Biclustering Algorithms for Biological Data Analysis: A Survey

Sara C. Madeira and Arlindo L. Oliveira

Abstract—A large number of clustering approaches have been proposed for the analysis of gene expression data obtained from microarray experiments. However, the results from the application of standard clustering methods to genes are limited. This limitation is imposed by the existence of a number of experimental conditions where the activity of genes is uncorrelated. A similar limitation exists when clustering of conditions is performed. For this reason, a number of algorithms that perform simultaneous clustering on the row and column dimensions of the data matrix has been proposed. The goal is to find submatrices, that is, subgroups of genes and subgroups of conditions, where the genes exhibit highly correlated activities for every condition. In this paper, we refer to this class of algorithms as biclustering. Biclustering is also referred in the literature as coclustering and direct clustering, among others names, and has also been used in fields such as information retrieval and data mining. In this comprehensive survey, we analyze a large number of existing approaches to biclustering, and classify them in accordance with the type of biclusters they can find, the patterns of biclusters that are discovered, the methods used to perform the search, the approaches used to evaluate the solution, and the target applications.

Index Terms—Biclustering, simultaneous clustering, coclustering, subspace clustering, bidimensional clustering, direct clustering, block clustering, two-way clustering, two-mode clustering, two-sided clustering, microarray data analysis, biological data analysis, gene expression data.

1 Introduction

DNA chips and other techniques measure the expression level of a large number of genes, perhaps all genes of an organism, within a number of different experimental samples (conditions) [5]. The samples may correspond to different time points or different environmental conditions. In other cases, the samples may have come from different organs, from cancerous or healthy tissues, or even from different individuals. Simply visualizing this kind of data, which is widely called *gene expression data* or, simply, *expression data*, is challenging and extracting biologically relevant knowledge is harder still [34].

Usually, gene expression data is arranged in a data matrix, where each gene corresponds to one row and each condition to one column. Each element of this matrix represents the expression level of a gene under a specific condition, and is represented by a real number, which is usually the logarithm of the relative abundance of the mRNA of the gene under the specific condition. Gene expression matrices have been extensively analyzed in two dimensions: the gene dimension and the condition dimension. These analysis correspond, respectively, to analyze the expression patterns of genes by comparing the rows in the matrix, and to analyze the expression patterns of samples by comparing the columns in the matrix.

Common objectives pursued when analyzing gene expression data include:

1. Grouping of genes according to their expression under multiple conditions.
2. Classification of a new gene, given the expression of other genes, with known classification.
3. Grouping of conditions based on the expression of a number of genes.
4. Classification of a new sample, given the expression of the genes under that experimental condition.

Clustering techniques can be used to group either genes or conditions and, therefore, to pursue directly objectives 1 and 3 above and, indirectly, objectives 2 and 4. However, applying clustering algorithms to gene expression data runs into a significant difficulty. Many activation patterns are common to a group of genes only under specific experimental conditions. In fact, our general understanding of cellular processes leads us to expect subsets of genes to be coregulated and coexpressed only under certain experimental conditions, but to behave almost independently under other conditions. Discovering such local expression patterns may be the key to uncovering many genetic pathways that are not apparent otherwise. It is therefore highly desirable to move beyond the clustering paradigm, and to develop approaches capable of discovering local patterns in microarray data [6].

The term biclustering was first used by Cheng and Church [10] in gene expression data analysis. It refers to a distinct class of clustering algorithms that perform simultaneous row-column clustering. Biclustering algorithms have also been proposed and used in other application fields. Names such as coclustering, bidimensional clustering, and subspace clustering, among others, are often used in the literature to refer to the same problem formulation. One of the earliest biclustering formulations is the direct clustering

- *S.C. Madeira is with the University of Beira Interior, Rua Marquês D'Ávila e Bolama, 6200-001 Covilhã, Portugal. She is also with INESC-ID, Lisbon, Portugal. E-mail: smadeira@di.ubi.pt.*
- *A.L. Oliveira is with the Instituto Superior Técnico, Lisbon Technical University, Rua Alves Redol 9, Apartado 13069, 1000-029 Lisbon, Portugal. E-mail: aml@inesc-id.pt.*

Manuscript received 29 Jan. 2004; revised 19 May 2004; accepted 14 June 2004.
For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBB-0009-0104.

1545-5963/04/$20.00 © 2004 IEEE Published by the IEEE CS, NN, and EMB Societies & the ACM

algorithm introduced by Hartigan [24], also known as block clustering [36].

What is then the difference between clustering and biclustering? Why and when should we use biclustering instead of clustering? Clustering can be applied to either the rows or the columns of the data matrix, separately. Biclustering, on the other hand, performs clustering in these two dimensions simultaneously. This means that clustering derives a *global model* while biclustering produces a *local model*. When clustering algorithms are used, each gene in a given gene cluster is defined using all the conditions. Similarly, each condition in a condition cluster is characterized by the activity of all the genes that belong to it. However, each gene in a bicluster is selected using only a subset of the conditions and each condition in a bicluster is selected using only a subset of the genes. The goal of biclustering techniques is thus to identify subgroups of genes and subgroups of conditions, by performing simultaneous clustering of both rows and columns of the gene expression matrix, instead of clustering these two dimensions separately. We can then conclude that, unlike clustering algorithms, biclustering algorithms identify groups of genes that show similar activity patterns under a specific *subset* of the experimental conditions. Therefore, biclustering approaches are the key technique to use when one or more of the following situations applies:

1. Only a small set of the genes participates in a cellular process of interest.
2. An interesting cellular process is active only in a subset of the conditions.
3. A single gene may participate in multiple pathways that may or not be coactive under all conditions.

For these reasons, biclustering should identify groups of genes and conditions, obeying the following restrictions:

1. A cluster of genes should be defined with respect to only a subset of the conditions.
2. A cluster of conditions should be defined with respect to only a subset of the genes.
3. The clusters should not be exclusive and/or exhaustive: A gene/condition should be able to belong to more than one cluster or to no cluster at all and be grouped using a subset of conditions/genes.

Additionally, robustness in biclustering algorithms is especially relevant because of two additional characteristics of the systems under study. The first characteristic is the sheer complexity of gene regulation processes that require powerful analysis tools. The second characteristic is the level of noise in actual gene expression experiments that makes the use of intelligent statistical tools indispensable.

2 Definitions and Problem Formulation

We will be working with an n by m data matrix, where each element a_{ij} will be, in general, a given real value. In the case of gene expression matrices, a_{ij} represents the expression level of gene i under condition j. Table 1 illustrates the arrangement of a gene expression matrix.

TABLE 1
Gene Expression Data Matrix

	Condition 1	...	Condition j	...	Condition m
Gene 1	a_{11}	...	a_{1j}	...	a_{1m}
Gene ...	...	...	...	...	...
Gene i	a_{i1}	...	a_{ij}	...	a_{im}
Gene ...	...	...	...	...	...
Gene n	a_{n1}	...	a_{nj}	...	a_{nm}

A large fraction of applications of biclustering algorithms deal with gene expression matrices. However, there are many other applications for biclustering. For this reason, we will consider the general case of a data matrix, A, with set of rows X and set of columns Y, where the element a_{ij} corresponds to a value representing the relation between row i and column j. Such a matrix A, with n rows and m columns, is defined by its set of rows, $X = \{x_1, \dots, x_n\}$, and its set of columns, $Y = \{y_1, \dots, y_m\}$. We will use (X, Y) to denote the matrix A. Considering that $I \subseteq X$ and $J \subseteq Y$ are subsets of the rows and columns, respectively, $A_{IJ} = (I, J)$ denotes the submatrix of A that contains only the elements a_{ij} belonging to the submatrix with set of rows I and set of columns J.

Given the data matrix A, as defined above, we define a *cluster of rows* as a subset of rows that exhibit similar behavior across the set of all columns. This means that a row cluster $A_{IY} = (I, Y)$ is a subset of rows defined over the set of all columns Y, where $I = \{i_1, \dots, i_k\}$ is a subset of rows ($I \subseteq X$ and $k \leq n$). A cluster of rows (I, Y) can thus be defined as a k by m submatrix of the matrix A.

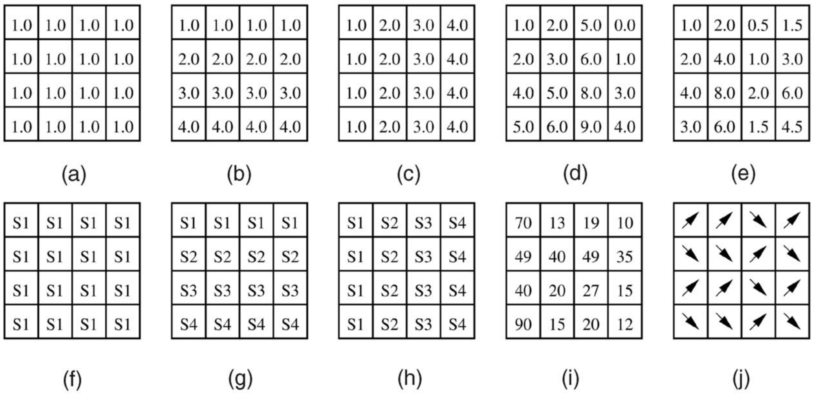
Similarly, a *cluster of columns* is a subset of columns that exhibit similar behavior across the set of all rows. A column cluster $A_{XJ} = (X, J)$ is a subset of columns defined over the set of all rows X, where $J = \{j_1, \dots, j_s\}$ is a subset of columns ($J \subseteq Y$ and $s \leq m$). A cluster of columns (X, J) can then be defined as an n by s submatrix of the matrix A.

A *bicluster* is a subset of rows that exhibit similar behavior across a subset of columns, and vice versa. The bicluster $A_{IJ} = (I, J)$ is thus a subset of rows and a subset of columns where $I = \{i_1, \dots, i_k\}$ is a subset of rows ($I \subseteq X$ and $k \leq n$), and $J = \{j_1, \dots, j_s\}$ is a subset of columns ($J \subseteq Y$ and $s \leq m$). A bicluster (I, J) can be defined as a k by s submatrix of the matrix A.

The specific problem addressed by biclustering algorithms can now be defined. Given a data matrix, A, we want to identify a set of biclusters $B_k = (I_k, J_k)$ such that each bicluster B_k satisfies some specific characteristics of homogeneity. The exact characteristics of homogeneity vary from approach to approach, and will be studied in Section 3.

2.1 Weighted Bipartite Graph and Data Matrices

An interesting connection between data matrices and graph theory can be established. A data matrix can be viewed as a *weighted bipartite graph*. A graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, is said to be bipartite if its vertices can be partitioned into two sets L and R such that every edge in E has exactly one end in L and the other in R: $V = L \bigcup R$. The data matrix $A = (X, Y)$ can be viewed as a weighted bipartite graph where each node $n_i \in L$ corresponds to a row and each node $n_j \in R$ corresponds to a column. The edge between node n_i and n_j has weight a_{ij}, denoting the element of the matrix in the intersection between row i and column j (and the strength of the activation level, in the case of gene expression matrices).



(a)

1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0

(b)

1.0	1.0	1.0	1.0
2.0	2.0	2.0	2.0
3.0	3.0	3.0	3.0
4.0	4.0	4.0	4.0

(c)

1.0	2.0	3.0	4.0
1.0	2.0	3.0	4.0
1.0	2.0	3.0	4.0
1.0	2.0	3.0	4.0

(d)

1.0	2.0	5.0	0.0
2.0	3.0	6.0	1.0
4.0	5.0	8.0	3.0
5.0	6.0	9.0	4.0

(e)

1.0	2.0	0.5	1.5
2.0	4.0	1.0	3.0
4.0	8.0	2.0	6.0
3.0	6.0	1.5	4.5

(f)

S1	S1	S1	S1
S1	S1	S1	S1
S1	S1	S1	S1
S1	S1	S1	S1

(g)

S1	S1	S1	S1
S2	S2	S2	S2
S3	S3	S3	S3
S4	S4	S4	S4

(h)

S1	S2	S3	S4
S1	S2	S3	S4
S1	S2	S3	S4
S1	S2	S3	S4

(i)

70	13	19	10
49	40	49	35
40	20	27	15
90	15	20	12

(j)

↗	↗	↘	↗
↘	↘	↗	↘
↗	↗	↘	↗
↘	↘	↗	↘

Fig. 1. Examples of different types of biclusters. (a) Constant bicluster, (b) constant rows, (c) constant columns, (d) coherent values (addictive model), (e) coherent values (multiplicative model), (f) overall coherent evolution, (g) coherent evolution on the rows, (h) coherent evolution on the columns, (i) coherent evolution on the columns, and (j) coherent sign changes on rows and columns.

This connection between matrices and graph theory leads to very interesting approaches to the analysis of expression data based on graph algorithms.

2.2 Problem Complexity

Although the complexity of the biclustering problem may depend on the exact problem formulation and, specifically, on the merit function used to evaluate the quality of a given bicluster, almost all interesting variants of this problem are NP-complete. In its simplest form the data matrix A is a binary matrix, where every element a_{ij} is either 0 or 1. When this is the case, a bicluster corresponds to a biclique in the corresponding bipartite graph. Finding a maximum size bicluster is therefore equivalent to finding the maximum edge biclique in a bipartite graph, a problem known to be NP-complete [38]. More complex cases, where the actual numeric values in the matrix A are taken into account to compute the quality of a bicluster, have a complexity that is necessarily no lower than this one since, in general, they could also be used to solve the more restricted version of the problem, known to be NP-complete. Given this, the large majority of the algorithms use heuristic approaches to identify biclusters in many cases preceded by a normalization step that is applied to the data matrix in order to make more evident the patterns of interest. Some of them avoid heuristics but exhibit an exponential worst case runtime.

2.3 Dimensions of Analysis

Given the already extensive literature on biclustering algorithms, it is important to structure the analysis to be presented. To achieve this, we classified the surveyed biclustering algorithms along four dimensions:

- The type of biclusters they can find. The bicluster type is determined by the merit functions that define the type of homogeneity that they seek in each bicluster. This analysis is presented in Section 3.
- The way multiple biclusters are treated and the bicluster structure produced. Some algorithms find only one bicluster, others find nonoverlapping biclusters, others, more general, extract multiple, overlapping biclusters. This dimension is studied in Section 4.
- The specific algorithm used to identify each bicluster. Section 5 shows that some proposals use greedy methods, while others use more expensive global approaches or even exhaustive enumeration.
- The domain of application of each algorithm. Biclustering applications range from a number of microarray data analysis tasks to more exotic applications like recommendation systems, direct marketing and elections analysis. Applications of biclustering with special emphasis on biological data analysis are addressed in Section 7.

3 Bicluster Type

An interesting criterion to evaluate a biclustering algorithm concerns the identification of the type of biclusters the algorithm is able to find. We identified four major classes:

1. Biclusters with constant values.
2. Biclusters with constant values on rows or columns.
3. Biclusters with coherent values.
4. Biclusters with coherent evolutions.

