

Giovanni De Micheli is a professor and director of the Integrated Systems Center at EPF Lausanne, Switzerland. Previously, he was a professor of electrical engineering at Stanford University. His research interests include several aspects of design technologies for integrated systems on silicon, such as synthesis, hw/sw codesign and low-power design, as well as systems on heterogeneous platforms including electrical, optical, micromechanical, and biological components. He is the author of *Synthesis and Optimization of Digital Circuits* (McGraw-Hill, 1994), coauthor and/or coeditor of five other books, and of more than 300 technical articles. He is, or has been, a member of the technical advisory board of several companies, including Magma Design Automation, Coware, Aplus Design Technologies, IROC, Ambit Design Systems, and STMicroelectronics. Dr. De Micheli is the recipient of the 2003 IEEE Emanuel Piore Award for contributions to computer-aided synthesis of digital systems. He is a fellow of the IEEE, the IEEE Computer Society, and the ACM. He received the Golden Jubilee Medal for outstanding contributions to the IEEE CAS Society in 2000. He received the 1987 D. Pederson Award for the best paper in the *IEEE Transactions on CAD/ICAS*, two Best Paper Awards at the Design Automation Conference, in 1983 and in 1993, and a best paper award at the DATE Conference in 2005. He was president of the IEEE CAS Society in 2003. He was editor-in-chief of the *IEEE Transactions on CAD/ICAS* from 1987-2001. Dr. De Micheli was the program chair and general chair of the Design Automation Conference (DAC) from 1996-1997 and 2000, respectively. He was the program and general chair of the International Conference on Computer Design (ICCD) in 1988 and 1989, respectively.