The first three classes analyze directly the numeric values in the data matrix and try to find subsets of rows and subsets of columns with similar behaviors. These behaviors can be observed on the rows, on the columns, or in both dimensions of the data matrix, as in Figs. 1a, 1b, 1c, 1d, and 1e. The fourth class aims to find coherent behaviors regardless of the exact numeric values in the data matrix. As such, biclusters with coherent evolutions view the elements in the data matrix as symbols. These symbols can be purely nominal, as in Figs. 1f, 1g, and 1h; may correspond to a given order, as in Fig. 1i; or represent coherent positive and negative changes relatively to a normal value, as in Fig. 1j.

In the case of gene expression data, constant biclusters reveal subsets of genes with similar expression values within a subset of conditions. A bicluster with constant values in the rows identifies a subset of genes with similar expression values across a subset of conditions, allowing the expression levels to differ from gene to gene. Similarly, a bicluster with constant columns identifies a subset of conditions within which a subset of genes present similar expression values assuming that the expression values may differ from condition to condition. However, one can be interested in identifying more complex relations between the genes and the conditions by looking directly at the numeric values or regardless of them. As such, a bicluster with coherent values identifies a subset of genes and a subset of conditions with coherent values on both rows and columns. On the other hand, identifying a bicluster with coherent evolutions may be helpful if one is interested in finding a subset of genes that are upregulated or down-regulated across a subset of conditions without taking into account their actual expression values; or if one is interested in identifying a subset of conditions that have always the same or opposite effects on a subset of genes.

The simplest biclustering algorithms identify subsets of rows and subsets of columns with constant values. An example of a constant bicluster is presented in Fig. 1a. These algorithms are studied in Section 3.2.

Other biclustering approaches look for subsets of rows and subsets of columns with constant values on the rows or on the columns of the data matrix. The bicluster presented in Fig. 1b is an example of a bicluster with constant rows, while the bicluster depicted in Fig. 1c is an example of a bicluster with constant columns. Note that in the case of a bicluster with constant rows, every row in the bicluster can be obtained by adding a constant value to each of the other rows, or by multiplying them by a constant value. Similarly, each of the columns in a bicluster with constant columns can be obtained by adding a constant to each of the other columns, or by multiplying them by a constant value. Section 3.3 studies algorithms that discover biclusters with constant values on rows or columns.

More sophisticated biclustering approaches look for biclusters with coherent values on both rows and columns. The biclusters in Figs. 1d and 1e are examples of this type of bicluster. In both examples, each row can be obtained by adding a constant to each of the rows or by multiplying each of the rows by a constant value. Moreover, each column can also be obtained similarly by adding a constant to each of the columns or by multiplying each of the columns by a constant value. These algorithms are studied in Section 3.4.

The last type of biclustering approaches we analyzed addresses the problem of finding biclusters with coherent evolutions. The coevolution property can be observed on the entire bicluster, that is, on both rows and columns of the submatrix, as in Figs. 1f and 1j; on the rows of the bicluster, as in Fig. 1g; or on the columns of the bicluster, as in Figs. 1h and 1i. These approaches are addressed in Section 3.5.

According to the specific properties of each problem, one or more of these different types of biclusters is generally considered interesting. Moreover, a different type of merit function should be used to evaluate the quality of the biclusters identified. The choice of the merit function is strongly related with the characteristics of the biclusters each algorithm aims to find. The great majority of the algorithms we surveyed perform simultaneous clustering on both dimensions of the data matrix in order to find biclusters of the previous four classes. However, we also analyzed two-way clustering approaches that use one-way clustering to produce clusters on each of the two dimensions of the data matrix separately. These one-dimension results are then combined to produce subgroups of rows and columns whose properties allow us to consider the final result as biclustering. The type of biclusters produced by these algorithms depends, then, on the distance or similarity measure used by the one-way clustering algorithms. These algorithms will be considered in Sections 3.2, 3.3, 3.4, and 3.5, depending on the type of bicluster produced.

3.1 Notation

We will now introduce some notation used in the remaining of the section. Given the matrix $A = (X, Y)$, with set of rows X and set of columns Y, a bicluster is a submatrix (I, J), where I is a subset of the rows X, J is a subset of the columns Y and a_{ij} is the value in the matrix A corresponding to row the relation between i and column j. We denote by a_{iJ} the mean of the ith row in the bicluster, a_{Ij} the mean of the jth column in the bicluster and a_{IJ} the mean of all elements in the bicluster:

$$a_{iJ} = \frac{1}{|J|}\sum_{j \in J} a_{ij}, \tag{1}$$

$$a_{Ij} = \frac{1}{|I|}\sum_{i \in I} a_{ij}, \tag{2}$$

$$a_{IJ} = \frac{1}{|I||J|}\sum_{i \in I, j \in J} a_{ij}, \tag{3}$$

$$a_{IJ} = \frac{1}{|I|}\sum_{i \in I} a_{iJ} = \frac{1}{|J|}\sum_{j \in J} a_{Ij}. \tag{4}$$

3.2 Biclusters with Constant Values

When the goal of a biclustering algorithm is to find a constant bicluster or several constant biclusters, it is natural to consider ways of reordering the rows and columns of the matrix in order to group together similar rows and similar columns, and discover biclusters with similar values. Since this approach only produces good results when it is performed on nonnoisy data, which does not correspond to the great majority of available data, more sophisticated approaches can be used to pursue the goal of finding constant biclusters. The bicluster in Fig. 1a is an example of this type of bicluster.

A *perfect* constant bicluster is a submatrix (I, J), where all values are equal, for all $i \in I$ and $j \in J$:

$$a_{ij} = \mu. \tag{5}$$

Although these "ideal" biclusters can be found in some data matrices, in real data, constant biclusters are usually masked by noise. This means that the values a_{ij} found in what can be considered a constant bicluster are generally presented as $\eta_{ij} + \mu$, where η_{ij} is the noise associated with the real value μ of a_{ij}. The merit function used to compute and evaluate constant biclusters is, in general, the variance.

Hartigan [24] introduced a partition-based algorithm called direct clustering that became known as *Block Clustering*. This algorithm splits the original data matrix into a set of submatrices (biclusters) and uses the *variance* to evaluate the quality of each bicluster (I, J):

$$VAR(I, J) = \sum_{i \in I, j \in J} (a_{ij} - a_{IJ})^2. \tag{6}$$

According to this criterion, a perfect bicluster is a submatrix with variance equal to zero. Hence, every single-row, single-column matrix (I, J) in the data matrix, which corresponds to each element a_{ij}, is an ideal bicluster. As such, and in order to avoid the partitioning of the data matrix into biclusters with only one row and one column, Hartigan assumes that there are K biclusters within the data matrix: $(I, J)_k$ for $k \in 1, \ldots, K$. The algorithm stops when the data matrix is partitioned into K biclusters and the quality of the resulting biclustering is computed using the overall variance of the K biclusters:

$$VAR(I,J)_K = \sum_{k=1}^{K} \sum_{i \in I, j \in J} (a_{ij} - a_{IJ})^2. \tag{7}$$

Tibshirani et al. [46] added a backward pruning method to the block splitting algorithm introduced by Hartigan [24] and designed a permutation-based method to induce the optimal number of biclusters, K. The merit function used is also the variance and, consequently, they still find constant biclusters.

Cho et al. [11] also used the variance to find constant biclusters together with an alternative measure to enable the discovery of more complex biclusters (see Section 3.4).

3.3 Biclusters with Constant Values on Rows or Columns

There exists great practical interest in discovering biclusters that exhibit coherent variations on the rows or on the columns of the data matrix. As such, many biclustering algorithms aim at finding biclusters with constant rows or columns. The biclusters in Figs. 1b and 1c are examples of perfect biclusters with constant rows and columns, respectively.

A *perfect* bicluster with constant rows is a submatrix (I, J), where all the values within the bicluster can be obtained using one of the following expressions:

$$a_{ij} = \mu + \alpha_i, \tag{8}$$

$$a_{ij} = \mu \times \alpha_i, \tag{9}$$

where μ is the typical value within the bicluster and α_i is the adjustment for row $i \in I$. This adjustment can be obtained either in an additive (8) or multiplicative way (9).

Similarly, a *perfect* bicluster with constant columns is a submatrix (I, J), where all the values within the bicluster can be obtained using one of the following expressions:

$$a_{ij} = \mu + \beta_j, \tag{10}$$

$$a_{ij} = \mu \times \beta_j, \tag{11}$$

where μ is the typical value within the bicluster and β_j is the adjustment for column $j \in J$.

This class of biclusters cannot be found simply by computing the variance of the values within the bicluster, as we have seen in Section 3.2, or by computing similarities between the rows and columns of the data matrix. The straightforward approach to identify nonconstant biclusters is to normalize the rows or the columns of the data matrix using the row mean and the column mean, respectively. By doing this, the biclusters in Figs. 1b and 1c would both be transformed into the constant bicluster presented in Fig. 1a. This means that the row and column normalizations allow the identification of biclusters with constant values on the rows or on the columns of the data matrix, respectively, by transforming these biclusters into constant biclusters before the biclustering algorithm is applied.

A variant of this simple normalization approach was used by Getz et al. [21], who perform a relatively complex normalization step before their algorithm is applied. By doing this, Getz et al. not only manage to find biclusters with constant rows or constant columns defined, respectively, by (9) and (11), but also more complex biclusters with coherent values (see Section 3.4).

However, other biclustering algorithms that also aim at finding biclusters with constant rows or constant columns have different approaches that do not rely on a normalization step. Moreover, since perfect biclusters with constant rows or columns are hard to find in real data due to noise, the approaches described in the next paragraphs consider the possible existence of multiplicative noise, or that the values in the rows/columns belong to a certain interval, in order to allow the discovery of nonperfect biclusters.

Califano et al. [9] aim at finding δ-valid ks-patterns. They define a δ-valid ks-pattern as a subset of rows, I, with size k, and a subset of columns, J, with size s, such that the maximum and minimum value of each row in the chosen columns differ less than δ. This means that, for each row $i \in I$:

$$\max(a_{ij}) - \min(a_{ij}) < \delta, \forall j \in J. \tag{12}$$

The number of columns, s, is called the support of the ks-pattern. A δ-valid ks-pattern is defined as maximal if it cannot be extended into a δ-valid $k's$-pattern, with $k' > k$, by adding rows to its row set, and, similarly, it cannot be extended to a δ-valid ks'-pattern, $s' > s$, by adding columns to its column set. The goal is to discover maximal δ-valid gene expression patterns that are, in fact, biclusters with constant values on rows, by identifying sets of genes with coherent expression values across a subset of conditions. A statistically significance test is used to evaluate the quality of the patterns discovered.

Sheng et al. [42] tackled the biclustering problem in the Bayesian framework, by presenting a strategy based on a frequency model for the pattern of a bicluster and on Gibbs sampling for parameter estimation. Their approach not only unveils sets of rows and columns, but also represents the pattern of a bicluster as a probabilistic model described by the posterior frequency of every discretized value discovered under each column of the bicluster. They use multinomial distributions to model the data under every column in a bicluster, and assume that the multinomial distributions for different columns in a bicluster are mutually independent. Sheng et al. assumed a row-column orientation of the data matrix and ask that the values within the bicluster are consistent across the rows of the bicluster for each of the selected columns, although these values may differ for each column. By doing this, they manage to identify biclusters with constant values on the columns. However, the same approach can be followed using the column-row orientation of the data matrix leading to the identification of biclusters with constant rows.

Segal et al. [41] introduced a probabilistic model, which is based on the probabilistic relational models (PRMs). These models extend Bayesian networks to a relational setting with multiple independent objects such as genes and conditions. By using this approach, Segal et al. also manage to discover a set of biclusters with constant values on their columns. However, their approach is more general in the sense that it may also model a dependency of the expression levels on specific properties of genes and/or conditions. By inferring the appropriate dependencies between expression levels and gene (or condition) attributes, their approach is able to model biclusters that optimize an expression that is effectively more general than (11), or even than (13), in the next section. Moreover, Segal et al. considered the task of selecting among the many possible PRMs, where each of the possible models specified the set of parents for each attribute and the structure of the CPD-tree (Condition Probability Distribution-tree) [17]. In order to do that, they considered a *scoring function*, used to evaluate the quality of different candidate structures relatively to the data. The higher the scoring value the better the model.

3.4 Biclusters with Coherent Values

An overall improvement over the methods considered in the previous section, which presented biclusters with constant values either on rows or columns, is to consider biclusters with coherent values on both rows and columns. The biclusters in Figs. 1d and 1e are examples of this type of biclusters.

This class of biclusters cannot be found simply by considering that the values within the bicluster are given by additive or multiplicative models that consider an adjustment for either the rows or the columns, as it was described in (8), (9), (10), and (11). More sophisticated approaches perform an analysis of variance between groups and use a particular form of covariance between both rows and columns in the bicluster to evaluate the quality of the resulting bicluster or set of biclusters.

Following the same reasoning of Section 3.3, the biclustering algorithms that look for biclusters with coherent values can be viewed as based on an *additive model*. When an additive model is used within the biclustering framework, a *perfect* bicluster with coherent values, (I, J), is defined as a subset of rows and a subset of columns, whose values a_{ij} can be predicted using the following expression:

$$a_{ij} = \mu + \alpha_i + \beta_j, \tag{13}$$

where μ is the typical value within the bicluster, α_i is the adjustment for row $i \in I$, and β_j is the adjustment for column $j \in J$. The bicluster in Fig. 1d is an example of a bicluster with coherent values on both rows and columns, whose values can be described using an additive model. The biclusters in Figs. 1b and 1c can be considered special cases of this general additive model where the coherence of values can be observed on the rows and columns of the bicluster, respectively. This means that (8) and (10) are special cases of (13) when $\alpha_i = 0$ and $\beta_j = 0$, respectively.

Other biclustering approaches assume that biclusters with coherent values can be modeled using a *multiplicative model* to predict the values a_{ij} within the bicluster:

$$a_{ij} = \mu' \times \alpha'_i \times \beta'_j. \tag{14}$$

These approaches are effectively equivalent to the additive model in (13), when $\mu = \log(\mu')$, $\alpha_i = \log(\alpha'_i)$, and $\beta_j = \log(\beta'_j)$. In this model, each element a_{ij} is seen as the product between the typical value within the bicluster, μ', the adjustment for row i, α'_i, and the adjustment for column j, β'_j. The bicluster in Fig. 1e is an example of a bicluster with coherent values on both rows and columns, whose values can be described using a multiplicative model. Furthermore, the biclusters in Figs. 1b and 1c can also be considered special cases of this multiplicative model since (9) and (11) are special cases of (14) when $\alpha'_i = 0$ and $\beta'_j = 0$, respectively.

Several biclustering algorithms assume either additive or multiplicative models.

Cheng and Church [10] defined a bicluster as a subset of rows and a subset of columns with a high similarity score. The similarity score introduced and called *mean squared residue*, H, was used as a measure of the coherence of the rows and columns in the bicluster. Given the data matrix $A = (X, Y)$, a bicluster was defined as a uniform submatrix (I, J) having a low mean squared residue score. A submatrix (I, J) is considered a δ-bicluster if $H(I, J) < \delta$ for some $\delta \geq 0$. In particular, they aim at finding large and maximal biclusters with scores below a certain threshold δ. In a *perfect* δ-bicluster each row/column or both rows and columns exhibits an absolutely consistent bias ($\delta = 0$). The biclusters in Figs. 1b, 1c, and 1d are examples of this kind of perfect biclusters. This means that the values in each row or column can be generated by shifting the values of other rows or columns by a common offset. When this is the case, $\delta = 0$ and each element a_{ij} can be uniquely defined by its row mean, a_{iJ}, its column mean, a_{Ij}, and the bicluster mean, a_{IJ}. The difference $a_{Ij} - a_{IJ}$ is the relative bias held by the column j with respect to the other columns in the δ-bicluster. The same reasoning applied to the rows leads to the definition that, in a perfect δ-bicluster, the value of an element, a_{ij}, is given by a row-constant plus a column-constant plus a constant value:

$$a_{ij} = a_{iJ} + a_{Ij} - a_{IJ}. \tag{15}$$

Note that this corresponds to consider $\mu = a_{IJ}$, $\alpha_i = a_{iJ} - a_{IJ}$, and $\beta_j = a_{Ij} - a_{IJ}$ in (13).

Unfortunately, due to noise in data, δ-biclusters may not always be perfect. The concept of *residue* was thus introduced to quantify the difference between the actual value of an element a_{ij} and its expected value predicted from the corresponding row mean, column mean, and bicluster mean. The residue of an element a_{ij} in the bicluster (I, J), $r(a_{ij})$, and the value of a_{ij} in a nonperfect bicluster, are given by:

$$r(a_{ij}) = a_{ij} - a_{iJ} - a_{Ij} + a_{IJ}, \tag{16}$$

$$a_{ij} = r(a_{ij}) + a_{iJ} + a_{Ij} - a_{IJ}. \tag{17}$$

In order to assess the overall quality of a δ-bicluster, Cheng and Church defined the *mean squared residue*, H, of a bicluster (I, J) as the sum of the squared residues:

$$H(I, J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} r(a_{ij})^2. \tag{18}$$

Cho et al. [11] also used the mean squared residue score as the merit function to be minimized in their biclustering algorithm. In order to evaluate the homogeneity of a bicluster, they used two different measures of residue. The first was the variance used by Hartigan [24]:

$$r(a_{ij}) = a_{ij} - a_{IJ}. \tag{19}$$

This measure is specially useful to identify biclusters with constant values as we saw in Section (3.2). For this reason, Cheng and Church [10] and Yang et al. [51] had also used it as residue score to discard constant biclusters, which they considered trivial. However, to enable a more efficient discovery of biclusters with coherent values defined by the additive model in (13), Cho et al. [11] used the residue defined in (18) as a second measure. Since they compute all the biclusters simultaneously, the merit function minimized is the *total squared residue*, which is the sum of the squared residues of each bicluster (I, J):

$$\sum_{I,J} H(I,J). \tag{20}$$

Note that using expression (19) for the residue makes (6) and (7) equivalent to (18) and (20).

The mean squared residue score defined by Cheng and Church assumes there are no missing values in the data matrix. To guarantee this precondition, they replace the missing values by random numbers, during a preprocessing phase. Yang et al. [50], [51] generalized the definition of a δ-bicluster to cope with missing values and avoid the interference caused by the random fillings used by Cheng and Church. They defined a δ-bicluster as a subset of rows and a subset of columns exhibiting coherent values on the specified (nonmissing) values of the rows and columns considered.

The FLOC (FLexible Overlapped biClustering) algorithm [50], [51] introduced an *occupancy* threshold, ϑ, and defined a δ-bicluster of ϑ occupancy as a submatrix (I, J), where for each row $i \in I$, $\frac{|J_i'|}{|J|} > \vartheta$, and for each $j \in J$, $\frac{|I_j'|}{|I|} > \vartheta$. $|J_i'|$ and $|I_j'|$ are the number of specified elements on row i and column j, respectively. The *volume* of the δ-bicluster, υ_{IJ}, was defined as the number of specified values of a_{ij}. Note that the definition of Cheng and Church is a special case of this definition when $\vartheta = 1$. The term *base* was used to represent the bias of a row or column within a δ-bicluster (I, J). The base of a row i, the base of a column j, and the base of the δ-bicluster (I, J) are the mean of all specified values in row i, in column j and in the bicluster (I, J), respectively. This allows us to redefine a_{iJ}, a_{Ij}, a_{IJ}, $r(a_{ij})$, and $H(I, J)$, in (1), (2), (3), and (16), respectively, so that their calculations do not take into account missing values:

$$a_{iJ} = \frac{1}{|J_i'|}\sum_{j\in J_i'} a_{ij}, \tag{21}$$

$$a_{Ij} = \frac{1}{|I_j'|}\sum_{i\in I_j'} a_{ij}, \tag{22}$$

$$a_{IJ} = \frac{1}{\upsilon_{IJ}}\sum_{i\in I_i', j\in J_j'} a_{ij}, \tag{23}$$

$$r(a_{ij}) = \begin{cases} a_{ij} - a_{iJ} - a_{Ij} + a_{IJ}, & \text{if } a_{ij} \text{ is specified} \\ 0, & \text{otherwise.} \end{cases} \tag{24}$$

Yang et al. also considered that the coherence of a bicluster can be computed using the mean residue of all (specified) values. Moreover, they considered that this mean can be either arithmetic, geometric, or the mean of squares. The arithmetic mean, defined in (25), was used in [50]. The mean of squares, defined in (26), was used in [51] and redefines Cheng and Church's score in (18).

$$H(I,J) = \frac{1}{\upsilon_{IJ}}\sum_{i\in I_j', j\in J_i'} |r(a_{ij})|, \tag{25}$$

$$H(I,J) = \frac{1}{\upsilon_{IJ}}\sum_{i\in I_j', j\in J_i'} r(a_{ij})^2. \tag{26}$$

To access the quality of a biclustering with K biclusters, Yang et al. used the *average residue*:

$$\frac{1}{K}\sum_{k=1}^{K} H(I,J)_k. \tag{27}$$

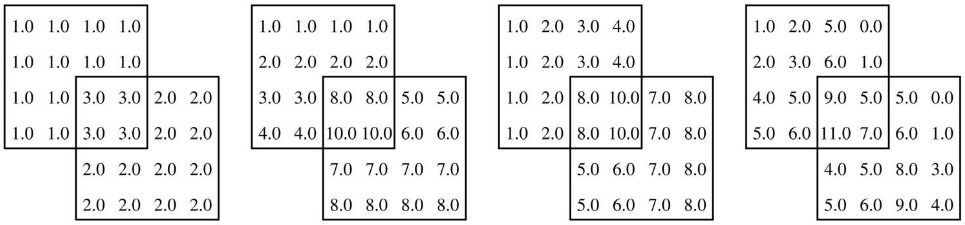
Wang et al. [48] also assume the additive model in (13) and seek to discover δ-pClusters. Given a submatrix (I, J) of A, they consider each 2×2 submatrix $M = (I_{i_1 i_2}, J_{j_1 j_2})$ defined by each pair of rows $i_1, i_2 \in I$ and each pair of columns $j_1, j_2 \in J$. The *pscore(M)* is computed as follows:

$$pscore(M) = |(a_{i_1 j_1} - a_{i_1 j_2}) - (a_{i_2 j_1} - a_{i_2 j_2})|. \tag{28}$$

They consider that the submatrix (I, J) is a δ-pCluster if for any 2×2 submatrix $M \subset (I, J)$, $pscore(M) < \delta$. They aim at finding δ-pClusters (pattern clusters), which are in fact biclusters with coherent values. An example of a perfect δ-pCluster modeled using an additive model is the one presented in Fig. 1d. However, if the values a_{ij} in the data matrix are transformed using $a_{ij} = log(a_{ij})$ this approach can also identify biclusters defined by the multiplicative model in (14). An example of a perfect δ-pCluster modeled using a multiplicative model is the one presented in Fig. 1e.

Kluger et al. [32] also addressed the problem of identifying biclusters with coherent values and looked for checkerboard structures in the data matrix by integrating biclustering of rows and columns with normalization of the data matrix. They assumed that after a particular normalization, which was designed to accentuate biclusters if they exist, the contribution of a bicluster is given by a multiplicative model as defined in (14). Moreover, they use gene expression data and see each value a_{ij} in the data matrix as the product of the background expression level of gene i, the tendency of gene i to be expressed in all conditions and the tendency of all genes to be expressed in condition j. In order to access the quality of a biclustering, Kluger et al. tested the results against a null hypothesis of no structure in the data matrix.

Tang et al. [45] introduced the Interrelated Two-Way Clustering (ITWC) algorithm that combines the results of one-way clustering on both dimensions of the data matrix in order to produce biclusters. After normalizing the rows of the data matrix, they compute the vector-angle cosine value between each row and a predefined stable pattern to test whether the row values vary much among the columns and remove the ones with little variation. After that, they use a correlation coefficient as similarity measure to measure the strength of the linear relationship between two rows or two columns, to perform two-way clustering. As this similarity measure depends only on the pattern and not on the absolute magnitude of the spatial vector, ITWC also identifies biclusters with coherent values defined by the multiplicative model in (14).



1.0	1.0	1.0	1.0		
1.0	1.0	1.0	1.0		
1.0	1.0	3.0	3.0	2.0	2.0
1.0	1.0	3.0	3.0	2.0	2.0
		2.0	2.0	2.0	2.0
		2.0	2.0	2.0	2.0

(a)

1.0	1.0	1.0	1.0		
2.0	2.0	2.0	2.0		
3.0	3.0	8.0	8.0	5.0	5.0
4.0	4.0	10.0	10.0	6.0	6.0
		7.0	7.0	7.0	7.0
		8.0	8.0	8.0	8.0

(b)

1.0	2.0	3.0	4.0		
1.0	2.0	3.0	4.0		
1.0	2.0	8.0	10.0	7.0	8.0
1.0	2.0	8.0	10.0	7.0	8.0
		5.0	6.0	7.0	8.0
		5.0	6.0	7.0	8.0

(c)

1.0	2.0	5.0	0.0		
2.0	3.0	6.0	1.0		
4.0	5.0	9.0	5.0	5.0	0.0
5.0	6.0	11.0	7.0	6.0	1.0
		4.0	5.0	8.0	3.0
		5.0	6.0	9.0	4.0

(d)

Fig. 2. Overlapping biclusters with general additive model. (a) Constant biclusters, (b) constant rows, (c) constant columns, and (d) coherent values.

Another approach that aims at finding biclusters with coherent values defined using the multiplicative model in (14) is the Double Conjugated Clustering (DCC) introduced by Busygin et al. [8]. DCC is a two-way clustering approach to biclustering that enables the use of any clustering algorithm. Busygin et al. use self-organizing maps (SOMs) and the angle metric (dot product) to compute the similarity between the rows and columns when performing one-way clustering.

Getz et al. [21] introduced the Coupled Two-Way Clustering (CTWC) algorithm. When CTWC is applied to gene expression data, it aims at finding subsets of genes and subsets of conditions, such that a single cellular process is the main contributor to the expression of the gene subset over the condition subset. This two-way clustering algorithm repeatedly performs one-way clustering on the rows and columns of the data matrix using stable clusters of rows as attributes for column clustering and vice versa. Any reasonable choice of clustering method and definition of stable cluster can be used within the framework of CTWC. Getz et al. used a hierarchical clustering algorithm, whose input is a similarity matrix between the rows computed according to the column set, and vice versa. The Euclidean distance is used as similarity measure after a preprocessing step where each column of the data matrix is divided by its mean and each row is normalized such that its mean vanishes and its norm is one. Due to the normalization step, CTWC also identifies biclusters with coherent values defined using the multiplicative model in (14).

The previous biclustering approaches are based either on additive or multiplicative models, which evaluate separately the contribution of each bicluster without taking into consideration the interactions between biclusters. In particular, they do not explicitly take into account that the value of a given element, a_{ij}, in the data matrix can be seen as a sum of the contributions of the different biclusters to whom the row i and the column j belong.

Lazzeroni and Owen [34] addressed this limitation by introducing the plaid model where the value of an element in the data matrix is viewed as a sum of terms called layers. In the plaid model the data matrix is described as a linear function of variables (layers) corresponding to its biclusters. The plaid model is defined as follows:

$$a_{ij} = \sum_{k=0}^{K} \theta_{ijk}\rho_{ik}\kappa_{jk}, \tag{29}$$

where K is the number of layers (biclusters) and the value of θ_{ijk} specifies the contribution of each bicluster k specified by ρ_{ik} and κ_{jk}. The terms ρ_{ik} and κ_{jk} are binary values that represent, respectively, the membership of row i and column j in bicluster k.

The plaid model described in (29) can be seen as a generalization of the additive model in (13). We will call this model the *general additive model*. For every element a_{ij}, it represents a sum of additive models representing the contribution of each bicluster $(I, J)_k$ to the value of a_{ij} in case $i \in I$ and $j \in J$.

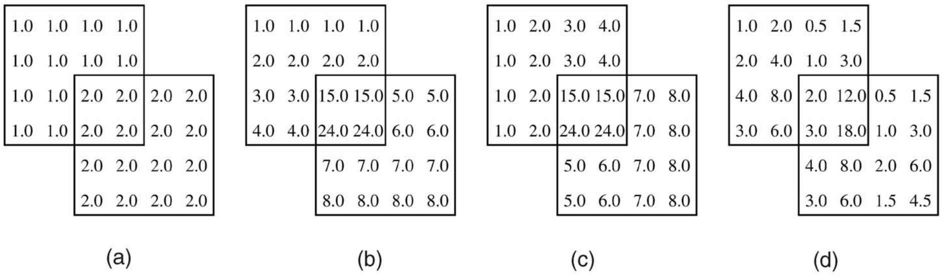
Lazzeroni and Owen [34] want to obtain a plaid model, which describes the interactions between the several biclusters on the data matrix and minimizes the following merit function:

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}\Big(a_{ij} - \theta_{ij0} - \sum_{k=1}^{K}\theta_{ijk}\rho_{ik}\kappa_{jk}\Big)^2, \tag{30}$$

where the term θ_{ij0} considers the possible existence of a single bicluster that covers the whole matrix and that explains away some variability that is not particular to any specific bicluster.

The notation θ_{ijk} makes this model powerful enough to identify different types of biclusters by using θ_{ijk} to represent either μ_k, $\mu_k + \alpha_{ik}$, $\mu_k + \beta_{jk}$, or $\mu_k + \alpha_{ik} + \beta_{jk}$. In its simplest form, that is when $\theta_{ijk} = \mu_k$, the plaid model identifies a set of K constant biclusters (see (5) in Section 3.2). When $\theta_{ijk} = \mu_k + \alpha_{ik}$, the plaid model identifies a set of biclusters with constant rows (see (8) in Section 3.3). Similarly, when $\theta_{ijk} = \mu_k + \beta_{jk}$, biclusters with constant columns are found (see (10) in Section 3.3). Finally, when $\theta_{ijk} = \mu_k + \alpha_{ik} + \beta_{jk}$, the plaid model identifies biclusters with coherent values by assuming the additive model in (13) for every bicluster k to whom row i and column j belong. Figs. 2a, 2b, 2c, and 2d show examples of different types of overlapping biclusters described by a general additive model where the values in the matrix are seen as a sum of the contributions of the different biclusters they belong to.

Segal et al. [40] also assumed the additive model in (13), the existence of a set of biclusters in the data matrix, and that the value of an element in the data matrix is a sum of terms called processes (see (29)). However, they assumed that the row contribution is the same for each bicluster and considered that each column belongs to every bicluster. This means that $\alpha_{ik} = 0$, for every row i in (13), $\theta_{ijk} = \mu_k + \beta_{jk}$ and $\kappa_{jk} = 1$, for all columns j and all biclusters k in (29). Furthermore, they introduced an extra degree of freedom by considering that each value in the data matrix is generated by a Gaussian distribution with a variance σ_k^2 that depends (only) on the bicluster index, k. As such, they want to minimize (31), where a_{ijk} is the sum of the



(a)

1.0	1.0	1.0	1.0		
1.0	1.0	1.0	1.0		
1.0	1.0	2.0	2.0	2.0	2.0
1.0	1.0	2.0	2.0	2.0	2.0
		2.0	2.0	2.0	2.0
		2.0	2.0	2.0	2.0

(b)

1.0	1.0	1.0	1.0		
2.0	2.0	2.0	2.0		
3.0	3.0	15.0	15.0	5.0	5.0
4.0	4.0	24.0	24.0	6.0	6.0
		7.0	7.0	7.0	7.0
		8.0	8.0	8.0	8.0

(c)

1.0	2.0	3.0	4.0		
1.0	2.0	3.0	4.0		
1.0	2.0	15.0	15.0	7.0	8.0
1.0	2.0	24.0	24.0	7.0	8.0
		5.0	6.0	7.0	8.0
		5.0	6.0	7.0	8.0

(d)

1.0	2.0	0.5	1.5		
2.0	4.0	1.0	3.0		
4.0	8.0	2.0	12.0	0.5	1.5
3.0	6.0	3.0	18.0	1.0	3.0
		4.0	8.0	2.0	6.0
		3.0	6.0	1.5	4.5

Fig. 3. Overlapping biclusters with general multiplicative model. (a) Constant biclusters, (b) constant rows, (c) constant columns, and (d) coherent values.

predicted value for the element a_{ij} in each bicluster k, which is computed using (29) with the above restrictions. This change allows one to consider as less important variations in the biclusters that are known to exhibit a higher degree of variability.

$$\sum_{k=1}^{K} \frac{(a_{ijk} - \theta_{ijk}\rho_{ik})^2}{2\sigma_k^2}, \tag{31}$$

$$a_{ij} = \sum_{k=1}^{K} a_{ijk}. \tag{32}$$

Following this reasoning, an obvious extension to (30) that has not been, to our knowledge, used by any published approach, is to assume that rows and columns, which represent, respectively, genes and conditions, in the case of gene expression data, can also exhibit different degrees of variability, that should be considered as having different weights. The expression to be minimized is therefore:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \frac{\left(a_{ij} - \theta_{ij0} - \sum_{k=1}^{K} \theta_{ijk}\rho_{ik}\kappa_{jk}\right)^2}{2(\sigma_{iJ}^2 + \sigma_{Ij}^2 + \sigma_{IJ}^2)}, \tag{33}$$

where σ_{iJ}^2, σ_{Ij}^2, and σ_{IJ}^2 are the row variance, the column variance, and the bicluster variance, respectively. This allows one to consider as less important variations in the rows, the columns and also the biclusters, that are know to exhibit a higher degree of variability.

Another possibility that has not been, to our knowledge, used by any published approach, is to consider that the value of a given element, a_{ij}, in the matrix is given by the product of the contributions of the different biclusters to which row i and column j belong, instead of a sum of contributions as considered by the plaid model. In this approach, which we will call the *general multiplicative model*, the value of each element a_{ij} in the matrix is given by the following expression:

$$a_{ij} = \prod_{k=0}^{K} \theta_{ijk}\rho_{ik}\kappa_{jk}. \tag{34}$$

Similar to the plaid model that sees a bicluster as a sum of layers (biclusters), (34) describes the value a_{ij} in the data matrix as a product of layers. The notation θ_{ijk} is now used to represent either μ_k, $\mu_k \times \alpha_{ik}$, $\mu_k \times \beta_{jk}$, or $\mu_k \times \alpha_{ik} \times \beta_{jk}$. Hence, in its general case, θ_{ijk} is now given by the multiplicative model in (14) instead of being defined by the additive model in (13). Figs. 3a, 3b, 3c, and 3d show examples of different types of overlapping biclusters described by a general multiplicative model where the values in the data matrix are seen as a product of the contributions of the different biclusters they belong to.

Conceptually, it is also possible to combine the general multiplicative model in (34) with θ_{ijk} given by the additive model in (13). Such a combination would consider an additive model for each bicluster, but a multiplicative model for the combination of the contributions given by the several biclusters. Similarly, it is also possible to combine the general additive model in (13) with θ_{ijk} given by the multiplicative model in (29). This means that each bicluster is generated using a multiplicative model, but the combination of biclusters is performed using an additive model. These combinations, however, are less likely to be useful than the general additive model ((13) and (29)) and the general multiplicative model ((14) and (34)).

3.5 Biclusters with Coherent Evolutions

In the previous section, we revised several biclustering algorithms that aimed at discovering biclusters with coherent values. Other biclustering algorithms address the problem of finding coherent evolutions across the rows and/or columns of the data matrix regardless of their exact values. The biclusters presented in Figs. 1h, 1i, and 1j are examples of biclusters with coherent evolutions.

Ben-Dor et al. [6] defined a bicluster as an order-preserving submatrix (OPSM). According to their definition, a bicluster is a group of rows whose values induce a linear order across a subset of the columns. Their work focuses on the relative order of the columns in the bicluster rather than on the uniformity of the actual values in the data matrix as the plaid model [34]. More specifically, they want to identify large OPSMs. A submatrix is order-preserving if there is a permutation of its columns under which the sequence of values in every row is strictly increasing. The bicluster presented in Fig. 1i is an example of an OPSM, where $a_{i4} \leq a_{i2} \leq a_{i3} \leq a_{i1}$, and represents a bicluster with coherent evolutions on its columns. Furthermore, Ben-Dor et al. defined a complete model as the pair (J, π), where J is a set of s columns and $\pi = (j_1, j_2, \ldots, j_s)$ is a linear ordering of the columns in J. They say that a row supports (J, π) if the s corresponding values, ordered according to the permutation π are monotonically increasing.

Although the straightforward approach to the OPSM problem would be to find a maximum support complete model, that is, a set of columns with a linear order supported by a maximum number of rows, Ben-Dor et al. aimed at finding a complete model with highest statistically significant support. In the case of expression data, such a submatrix is determined by a subset of genes and a subset of conditions, such that, within the set of conditions, the expression levels of all genes have the same linear ordering. As such, Ben-Dor et al. addressed the identification and statistical assessment of coexpressed patterns for large sets of genes, and considered that, in many cases, data contains more than one such pattern.

Following the same idea, Liu and Wang [35] defined a bicluster as an OP-Cluster (Order Preserving Cluster). Their goal is also to discover biclusters with coherent evolutions on the columns. Hence, the bicluster presented in Fig. 1i is an example of an OPSM and also of an OP-Cluster.

Murali and Kasif [37] aimed at finding conserved gene expression motifs (xMOTIFs). They defined an xMOTIF as a subset of genes (rows) that is simultaneously conserved across a subset of the conditions (columns). The expression level of a gene is conserved across a subset of conditions if the gene is in the same state in each of the conditions in this subset. They consider that a gene state is a range of expression values and assume that there are a fixed given number of states. These states can simply be upregulation and downregulation, when only two states are considered. An example of a perfect bicluster in this approach is the one presented in Fig. 1g, where S_i is the symbol representing the preserved state of the row (gene) i.

Murali and Kasif assumed that the data may contain several xMOTIFs (biclusters) and aimed at finding the largest xMOTIF: the bicluster that contains the maximum number of conserved rows. The merit function used to evaluated the quality of a given bicluster is thus the size of the subset of rows that belong to it (subset of rows that satisfy the conservation property in the subset of conditions). Together with the conservation property, an xMOTIF must also satisfy size and maximality properties: The number of columns must be in at least an α-fraction of all the columns in the data matrix, and for every row not belonging to the xMOTIF, the row must be conserved only in a β-fraction of the columns in it. As such, an xMOTIF is discarded if the size and maximality conditions are not satisfied.

Tanay et al. [44] defined a bicluster as a subset of genes (rows) that *jointly respond* across a subset of conditions (columns). A gene is considered to respond to a certain condition if its expression level changes significantly at that condition with respect to its normal level. Before SAMBA (Statistical-Algorithmic Method for Bicluster Analysis) is applied, the expression data matrix is modeled as a bipartite graph whose two parts correspond to conditions (columns) and genes (rows), respectively, with one edge for each significant expression change. Tanay et al. present two statistical models for the resulting graph. In the simpler model, they are looking for biclusters that manifest changes relatively to their normal level, without considering if the change was an increase or a decrease in the expression level. In the refined model, they look for consistent biclusters, in which every two conditions must always have the same effect or always have the opposite effect on each of the genes.

In the simpler model, it is assumed that all the genes in a given bicluster are regulated (up or down). This means that their values changed relatively to its normal level, in the subset of conditions that form the bicluster. The goal is then to find the largest biclusters with the regulation property. In order to do that, SAMBA does not try to find any kind of coherence on the values a_{ij}. It assumes that regardless of its true values, a_{ij} can be represented by two symbols: $S0$ or $S1$, where $S1$ means change and $S0$ means no-change. As such, the model graph has an edge between a gene and a column when there is a change in the expression level of that gene in that specific condition. No edge means no change. A large bicluster is, in this case, one with a maximum number of genes (rows) whose symbol standing for a_{ij} is expected to be $S1$. The bicluster presented in Fig. 1f is an example of the type of bicluster SAMBA produces, if we say that $S1$ is the symbol that represents a coherent change relative to normal expression.

In the refined model, the sign of the change is taken into account. This is achieved by assigning a signal $c_{ij} \in \{-1, 1\}$ to each edge of the graph, and then looking for a bicluster (I, J) and an assignment $\tau : I \cup J \rightarrow \{-1, 1\}$ such that $c_{ij} = \tau(i)\tau(j)$. This is equivalent to the selection of a set of columns (conditions) that have always the same or opposite effects on the set of rows. As such, the model in Fig. 1j represents the type of biclusters that SAMBA can now find.

However, the approach of Tanay et al. is not purely symbolic since the merit function used to evaluate the quality of a computed bicluster using SAMBA is the weight of the subgraph that models it. Its statistical significance is evaluated by computing the probability of finding at random a bicluster with at least its weight. Given that the weight of a subgraph is defined as the sum of the weights of gene-condition (row-column) pairs in it including edges and nonedges, weights are assigned to the edges of the bipartite subgraph so that heavy subgraphs correspond to statistical significant biclusters.

4 Bicluster Structure

Biclustering algorithms assume one of the following situations: Either there is only *one bicluster* in the matrix as in Fig. 4a, or the matrix contains K *biclusters*, where K is the number of biclusters we expect to identify and is usually defined *apriori*.

While most algorithms assume the existence of several biclusters [24], [10], [21], [9], [34], [41], [45], [50], [8], [44], [51], [32], [42], [40], [35], [11], others only aim at finding one bicluster. In fact, even though these algorithms can possibly find more than one bicluster, the target bicluster is usually the best according to some criterion [6], [37].

When the biclustering algorithm assumes the existence of several biclusters in the data matrix, the following bicluster structures can be obtained (see Figs. 4b, 4c, 4d, 4e, 4f, 4g, 4h, and 4i):

1. Exclusive row and column biclusters (rectangular diagonal blocks after row and column reorder).
2. Nonoverlapping biclusters with checkerboard structure.
3. Exclusive-rows biclusters.
4. Exclusive-columns biclusters.
5. Nonoverlapping biclusters with tree structure.

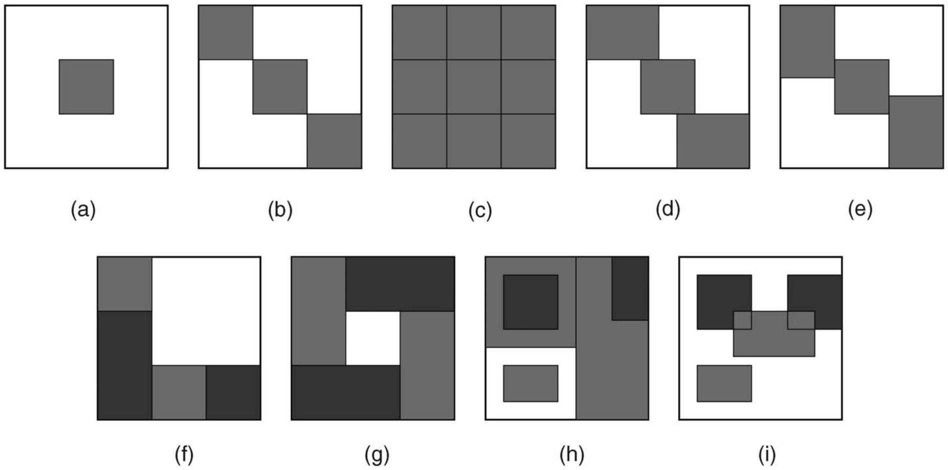


Fig. 4. Bicluster structure. (a) Single bicluster, (b) exclusive row and column biclusters, (c) checkerboard structure, (d) exclusive rows biclusters, (e) exclusive columns biclusters, (f) nonoverlapping biclusters with tree structure, (g) nonoverlapping nonexclusive biclusters, (h) overlapping biclusters with hierarchical structure, and (i) arbitrarily positioned overlapping biclusters.

6. Nonoverlapping nonexclusive biclusters.
7. Overlapping biclusters with hierarchical structure.
8. Arbitrarily positioned overlapping biclusters.

A natural starting point to achieve the goal of identifying several biclusters in a data matrix A is to form a color image of it with each element colored according to the value of a_{ij}. It is natural then to consider ways of reordering the rows and columns in order to group together similar rows and similar columns, thus forming an image with blocks of similar colors. These blocks are subsets of rows and subsets of columns with similar expression values, hence, biclusters. An ideal reordering of the matrix would produce an image with some number K of rectangular blocks on the diagonal (see Fig. 4b). Each block would be nearly uniformly colored, and the part of the image outside of these diagonal blocks would be of a neutral background color. This ideal corresponds to the existence of K mutually exclusive and exhaustive clusters of rows, and a corresponding K-way partitioning of the columns, that is, K exclusive row and column biclusters. In this structure, every row in the row-block k is expressed within, and only within, those columns in condition-block k. That is, every row and every column in the matrix belongs exclusively to one of the K biclusters (see Fig. 4b).

Although this can be the first approach to extract relevant knowledge from gene expression data, it has long been recognized that such an ideal reordering will seldom exist in real data [34]. Facing this fact, the next natural step is to consider that rows and columns may belong to more than one bicluster, and assume a checkerboard structure in the data matrix (see Fig. 4c). By doing this, we allow the existence of K nonoverlapping and nonexclusive biclusters where each row in the data matrix belongs to exactly K biclusters. The same applies to columns. Kluger et al. [32] and Cho et al. [11] assumed this structure. The Double Conjugated Clustering (DCC) approach introduced by Busygin et al. [8] can also identify this biclustering structure. However, DCC tends to produce the structure in Fig. 4b.

Other approaches assume that rows can only belong to one bicluster, while columns, which correspond to conditions in the case of gene expression data, can belong to several biclusters. This structure, which is presented in Fig. 4d, assumes exclusive-rows biclusters and was used by Sheng et al. [42] and Tang et al. [45]. However, these approaches can also produce exclusive-columns biclusters when the algorithm uses the opposite orientation of the data matrix. When this is the case, the columns can only belong to one bicluster while the rows can belong to one or more biclusters (see Fig. 4e).

The structures presented in Figs. 4b, 4c, 4d, and 4e assume that the biclusters are exhaustive, that is, that every row and every column belongs to at least one bicluster. However, we can consider nonexhaustive variations of these structures that make it possible that some rows and columns do not belong to any bicluster. A nonexhaustive version of the structure in Fig. 4b was assumed by Segal et al. [41]. Other exhaustive bicluster structures include the tree structure considered by Hartigan [24] and Tibshirani et al. [46] that is depicted in Fig. 4f, and the structure in Fig. 4g. A nonexhaustive variation of the structure in Fig. 4g was assumed by Wang et al. [48]. None of these structures allow overlapping, that is, none of them makes it possible that a particular pair (row, column) belongs to more than one bicluster.

The previous bicluster structures are restrictive in many ways. On one hand, some of them assume that, for visualization purposes, all the identified biclusters should be observed directly on the data matrix and displayed as a contiguous representation after performing a common reordering of their rows and columns. On the other hand, others assume that the biclusters are exhaustive, that is, that every row and every column in the data matrix belongs to at least one bicluster.

However, it is more likely that, in real data, some rows or columns do not belong to any bicluster at all and that the biclusters overlap in some places. It is, however, possible to enable these two properties without relaxing the visualization property if the hierarchical structure proposed by Hartigan [24] is assumed. This structure, depicted in Fig. 4h, requires that either the biclusters are disjoint or that one includes the other. Two specializations of this structure are the tree structure presented in Fig. 4f, where the biclusters form a tree, and the checkerboard structure depicted in Fig. 4c, where the biclusters and the row and column clusters are all trees.

A more general bicluster structure allows the existence of K possibly overlapping biclusters without taking into account their direct observation on the data matrix with a common reordering of its rows and columns. Furthermore, these nonexclusive biclusters can also be nonexhaustive, which means that some rows or columns may not belong to any bicluster. Several biclustering algorithms [10], [34], [21], [9], [45], [44], [6], [37], [40], [35] allow this more general structure, which is presented in Fig. 4i.

5 Algorithms

Biclustering algorithms may have two different objectives: to identify one or to identify a given number of biclusters. Some approaches attempt to identify *one bicluster at a time*. Cheng and Church [10] and Sheng et al. [42] identify a bicluster, mask it with random numbers, and repeat the procedure in order to eventually find other biclusters. Lazzeroni and Owen [34] attempt to discover one bicluster at a time in an iterative process where a plaid model is obtained. Ben-Dor et al. [6] also follow this strategy.

Other biclustering approaches discover *one set of biclusters at a time*. Hartigan [24] identifies two biclusters at the time by splicing each existing bicluster into two pieces at each iteration. CTWC [21] performs two-way clustering on the row and column dimensions of the data matrix separately. It uses a hierarchical clustering algorithm that generates stable clusters of rows and columns, at each iteration and, consequently, discovers a set of biclusters at a time. A similar procedure is followed by ITWC [45].

We also analyzed algorithms that perform *simultaneous bicluster identification*, which means that the biclusters are discovered all at the same time. FLOC [50], [51] follows this approach. It first generates a set of initial biclusters by adding each row/column to each one of them with independent probability and then iteratively improves the quality of the biclusters. Murali and Kasif [37] also identify several xMOTIFs (biclusters) simultaneously, although they only report the one that is considered the best according to the size and maximality criteria used. Tanay et al. [44], Liu and Yand [35], and Yang et al. [48] used exhaustive bicluster enumeration to perform simultaneous biclustering identification. Busygin et al. [8], Kluger et al. [32], Califano et al. [9], and Cho et al. [11] also discover all the biclusters at the same time.

Given the complexity of the problem, a number of different heuristic approaches has been used to address this problem. They can be divided into five classes:

1. Iterative row and column clustering combination.
2. Divide and conquer.
3. Greedy iterative search.
4. Exhaustive bicluster enumeration.
5. Distribution parameter identification.

The straightforward way to identify biclusters is to apply clustering algorithms to the rows and columns of the data matrix, separately, and then to combine the results using some sort of iterative procedure to combine the two cluster arrangements. Several algorithms use this *iterative row and column clustering combination* idea, and are described in Section 5.1. Other approaches, described in Section 5.2, use a *divide-and-conquer* approach: They break the problem into several subproblems that are similar to the original problem but smaller in size, solve the problems recursively, and then combine the solutions to create a solution to the original problem [13]. A large number of methods, studied in Section 5.3, perform some form of *greedy iterative search*. They always make a locally optimal choice in the hope that this choice will lead to a globally good solution [13]. Some authors proposed methods that perform *exhaustive bicluster enumeration*. A number of methods have been used to speed up exhaustive search, in some cases by assuming restrictions on the size of the biclusters that should be listed. These algorithms are revised in Section 5.4. The last type of approaches perform *distribution parameter identification* and are described in Section 5.5. They assume that the biclusters are generated using a given statistical model and try to identify the distribution parameters that fit, in the best way, the available data, by minimizing a certain criterion through an iterative approach.

5.1 Iterative Row and Column Clustering Combination

The conceptually simpler way to perform biclustering using existing techniques is to apply standard clustering methods on the column and row dimensions of the data matrix, and then combine the results to obtain biclusters. A number of authors have proposed methods based on this idea.

The Coupled Two-Way Clustering (CTWC) [21] seeks to identify couples of relatively small subsets of features (F_i) and objects (O_j), where both F_i and O_j can be either rows or columns, such that when only the features in F_i are used to cluster the corresponding objects O_j, stable and significant partitions emerge. It uses a heuristic to avoid brute-force enumeration of all possible combinations: Only subsets of rows or columns that are identified as stable clusters in previous clustering iterations are candidates for the next iteration. CTWC begins with only one pair of rows and columns, where each pair is the set containing all rows and the set that contains all columns, respectively. A hierarchical clustering algorithm is applied on each set generating stable clusters of rows and columns, and consequently a set of biclusters at a time. A tunable parameter T controls the resolution of the performed clustering. The clustering starts at $T = 0$ with a single cluster that contains all the rows and columns. As T increases, phase transitions take place, and this cluster breaks into several subclusters. Clusters keep breaking up as T is further increased, until at high enough values of T each row and column forms its own cluster. The control parameter T is used to provide a measure for the stability of any particular cluster by the range of values ΔT at which the cluster remains unchanged. A stable cluster is expected to survive throughout a large ΔT, one which constitutes a significant fraction of the range it takes the data to break into single point clusters. During its execution, CTWC dynamically maintains two lists of stable clusters (one for row clusters and one for column clusters) and a list of pairs of row and column subsets. At each iteration, one row subset and one column subset are coupled and clustered mutually as objects and features. Newly generated stable clusters are added to the row and column lists and a pointer that identifies the parent pair is recorded to indicate where this cluster came from. The iteration continues until no new clusters that satisfy some criteria such as stability and critical size are found.

The Interrelated Two-Way Clustering (ITWC) [45] is an iterative algorithm based on a combination of the results obtained by clustering performed on each of the two dimensions of the data matrix separately. Within each iteration of ITWC there are five main steps. In the first step, clustering is performed in the row dimension of the matrix. The goal is to cluster n_1 rows into K groups, denoted as $I_i, i = 1, \dots, K$, each of which is an exclusive subset of the set of all rows X. The clustering technique can be any method that receives the number of clusters as input. Tang et al. used K-means. In the second step, clustering is performed in the column dimension of the data matrix. Based on each group $I_i, i = 1, \dots, k$, the columns are independently clustered into two clusters, represented by $J_{i,a}$ and $J_{i,b}$. Assume, for simplicity, that the rows have been clustered into two groups, I_1 and I_2. The third step combines the clustering results from the previous steps by dividing the columns into four groups, C_i, $i = 1, \dots, 4$, that correspond to the possible combinations of the column clusters $J_{1,x}$ and $J_{2,x}$, $x = \{a, b\}$. The fourth step of ITWC aims at finding heterogeneous pairs (C_s, C_t), $s, t = 1, \dots, 4$. Heterogeneous pairs are groups of columns that do not share row attributes used for clustering. The result of this step is a set of highly disjoint biclusters, defined by the set of columns in C_s and C_t and the rows used to define the corresponding clusters. Finally, ITWC sorts the rows in descending order of the cosine distance between each row and a row representative of each bicluster (obtained by considering the value 1 in each entry for columns in C_s and C_t, respectively). The first one third of rows is kept. By doing this, they obtain a reduced row set I' for each heterogeneous group. In order to select the row set I' that should be chosen for the next iteration they use cross-validation. After this final step, the number of rows is reduced from n_1 to n_2 and the above steps can be repeated using the n_2 selected rows until the termination conditions are satisfied.

The Double Conjugated Clustering (DCC) [8] performs clustering in the rows and columns dimensions/spaces of the data matrix using self-organizing maps (SOM) and the angle-metric as similarity measure. The algorithm starts by assigning every node in one space (either a row or a column) to a particular node of the second space, which is called conjugated node. The clustering is then performed in two spaces. The first one is called the feature space, having n dimensions representing the rows of the data matrix. In the second space, called the sample space, the roles of the features and samples have been exchanged. This space has m dimensions, corresponding to the columns of the data matrix, and is used to perform clustering on the n features which are now the rows of the data matrix.

To convert a node of one space to the other space, DCC makes use of the angle between the node and each of the patterns. More precisely, the ith conjugate entry is the dot product between the node vector and the ith pattern vector of the projected space when both the vectors are normalized to unit length. Formally, they introduce the matrices X_1 and X_2, which corresponds to the original data matrix X after its columns and rows have been normalized to unit length. The synchronization between feature and sample spaces is forced by alternating clustering in both spaces. The projected clustering results of one space are used to correct the positions of the corresponding nodes of the other space. If the node update steps are small enough, both processes will converge to a state defined by a compromise between the two clusterings. Since the feature and sample spaces maximize sample and feature similarity, respectively, such a solution is desirable. DCC works iteratively by performing a clustering cycle, and then transforming each node to the conjugate space where the next training cycle takes place. This process is repeated until the number of moved samples/features falls below a certain threshold in both spaces. DCC returns two results: one in feature space and one in sample space, each being the conjugate of the other. Since every sample cluster in the feature space corresponds to a feature in the sample space, DCC derives a group of rows for every group of columns, hence, a set of biclusters.

5.2 Divide-and-Conquer

Divide-and-conquer algorithms have the significant advantage of being potentially very fast. However, these approaches have the very significant drawback of being likely to miss good biclusters that may be split before they can be identified.

Block clustering was the first divide-and-conquer approach to perform biclustering. Block clustering is a top down, row and column clustering of the data matrix. The basic algorithm for forward block splitting was due to Hartigan [24] who called it direct clustering (see Section 3.2). The block clustering algorithm begins with the entire data in one block (bicluster). At each iteration, it finds the row or column that produces the largest reduction in the total "within block" variance by splitting a given block into two pieces. In order to find the best split into two groups the rows and columns of the data matrix are sorted by row and column mean, respectively. The splitting continues until a given number K of blocks is obtained or the overall variance within the blocks reaches a certain threshold.

Since the estimation of the optimal number of splicings is difficult, Duffy and Quiroz [16] suggested the use of permutation tests to determine when a given block split is not significant. Following this direction, Tibshirani et al. [46] added a backward pruning method to the block splitting algorithm introduced by Hartigan [24] and designed a permutation-based method to induce the optimal number of biclusters, K, called Gap Statistics. In their approach, the splitting continues until a large number of blocks are obtained. Some blocks are then recombined until the optimal number of blocks is reached. This approach is similar to the one followed in decision tree algorithms, where the tree is grown until a given depth and is then pruned.

5.3 Greedy Iterative Search

Greedy iterative search methods are based on the idea of creating biclusters by adding or removing rows/columns from them, using a criterion that maximizes the *local* gain. As such, and although these approaches may make wrong decisions and loose good biclusters, they have the potential to be very fast.

Cheng and Church [10] were the first to apply biclustering to gene expression data. Given a data matrix A and a maximum acceptable mean squared residue score (see (18)),

$\delta > 0$, the goal is to find δ-biclusters, that is, subsets of rows and subsets of columns, (I, J), with a score no larger than δ (see Section 3.4). In order to achieve this goal, Cheng and Church proposed several greedy row/column removal/addition algorithms that are then combined in an overall approach that makes it possible to find a given number K of δ-biclusters.

The single node deletion method iteratively removes the row or column that gives the maximum decrease of H. The multiple node deletion method follows the same idea. However, in each iteration, it deletes all rows and columns with row/column residue superior to a given threshold. Finally, the node addition method adds rows and columns that do not increase the actual score of the bicluster. In order to find a given number, K, of biclusters, greedy node deletion is performed first and is then followed by greedy node addition. The algorithm discovers one bicluster at a time. At each of the K iterations, the algorithms start with an initial bicluster that contains all rows and columns. This means that the algorithm starts with the entire matrix A and stops when no action decreases H or when $H < \delta$. The discovered bicluster is then reported, and masked with random numbers, so that no recognizable structures remain. The process is repeated until K biclusters are found.

Although masking previously generated biclusters might suggest that it is not possible to find overlapping biclusters, this is in fact possible since the node addition step is performed using the original values in the data matrix and not the random ones introduced during the masking process. However, the discovery of highly overlapping biclusters is not likely, since elements of already identified biclusters have been masked by random noise.

The FLOC (FLexible Overlapped biClustering) algorithm [50], [51] addresses this limitation (see Section 3.4). It is based on the bicluster definition used by Cheng and Church, but performs simultaneous bicluster identification. It is also robust agains missing values, which are handled by taking into account the bicluster volume (number of nonmissing elements) when computing the score (see (26)). FLOC avoids the introduction of random interference and discovers K possibly overlapping biclusters simultaneously.

The algorithm has two phases. In the first phase, K initial biclusters are generated by adding each row/column to each one of them with independent probability p. The second phase is an iterative process that improves the quality of these biclusters. During each iteration, each row and each column is examined to determine the best action that can be taken toward reducing the average score residue (see (27)). An action is uniquely defined at any stage with respect to a row/column and a bicluster. It represents the change of membership of a row/column with respect to a specific bicluster: A row/column can be added to the bicluster if it is not yet included in it, or it can be removed if it already belongs to it. Since there are K biclusters, there are K potential actions for each row/column. Among these K actions, the one that has the maximum gain is identified and executed. The gain of an action is defined as a function of the relative reduction of the bicluster residue and the relative enlargement of the bicluster volume. At each iteration, the set of selected actions is performed according to a random weighted order that assigns higher probabilities of execution to the actions with higher gains. The optimization process stops when the potential actions do not improve the overall quality of the biclustering.

Cho et al. [11] introduced two k-means like biclustering algorithms that discover k row clusters and l column clusters simultaneously while monotonically decreasing the respective squared residues defined by Cheng and Church [10] (see (18)). They use the partitioning model proposed by Hartigan [24], but optimize global merit functions instead of local ones. They also find $k \times l$ biclusters simultaneously, as opposed to finding a single bicluster at a time.

The authors partition the matrix A, $n \times m$, into k row clusters and l column clusters defined, respectively, by the functions $\rho : \{1, \ldots, n\} \rightarrow \{1, \ldots, k\}$ and $\gamma : \{1, \ldots, m\} \rightarrow \{1, \ldots, l\}$, where $\rho(i) = r$ implies that row i is in the row cluster r and $\gamma(j) = c$ implies column c is in the column cluster c. It is also assumed that row cluster r, $1 \leq r \leq k$, has n_r rows, so that $n_1 + \ldots + n_r = n$. Similarly, cluster c, $1 \leq c \leq l$ has m_l columns, so that $m_1 + \ldots + m_l = m$. This means that every row/column belongs to exactly one row/column cluster. Then, they define a row cluster indicator matrix R, $n \times k$, and a column cluster indicator matrix C, $m \times l$. Each column r of R has n_r nonzeros, each of which equals $n_r^{-1/2}$. The nonzeros of C are defined similarly. As every nonzero value in R means that row i belongs to a particular row cluster, the first column in R has n_1 nonzeros, representing the rows that belong to the first row cluster, and so on. Matrix C has a similar structure. When R and C are constrained to be cluster indicator matrices the problem of obtaining a global minimum for (18) is NP-hard. Given this, Cho et al. resort to two iterative algorithms that monotonically decrease (18) and converge to a local minimum.

The first algorithm is a batch iterative algorithm since, at each iteration, the column clustering C is updated only after determining the nearest column cluster for *every* column of A (likewise for rows). They defined $A^C = RR^TAC$ and $A^R = R^TACC^T$, where the columns and rows of the matrices A^C and A^R play the roles of column cluster prototypes and row cluster prototypes, respectively. The algorithm begins with an initialization step. Each iteration involves finding the closest column (row) prototype, given by a column (row) of A^C (A^R), for each column (row) of A and setting its column (row) cluster accordingly. The algorithm iterates until the decrease in the merit function (18) is smaller than a tolerance τ.

The second algorithm is an incremental algorithm, whose idea is similar to the definition of action in FLOC [50], [51]. Cho et al. formulated incremental schemes for moving columns (rows) between column (row) clusters if such a move leads to a decrease in the merit function. Each call of the incremental procedures tries to perform such a move for each row and column of the data matrix. Since moving a row or column from its current cluster to an empty cluster would always lead to a decrease in the merit function (assuming no-degeneracy), such a move is always made guaranteeing that no cluster is empty. Although one move is made at a time, the authors suggest making a chain of moves for obtaining better local minima. Variants of the algorithm perform any move that leads to a decrease in the merit function instead of insisting on the best possible move.

Since the difficulty in minimizing (18) is introduced by the strong structural constraints on R and C, Cho et al. suggest relaxing those constraints to just seek column orthogonal matrices R and C in order to ease minimization. They then use a spectral approximation for initialization. When this relaxed solution is obtained, it is then necessary to obtain a biclustering. The authors suggest using k-means, among other techniques, to cluster the rows of R and C and obtain row/column clusters from the clustered R and C.

Ben-Dor et al. [6] addressed the identification of large order-preserving submatrices (OPSMs) with maximum statistical significance (see Section 3.5). In order to do that, they assume a probabilistic model of the data matrix where there is a bicluster (I, J) determined by a set of rows I, a set of columns J and a linear ordering of the columns in J. Within each row of (I, J) the order of the elements is consistent with the linear ordering of J. They define a complete model as the pair (J, π) where J is a set of s columns and $\pi = (j_1, j_2, \ldots, j_s)$ is a linear ordering of the columns in J. A row supports (J, π) if the s corresponding values, ordered according to the permutation π, are monotonically increasing.

Since an exhaustive algorithm that tries all possible complete models is not feasible, the idea is to grow partial models iteratively until they become complete models. A partial model of order (a, b) specifies, in order, the indices of the a "smallest" elements $< j_1, \ldots, j_a >$ and the indices of the b "largest" elements $< j_{s-(b-1)}, \ldots, j_s >$ of a complete model (J, π) and its size s. The OPSM algorithm focus on the columns at the extremes of the ordering when defining partial models, assuming that these columns are more useful in identifying the target rows, that is, the rows that support the assumed linear order. The algorithm starts by evaluating all $(1, 1)$ partial models and keeping the best l of them. It then expands them to $(2, 1)$ models and keeps the best l of them. After that, it expands them to $(2, 2)$ models, $(3, 2)$ models, and so on, until it gets l $([s/2], [s/2])$ models, which are complete models. It then outputs the best one.

Murali and Kasif [37] introduced an algorithm that aims at finding xMOTIFs. An xMOTIF is a bicluster with coherent evolutions on its rows (see Section 3.5). The data is first discretized into a set of symbols by using a list of statistically significant intervals, for each row. To determine an xMOTIF, it is necessary to compute the set of conserved rows, I, the states that these rows are in, and the set of columns, J, that match the xMOTIF. Given the set of conserved rows, I, the states of the conserved rows, and one column c that matches a given motif, it is easy to compute the remaining conditions in J simply by checking, for each column c', if the rows in I are in the same state in c and c'. Column c is called a *"seed"* from which the entire motif can be computed. The motifs are computed starting with a set of randomly chosen columns that act as seeds. For each column, an additional randomly chosen set D of columns is selected, called a *discriminating set*. The selected bicluster contains all the rows that have states equal in the seed column and in the columns contained in the discriminating set D. The motif is discarded if less than an $\alpha\text{-fraction}$ of the columns match it. After all the seeds have been used to produce xMOTIFs, the largest xMOTIF (one with the largest number of rows) is returned.

Califano et al. [9] introduced an algorithm that addresses the problem of finding $\delta\text{-valid}$ $ks\text{-patterns}$ (see Section 3.4). Their goal is to find groups of rows that exhibit coherent values in a subset of the columns, but do not have any coherence of values in any of the remaining columns. After preprocessing the data, they use a pattern discovery algorithm to discover sets of rows and columns candidates to be statistically significant biclusters (the other candidates are discarded). Finally, an optimal set of patterns is chosen among the statistically significant ones using a greedy set covering algorithm that adds rows and columns to the existing patterns so that they become maximal patterns (see Section 3.4).

The pattern discovery algorithm used considers that each column of the data matrix is a string and discovers patterns in these strings by allowing all possible string alignments. A density constraint is used to limit the impact of random matches occurring over large distances on the strings and the strings are prealigned before the algorithm is used. The algorithm starts with a single pattern with no rows, all the columns, and an offset of zero for each column. The values in each column are then sorted and all subsets of continuous values that are $\delta\text{-valid}$ (see Section 3.4) are selected. Nonmaximal subsets that are completely contained within another subset are removed. Each subset is considered a potential super-pattern of a maximal pattern. All possible maximal combinations of these super-patterns are then created iteratively. As a result, all patterns that exists in the data matrix are generated hierarchically by pattern combination.

5.4 Exhaustive Bicluster Enumeration

Exhaustive bicluster enumeration methods are based on the idea that the best biclusters can only be identified using an exhaustive enumeration of all possible biclusters existent in the matrix. These algorithms certainly find the best biclusters, if they exist, but have a very serious drawback. Due to their high complexity, they can only be executed by assuming restrictions on the size of the biclusters. As such, these methods perform exhaustive enumeration assuming some restrictions on the input instances and using efficient algorithmic techniques designed to make the enumeration feasible.

Tanay et al. [44] introduced SAMBA (Statistical-Algorithmic Method for Bicluster Analysis), a biclustering algorithm that performs simultaneous bicluster identification by using exhaustive enumeration. SAMBA avoids an exponential runtime by restricting the number of rows the biclusters may exhibit. They use the graph formalism described in Section 2.1, and define as their objective the identification of a maximum weight subgraph, assuming that the weight of a subgraph will correspond to its statistical significance. Discovering the most significant biclusters under this weighting schemes is equivalent to the selection of the heaviest subgraphs in the model bipartite graph. SAMBA assumes that row vertices have $d\text{-bounded}$ degree. This corresponds to a restriction on the size of the discovered biclusters since the number of rows cannot exceed this value. In the case of gene expression data this restriction is justified by the fact that genes that very frequently exhibit high expression levels are generally not interesting.

SAMBA can be executed using two statistical models of the resulting bipartite graph. When the simpler model is used, Tanay et al. show how to compute an upper-bound on the probability of an observed bicluster. When a refined model that takes into account the rows and columns variability by including the direction of the expression change (up or down regulation) is used, they show how to assign weights to the vertex pairs so that a maximum weight bicluster corresponds to a maximum likelihood bicluster. In a first phase, SAMBA normalizes the data, defining a gene as up-regulated or down-regulated if its standardized expression level (with mean 0 and variance 1), is, respectively, above 1 or below -1. In the second phase, it finds the K heaviest bicliques in the graph. This is done by looking at a precomputed table with the weights of the bicliques intersecting every given column (condition) or row (gene) and choosing the K best bicliques. In order to improve the performance, rows (genes) with degree exceeding d are ignored and the hashing for each row (gene) is performed only on subsets of its neighbors whose size is in a given range. In a postprocessing phase, SAMBA performs greedy addition or removal of vertices to perform a local improvement on the biclusters and filter the similar ones. Two biclusters are considered similar if their vertex sets (subset of rows and subset of columns) differ only slightly. The intersection between two biclusters is defined as the number of shared columns times the number of shared rows.

Wang et al. [48] also proposed an algorithm that performs exhaustive bicluster enumeration, subject to a restriction that they should possess a minimum number of rows and a minimum number of columns. To speed up the process and avoid the repetition of computations, they use a suffix tree to efficiently enumerate the possible combinations of row and column sets that represent valid biclusters.

The algorithm starts by deriving a set of candidate *Maximum Dimension Sets* (MDS) for each pair of rows and for each pair of columns. An (x, y) row-pair MDS is a set of columns that defines a maximum width bicluster that includes rows x and y. A similar definition holds for a column-pair MDS. The set of candidate MDSs is computed using an efficient method that generates all possible MDS for each row pair and for each column pair. This is done in linear time by ordering the columns in increasing order of the differences between row elements (in the case of the row-pair MDS), and performing a left to right scanning of these ordered array of columns. The set of candidate MDSs is then pruned using properties that relate row-pair MDSs with column-pair MDSs. The suffix tree [23] is built by assuming a given, arbitrary, lexicographic order on the columns. A node in the tree is associated with a set of columns, T, given by the path from the root, and a set of rows, O. A postorder traversal of this tree generates all possible biclusters using the following method: For each node, containing set of rows O and set of columns T, add the objects in O to nodes in the tree whose column set $T' \in T$ and $|T'| = |T| - 1$. Since the nodes that correspond to T' are necessarily higher in the suffix tree, the postorder traversal of this tree will generate all the existing biclusters in the matrix. However, the number of biclusters and, therefore, the execution time, can be exponential on the number of columns in the matrix.

Liu and Wang [35] also proposed an exhaustive bicluster enumeration algorithm. Since they are looking for order-preserving biclusters with a minimum number of rows and a minimum number of columns, the input data to their algorithm is a set of rows with symbols that represent the ordering of the values between these rows. A given row may then be represented by *adbc* and another one by *abdc*. Their goal of finding all the biclusters that, after column reordering, represent coherent evolutions of the symbols in the matrix is achieved by using a pattern discovery algorithm heavily inspired in sequential pattern mining algorithms [26].

The structure used to perform efficient enumeration of all common patterns in the rows uses an OPC-tree, which is a modified prefix tree, where a path from the root represents a sequence of symbols. In the starting tree, constructed using all symbol sequences present in the rows, leaves are labeled with the rows that correspond to the sequence of tree nodes that leads to that leaf. This tree is then iteratively modified by applying the following procedure to each node n of the tree, starting at the root: For each child n_c of node n, insert suffixes of subtrees of n_c in the child of n that has a label that matches the symbol that is in the root of the subtree. This procedure, complemented by appropriate pruning operations performed when there is not enough quorum to reach the target minimum bicluster dimension, generates all possible OP-clusters.

5.5 Distribution Parameter Identification

Distribution parameter identification approaches assume a given statistical model and try to identify the distribution parameters used to generate the data by iteratively minimizing a certain criterion.

Lazzeroni and Owen [34] want to obtain a plaid model that minimizes (30). Assuming that $K - 1$ layers (biclusters) have already been identified, they select the Kth bicluster that minimizes the sum of squared errors, Q. The residual from the first $K - 1$ biclusters, Z_{ij}, and Q are computed as follows:

$$Q = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}(Z_{ij} - \theta_{ijK}\rho_{iK}\kappa_{jK})^2, \qquad (35)$$

$$Z_{ij} = a_{ij} - \theta_{ij0} - \sum_{k=1}^{K-1}\theta_{ijk}\rho_{ik}\kappa_{jk}. \qquad (36)$$

Q is minimized through an iterative approach where the θ_{ijK} values, the ρ_{iK} values, and the κ_{jK} values are updated in turn. By doing this, one bicluster is discovered at a time. The iteration process is similar to the Expectation-Maximization (EM) algorithm. Lagrange Multipliers are used to estimate the parameters and improve the merit function along one direction at a time until a (possibly local) minimum is reached.

Let $\theta^{(s)}$, $\rho^{(s)}$, and $\kappa^{(s)}$ denote all the θ_{ijK}, the ρ_{iK}, and the κ_{jK} values at iteration s. The algorithm to find one layer works as follows: After selecting initial parameters $\rho^{(0)}$ and $\kappa^{(0)}$, S full update iterations are performed. At each of the $s = 1, \ldots, S$ iterations, the bicluster parameters $\theta^{(s)}$ are determined using $\rho^{(s-1)}$ and $\kappa^{(s-1)}$; the row membership $\rho^{(s)}$ are determined using $\theta^{(s)}$ and $\kappa^{(s-1)}$; and the column membership $\kappa^{(s)}$ are determined using $\theta^{(s)}$ and $\rho^{(s-1)}$. At intermediate stages, the values of θ_{ijK} describe a "fuzzy" membership function in which ρ_{iK} and κ_{jK} are not necessarily 0 or 1. To update the θ_{ijK} values given ρ_{iK} and κ_{jK}, the expression in (35) is minimized subject to the restrictions that every row and column has zero mean. The same reasoning is applied to estimate the remaining parameters. Given a set of K layers, the θ_{ijk} values can then be reestimated by cycling through $k = 1, \dots, K$ in turn.

Segal et al. [41] use PRMs (see Section 3.3) to model the joint distribution of the values of the gene and array membership functions ($\mu + \alpha_i + \beta_j$). The estimation of the parameters of the PRMs has two main components: learning the structure of the dependencies and estimation of the parameters of the distribution. Learning the structure of the dependencies between variables is equivalent to learning the structure of a Bayes network. To further restrict the different possible structures and the number of parameters, the authors assume that dependencies between variables can be represented by a CPD-tree [17]. The selection of the most likely structures is performed using Bayesian model selection techniques. The search is performed using an almost-greedy search method, where local minima are avoided by using a variant of simulated annealing. The search takes place by applying local transformations to the CPD-trees.

Once the structure is learned, parameter estimation is performed by obtaining the maximum likelihood estimator of each parameter, assuming multinomial distributions for the discrete parameters and Gaussian distributions for the continuous parameters. Since some variables are not observable (namely, the bicluster membership variables), the expectation-maximization method is applied. In this case, EM is applied by alternating the two following steps: 1) filling in the values obtained for each hidden variable given the current parameters of the distribution and 2) reestimating the parameters by assuming these guesses are correct using the standard maximum likelihood estimating procedure.

A more general model proposed by Segal et al. [40] allows overlapping biclusters and uses a similar approach to estimate the parameters of the distributions. However, in this case, the structure of the dependencies is fixed, since activity levels are assumed to depend on a specific and fixed way on the bicluster assignment variables. As such, the structure estimation phases are not required, and EM is used to estimate the parameters of the distribution. Unlike the plaid model [34], all the bicluster membership functions are reestimated at each step of the iteration, leading to a better fit of the experimental data.

Sheng et al. [42] introduced a biclustering approach based on Gibbs sampling. The row-column (gene-condition) orientation of the matrix is assumed. The algorithm could also be applied on the column-row (condition-gene) orientation. They use multinomial distributions to model the data for every column in a bicluster, and assume that the multinomial distributions for different columns in a bicluster are mutually independent. Gibbs sampling is used to estimate the parameters of the multinomial distributions used to model the data.

The algorithm to find one bicluster in the row-column orientation of the data matrix works as follows: The initialization step randomly assigns row labels and condition labels the value 1 or 0, where 1 means that the row/column belongs to the bicluster and 0 means it does not belong. In the second step of the algorithm, the goal is to fix the labels of the columns. In order to do that, for every row i, $i = 1, \dots, n$, the labels for all the other rows are fixed while the Bernoulli distribution for the given row i is computed. A label is then assigned to row i from the computed distribution. Similarly, step three sets the labels of the columns. The parameters for both row and column distributions are estimated using Gibbs sampling. These steps are iterated for a predefined number of iterations. In order to detect multiple biclusters, Sheng et al. mask the rows that belong to the previously found bicluster by setting the row labels of the found bicluster permanently to zero. Their approach discovers one bicluster at a time and it works by not considering as candidate rows for any future bicluster rows that have already been considered. This choice allows the unmasked dimension of the bicluster to be selected multiple times. The algorithm is iterated until no bicluster can be found for the unmasked part of the data matrix.

Kluger et al. [32] used a spectral approach to biclustering by assuming that, after normalization, the matrix contains a checkerboard structure. Supposing there are ways to normalize the original matrix A and the resulting matrix is A', the idea is to solve the eigenvalue problem $A'^T A' x = \lambda^2 x$ and examine the eigenvectors x. If the constants in an eigenvector can be sorted to produce a step-like structure, the column clusters can be identified accordingly. The row clusters are found similarly from y satisfying $A' A'^T y = \lambda^2 y$. More precisely, Kluger et al. show that the checkerboard pattern in a matrix A is reflected in the constant structures of the pair of eigenvectors x and y that solved the coupled eigenvalue problem $A'^T A' x = \lambda^2 x$ and $A'^T A' y = \lambda^2 y$, where x and y have a common eigenvalue. The algorithm depends critically on the normalization procedure used to transform the matrix.

Kluger et al. proposed three normalization methods. The first normalization method (independent rescaling of rows and columns) assumes the nonnormalized matrix is obtained by multiplying each row i by a scalar r_i and each column j by a scalar c_j, then $r_{i_1}/r_{i_2} =$ mean of row i_1/mean of row $i_2 = a_{i_1 J}/a_{i_2 J}$ (see (14)). Assuming that R is a diagonal matrix with entries r_i at the diagonal and C is a diagonal matrix defined similarly, then the eigen problem can be formulated by rescaling the data matrix: $\hat{A} \equiv R^{-1/2} A C^{-1/2}$. The second method (bistochastization) works by repeating the independent scaling of rows and columns until stability is reached. The final matrix has all rows sum to a constant and all columns sum to a different constant. The third method (log-interactions) assumes that if the original rows/columns differ by multiplicative constants, then after taking their logarithm, they differ by additive constants (see (13) and (14)). Moreover, each row and column is expected to have zero mean. This can be achieved by transforming each entry as follows: $a'_{ij} = a_{ij} - a_{Ij} - a_{iJ} + a_{IJ}$. Note that a'_{ij} is the residue of the each element a_{ij} of the data matrix A as it was defined in (16).

6 Overall Comparison of the Biclustering Algorithms

Table 2 presents a summary of the different algorithms in accordance with the different dimensions of analysis considered. The second column presents the type of biclusters (see Section 3). Column three lists the bicluster structure they can produce. The notation used is the one in Fig. 4 in Section 4. The last two columns summarize Section 5 and classify the different algorithms according to

TABLE 2
Overall Comparison of the Biclustering Algorithms

		Type	Structure	Discovery	Approach
Block Clustering	[24]	Constant	4(f)	One Set at a Time	Div-and-Conq
δ-biclusters	[10]	Coherent Values	4(i)	One at a Time	Greedy
FLOC	[50]	Coherent Values	4(i)	Simultaneous	Greedy
FLOC	[51]	Coherent Values	4(i)	Simultaneous	Greedy
pClusters	[48]	Coherent Values	4(g)	Simultaneous	Exh-Enum
Plaid Models	[34]	Coherent Values	4(i)	One at a Time	Dist-Ident
PRMs	[41]	Constant Columns	4(b)	Simultaneous	Dist-Ident
PRMs	[40]	Coherent Values	4(i)	Simultaneous	Dist-Ident
CTWC	[21]	Coherent Values	4(i)	One Set at a Time	Clust-Comb
ITWC	[45]	Coherent Values	4(d)/4(e)	One Set at a Time	Clust-Comb
DCC	[8]	Coherent Values	4(b)/4(c)	Simultaneous	Clust-Comb
δ-Patterns	[9]	Constant Rows	4(i)	Simultaneous	Greedy
Spectral	[32]	Coherent Values	4(c)	Simultaneous	Greedy
Gibbs	[42]	Constant Columns	4(d)/4(e)	One at a Time	Dist-Ident
OPSMs	[6]	Coherent Evolution	4(a)/4(i)	One at a Time	Greedy
SAMBA	[44]	Coherent Evolution	4(i)	Simultaneous	Exh-Enum
xMOTIFs	[37]	Coherent Evolution	4(a)/4(i)	Simultaneous	Greedy
OP-Clusters	[35]	Coherent Evolution	4(i)	Simultaneous	Exh-Enum
Co-Clusters	[11]	Constant/Coherent Values	4(i)	Simultaneous	Greedy

the way they discover the biclusters and the approach used. The notation used is the following: iterative row and column clustering combination (Clust-Comb), divide-and-conquer (Div-Conq), greedy iterative search (Greedy), exhaustive bicluster enumeration (Exh-Enum), and distribution parameter identification (Dist-Based).

An objective evaluation of the quality and efficiency of the different approaches is outside the scope of this survey since it requires extensive evaluations under controlled conditions. We plan, however, to conduct such evaluations in the future. Objective evaluations, in terms of the quality of the biclusters, are presented in the original references. This quality is assessed using one of the three following methods:

1. Value of the merit function. Authors that have used this approach evaluate the quality of the solution by analyzing the value of the merit function directly. This is a good indicator of the coherence observed in the resulting biclusters [24], [46], [10], [34], [50], [48], [51], [11]. In the cases where there is an explicit statistical model underlying the merit function, this approach can also be viewed directly as a statistical assessment of the quality of the solution [41], [40], [42], [37]. Two-way clustering approaches [21], [45], [8] do not evaluate the quality of the resulting biclustering directly. Instead, they evaluate the quality of the clustering performed on each of the two dimensions separately.
2. Statistical significance of the solution, measured against the null hypothesis. Authors that have used this approach assess the quality of the solutions by applying statistical significance tests. These tests derive the statistical significance of the solutions selected by the algorithms by computing the probability of random appearance of these biclusters in uncorrelated data, generated in accordance with a specific distribution [9], [44], [6], [32].
3. Comparison against known solutions. When tested in synthetic data, it is possible to compare the biclusters obtained with the biclusters that have been *planted* in the synthetic data. Several authors have used this approach. In real-world data, it is also sometimes possible (although considerably harder) to identify biclusters that correspond to known processes and to score the solutions by how effectively they approximate the known solutions. When the solutions are used to perform classification (gene or sample classification), this classification can also be compared with known results [46], [41], [9], [10], [21], [34], [45], [44], [8], [6], [50], [48], [40], [42], [32], [37], [51], [11].

An evaluation of the computational complexity and efficiency of the methods is very difficult to perform without resorting to extensive benchmarking. In fact, since the problem is NP-hard, all the methods resort to heuristics that are not easily amenable to direct complexity analysis. Iterative methods, used in the large majority of the approaches, are heavily dependent not only on the computational complexity per iteration that can be computed in a relatively straightforward way, but also on the number of iterations necessary to reach a solution. The number of iterations cannot, in general, be defined a priori, and represents the major factor constraining the efficiency of these methods.

Another important issue regards the adequacy of the models to real data. In this aspect, there is a clear move toward the use of more flexible merit functions and structures, reflecting the need to take into account the complex interactions between the processes that define the values in the matrix. In particular, general additive/multiplicative models and/or general probabilistic models that can model overlapping biclusters are required to model more precisely the data that is obtained from gene expression analysis. This move toward more complex

TABLE 3
Gene Expresion Data Sets Analyzed Using Biclustering Techniques and Its Applications

	Data Set		Applications	References
Yeast	*Yeast Cell Cycle 1*	[43]	Gene Functional Annotation Coregulation Identification	[44] [34]
	Yeast Cell Cycle 2	[12]	Coregulation Identification	[10] [50] [48] [51] [35] [11]
	Yeast Stress 1	[19]	Gene Functional Annotation Coregulation Identification	[41] [44] [40] [41] [44] [40]
	Yeast Stress 2	[18]	Gene Functional Annotation	[44]
	Yeast Compendium	[28]	Gene Functional Annotation	[41] [44]
	Yeast Galactose Utilization	[29]	Gene Functional Annotation	[44]
Human	*Response of Fibroblasts to Serum*	[30]	Sample Classification	[46]
	Lymphoma Microarray (B-Cells)	[2]	Coregulation Identification Sample Classification	[10] [44] [11] [37] [44] [32]
	Lymphoma Affimetrix	[31]	Sample Classification	[32]
	Leukemia Affimetrix 1 (ALL/AML)	[22]	Sample Classification	[21] [8] [37] [32]
	Leukemia Affimetrix 2	[4]	Sample Classification	[42]
	Colon Cancer	[3]	Sample Classification Coregulation Identification	[21] [37] [21]
	Multiple Sclerosis	[52]	Sample Classification	[45]
	Cancer Cell Lines Affimetrix	[49]	Sample Classification	[9]
	Breast Cancer 1	[25]	Coregulation Identification	[6]
	Breast Cancer 2	[33]	Sample Classification	[32]
	CNS Embryonal Tumor	[39]	Sample Classification	[32]

models also has an impact on the complexity of the algorithms required to find optimal solutions.

7 Biclustering Applications

Biclustering can be applied whenever the data to analyze has the form of a real-valued matrix A, where the set of values a_{ij} represent the relation between its rows i and its columns j, and the goal is to identify subsets of rows with certain coherence properties in a subsets of the columns.

Most biological applications of biclustering are performed using gene expression data obtained using microarray technologies that allow the measurement of the expression level of thousands of genes in target experimental conditions. Since this technology provides a snapshot of all the genes expressed in a cell at a given time, and gene expression is the fundamental link between genotype and phenotype, the analysis of gene expression data is bound to play a major role in our understanding of biological processes and systems including gene regulation, development, evolution, and disease mechanisms [5]. In this application domain, we can use biclusters to associate genes with specific clinical classes or for classifying genes and samples, among other potentially interesting applications.

The applications of biclustering to biological data analysis, including several examples using gene expression data and some examples using other biological data, are discussed in Section 7.1. Section 7.2 presents several nonbiological applications of biclustering. In fact, although the majority of the recent applications of biclustering are in biological data analysis, there are several interesting application of biclustering in other domains.

7.1 Biological Applications

Reported results of biclustering applied to gene expression data have used the data sets in Table 3. All these data sets contain expression data collected from either yeast or human cells using microarray technologies, which represents the expression level of a set of genes under specific conditions.

Several authors [10], [34], [41], [44], [50], [48], [51], [35], [40] used biclustering to analyze one or several of six expression matrices collected from yeast [43], [12], [19], [28], [18], [29] (see Table 3 for details). Other authors [46], [10], [21], [9], [45], [44], [6], [8], [37], [32], [42], [11] analyzed one or more of eleven different expression matrices with Human gene expression levels [49], [22], [3], [30], [2], [25], [31], [33], [52], [4], [39] (see Table 3 for details). Interestingly, almost all these data sets contain expression data related to the study of cancer. Some contain data from cancerous tissues at different stages of the disease; others analyze data from different individuals suffering from different types of cancer; and the remaining data sets contain data collected from several individuals with a particular cancer or healthy people.

These data sets have been used to test the applicability of biclustering approaches in three major tasks:

1. Identification of coregulated genes.
2. Gene functional annotation.
3. Sample classification.

Table 3 summarizes the biclustering applications to gene expression data analysis and relates the different tasks with the different datasets used.

A number of authors [10], [34], [21], [41], [44], [6], [50], [48], [40], [51], [35], [11] have studied the application of biclustering techniques to the problem of identification of coregulated genes. More specifically, the objective was to identify sets of genes that, under specific conditions, exhibited coherent activations that indicate coregulation. The results of this may be used to simply identify sets of coregulated genes or, more ambitiously, to identify specific regulation processes.

A less obvious application is to use the biclustering results directly to perform automatic gene functional annotation, as proposed by other authors [41], [44], [40]. The idea underlying this approach is to use biclusters where a large majority of genes belong to a specific class in the gene ontology to guess the class of nonannotated genes.

Another significant area of application is related with sample and/or tissue classification [46], [9], [21], [45], [44], [8], [32], [37], [42]. In leukemia diagnosis [21], [37], [42], for instance, the goal was to identify different responses to treatment, and the group of genes to be used as the most effective probe.

All the previous applications of biclustering analyzed data from gene expression matrices. However, biclustering can be interesting in the analysis of other biological data. Liu and Wang [35] applied biclustering to a drug activity data set. They wanted to find groups of chemical compounds with similar behaviors when subsets of compound descriptors were taken into account. Lazzeroni et al. [34] analyzed nutritional data to identify subsets of foods with similar properties on a subset of food attributes.

7.2 Other Applications

Apart from interesting biological applications, biclustering can also have successfully applications in relevant areas such as: information retrieval and text mining, collaborative filtering, recommendation systems, target marketing and market research, database research, and data mining.

Biclustering can be useful in collaborative filtering to identify subgroups of customers with similar preferences or behaviors toward a subset of products. The goal is to perform target marketing or use the information provided by the biclusters in recommendation systems. Recommendation systems and target marketing are important applications in the E-commerce area. Many authors applied biclustering to collaborative filtering using data where the rows represented customers and the columns movies [50], [51], [48], [47], [27]. The values a_{ij} in the data matrix show whether customer i watched movie j (binary values), or represent the rate customer i assigned to movie j (discrete values). Both Hoffman and Puzicha [27] and Ungar and Foster [47] used approaches similar to the one of Sheng et al. [42]. While Ungar and Foster used the Expectation-Maximization (EM) algorithm, Hoffman and Puzicha used Gibbs sampling.

Gaul and Schader [20] showed the relevance of biclustering in the market research and marketing by using data matrices containing data collected during marketing campaigns.

In information retrieval and text mining, biclustering can be used successfully to identify subgroups of documents with similar properties relatively to subgroups of attributes, such as words or images. This information can be very important in query and indexing in the domain of search engines. Several authors used data matrices where the rows represented words and the columns documents [14], [7], [15]. In the simpler problem formulations, the values a_{ij} show whether word i is present in document j (binary values). In more complex data, a nonzero element a_{ij} indicates the presence of word i in document j using a real value that represents its relevance relatively to that document and taking into account its presence in the collection of documents. In this application domain, this type of data is called *incidence matrix* and the term coclustering is generally used instead of biclustering. The approaches are similar to the ones analyzed in the previous sections. Dhillon [14], for instance, modeled the data matrix as a bipartite graph as Tanay et al. [44] and used a spectral approach similar to the one used by Kluger et al. [32].

Biclustering can also be used to perform dimensionality reduction in databases with tables with thousands of records (rows) with hundreds of fields (columns). This application of biclustering is what the database community calls automatic subspace clustering of high dimensional data, which is extremely relevant in data mining applications. This problem was addressed by Agrawal et al. [1].

More exotic applications of biclustering involved the analysis of data matrices with electoral data [24] and foreign exchange data [34]. Hartigan [24] used electoral data with the goal of identifying subgroups of rows (countries) with the same political ideas and electoral behaviors among a subset of the columns (issues). Lazzeroni et al. [34] analyzed foreign exchange data to identify subsets of currencies with similar behaviors in subsets of months.

8 Conclusions

We have presented a comprehensive survey of the models, methods, and applications developed in the field of biclustering algorithms. The list of applications presented is by no means exhaustive, and an all-inclusive list of potential applications would be prohibitively long. From the list of models and approaches analyzed in Sections 3, 4, 5, and 6, it is our opinion that the scientific community has already available a large plethora of models and algorithms to choose from. In particular, the general additive and multiplicative frameworks are rich enough to appropriately model very complex interactive processes.

The list of available algorithms is also very extense, and many combinations of ideas can be adapted to obtain new algorithms potentially more effective in particular applications. We believe that the systematic organization presented in this work can be used by the interested researcher as a good starting point to learn and apply some of the many techniques proposed in the last few years, and some of the older ones. The list of applications presented, long as it is already, represents, in our view, only a small fraction of the potential applications of this type of techniques. Many other applications in biological data analysis, gene network identification, data mining, and collaborative filtering remain to be explored.

Many interesting directions for future research have been uncovered by this review work. The tuning and validation of biclustering methods by comparison with known biological data is certainly one of the most important open issues. Another interesting area is the application of robust biclustering techniques to new and existing application domains. Many issues in biclustering algorithm design also remain open and should be addressed by the scientific community. From these open issues, we select the analysis of the statistical significance of biclusters as one of the most important ones, since the extraction of a large number of biclusters in real data may lead to results that are difficult to interpret.

Acknowledgments

The authors would like to thank Ana Teresa Freitas for many interesting and fruitful discussions on the subject of biclustering algorithms and applications.

References

[1] R. Agrawal, J. Gehrke, D. Gunopulus, and P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications," *Proc. ACM/SIGMOD Int'l Conf. Management of Data,* pp. 94-105, 1998.

[2] A.A. Alizadeh, M.B. Eisen, R.E. Davis, C. Ma, I.S. Lossos, A. Rosenwald, J.C. Boldrick, H. Sabet, T. Tran, X. Yu, J.I. Powell, L. Yang, G.E. Marti, T. Moore, J. Hudson, L. Lu, D.B. Lewis, R. Tibshirani, G. Sherlock, W.C. Chan, T.C. Greiner, D.D. Weisenburger, J.O. Armitage, R. Warnke, R. Levy, W. Wilson, M.R. Grever, J.C. Byrd, D. Botstein, P.O. Brown, and L.M. Staudt, "Distinct Types of Diffuse Large B-Cell Lymphoma Identified by Gene Expression Profiling," *Nature,* vol. 403, pp. 503-511, 2000.

[3] U. Alon, N. Barkai, D.A. Notterman, K. Gish, S. Ybarra, D. Mack, and A.J. Levine, "Broad Patterns of Gene Expression Revealed by Clustering of Tumor and Normal Colon Tissues Probed by Oligonucleotide Arrays," *Natural Academy of Sciences,* vol. 96, no. 12, pp. 6745-6750, 1999.

[4] S.A. Armstrong, J.E. Staunton, L.B. Silverman, R. Pieters, M.L. den Boer, M.D. Minden, S.E. Sallan, E.S. Lander, T.R. Golub, and S.J. Korsmeyer, "Mll Translocations Specify a Distinct Gene Expression Profile that Distinguishes a Unique Leukemia," *Nature Genetics,* vol. 30, pp. 41-47, 2002.

[5] P. Baldi and G.W. Hatfield, *DNA Microarrays and Gene Expression. From Experiments to Data Analysis and Modelling.* Cambridge Univ. Press, 2002.

[6] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini, "Discovering Local Structure in Gene Expression Data: The Order-Preserving Submatrix Problem," *Proc. Sixth Int'l Conf. Computational Biology (RECOMB '02),* pp. 49-57, 2002.

[7] P. Berkhin and J.D. Becher, "Learning Simple Relations: Theory and Applications," *Proc. Second SIAM Int'l Conf. Data Mining,* pp. 420-436, 2002.

[8] S. Busygin, G. Jacobsen, and E. Kramer, "Double Conjugated Clustering Applied to Leukemia Microarray Data," *Proc. Second SIAM Int'l Conf. Data Mining, Workshop Clustering High Dimensional Data,* 2002.

[9] A. Califano, G. Stolovitzky, and Y. Tu, "Analysis of Gene Expression Microarays for Phenotype Classification," *Proc. Int'l Conf. Computacional Molecular Biology,* pp. 75-85, 2000.

[10] Y. Cheng and G.M. Church, "Biclustering of Expression Data," *Proc. Eighth Int'l Conf. Intelligent Systems for Molecular Biology (ISMB '00),* pp. 93-103, 2000.

[11] H. Cho, I.S. Dhillon, Y. Guan, and S. Sra, "Minimum Sum-Squared Residue Cococlustering of Gene Expression Data," *Proc. Fourth SIAM Int'l Conf. Data Mining,* 2004.

[12] 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," *Molecular Cell,* vol. 2, pp. 65-73, 1998.

[13] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms,* The MIT Electrical Eng, and Computer Science Series, The MIT Press, second ed., 2001.

[14] I.S. Dhillon, "Co-Clustering Documents and Words Using Bipartite Spectral Graph Partitioning," *Proc. Seventh ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '01),* pp. 269-274, 2001.

[15] I.S. Dhillon, S. Mallela, and D.S. Modha, "Information-Theoretical Coclustering," *Proc. Ninth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '03),* pp. 89-98, 2003.

[16] D. Duffy and A. Quiroz, "A Permutation Based Algorithm for Block Clustering," *J. Classification,* vol. 8, pp. 65-91, 1991.

[17] N. Friedman and M. Goldszmidt, "Learning Bayesian Networks with Local Structure," *Learning in Graphical Models,* Kluwer, pp. 421-460, 1998.

[18] A.P. Gasch, M. Huang, S. Metzner, D. Botstein, S.J. Elledge, and P.O. Brown, "Genomic Expression Responses to DNA-Damaging Agents and the Regulatory Role of the Yeast ATR Homolog mec1p," *Molecular Biology of the Cell,* vol. 12, pp. 2987-3003, 2001.

[19] A.P. Gasch, P.T. Spellman, C.M. Kao, O. Carmel-Harel, M.B. Eisen, G. Storz, D. Botstein, and P.O. Brown, "Genomic Expression Programs in the Response of Yeast Cells to Environmental Changes," *Molecular Biology of the Cell,* vol. 11, pp. 4241-4257, 2000.

[20] W. Gaul and M. Schader, "A New Algorithm for Two-Mode Clustering," *Data Analysis and Information Systems,* H. Hermann and W. Polasek, eds., Springer, pp. 15-23, 1996.

[21] G. Getz, E. Levine, and E. Domany, "Coupled Two-Way Clustering Analysis of Gene Microarray Data," *Proc. Natural Academy of Sciences US,* pp. 12079-12084, 2000.

[22] T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, and E.S. Lander, "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring," *Science,* vol. 286, pp. 531-537, 1999.

[23] D. Gusfield, *Algorithms on Strings, Trees, and Sequences.* Computer Science and Computational Biology Series, Cambridge Univ. Press, 1997.

[24] J.A. Hartigan, "Direct Clustering of a Data Matrix," *J. Am. Statistical Assoc. (JASA),* vol. 67, no. 337, pp. 123-129, 1972.

[25] I. Hedenfalk, D. Duggan, Y. Chen, M. Radmacher, M. Bittner, R. Simon, P. Meltzer, B. Gusterson, M. Esteller, M. Raffeld, Z. Yakhini, A. Ben-Dor, E. Dougherty, J. Kononen, L. Bubendorf, W. Fehrle, S. Pittaluga, S. Gruvberger, N. Loman, O. Johannsson, H. Olsson, B. Wilfond, G. Sauter, O.P. Kallioniemi, A. Borg, and J. Trent, "Gene-Expression Profiles in Hereditary Breast Cancer," *New England J. Medicine,* vol. 344, no. 8, pp. 539-548, 2000.

[26] J. Hipp, U. Güntzer, and G. Nakhaeizadeh, "Algorithms for Association Rule Mining—A General Survey and Comparison," *SIGKDD Explorations,* vol. 2, no. 1, pp. 58-64, July 2000.

[27] T. Hofmann and J. Puzicha, "Latent Class Models for Collaborative Filtering," *Proc. Int'l Joint Conf. Artificial Intelligence,* pp. 668-693, 1999.

[28] T.R. Hughes, M.J. Marton, A.R. Jones, C.J. Roberts, R. Stoughton, C.D. Armour, H.A. Bennett, E. Coffey, H. Dai, Y.D. He, M.J. Kidd, A.M. King, M.R. Meyer, D. Slade, P.Y. Lum, S.B. Stepaniants, D.D. Shoemaker, D. Gachotte, K. Chakraburtty, J. Simon, M. Bard, and S.H. Friend, "Functional Discovery via a Compendium of Expression Profiles," *Cell,* vol. 102, pp. 109-126, 2000.

[29] T. Ideker, V. Thorsson, J.A. Ranish, R. Christmas, J. Buhler, J.K. Eng, R. Bumgarner, D.R. Goodlett, ? Aebersold, and L. Hood, "Integrated Genomic and Proteomic Analyses of a Systematically Perturbed Metabolic Network," *Science,* vol. 292, pp. 929-934, 2001.

[30] V.R. Iyer, M.B. Eisen, D.T. Ross, G. Schuler, T. Moore, J.C.F. Lee, J.M. Trent, L.M. Staudt, J. Hudson Jr., M.S. Boguski, D. Lashkari, D. Shalon, D. Botstein, and P.O. Brown, "The Transcriptional Program in the Response of Human Fibroblasts to Serum," *Science,* vol. 283, pp. 83-87, 1999.

[31] U. Klein, Y. Tu, G.A. Stolovitzky, M. Mattioli, G. Cattoretti, H. Husson, A. Freedman, G. Inghirami, L. Cro, L. Baldini, A. Neri, A. Califano, and R. Dalla-Favera, "Gene Expression Profiling of B-Cell Chronic Lymphocytic Leukemia Reveals a Homogeneous Phenotype Related to Memory B Cells," *J. Experimental Medicine,* vol. 194, pp. 1625-1638, 2001.

[32] Y. Klugar, R. Basri, J.T. Chang, and M. Gerstein, "Spectral Biclustering of Microarray Data: Coclustering Genes and Conditions," *Genome Research,* vol. 13, pp. 703-716, 2003.

[33] U. Kluger, B. Kacinski, Y. Kluger, O. Mironenko, M. Gilmore-Hebert, J. Chang, A. Perkins, and E. Sapi, "Microarray Analysis of Invasive and Metatastic Phenotypes in a Breast Cancer Model," *Poster Presented at the Gordon Conf. Cancer,* 2001.

[34] L. Lazzeroni and A. Owen, "Plaid Models for Gene Expression Data," technical report, Stanford Univ., 2000.

[35] J. Liu and W. Wang, "OP-Cluster: Clustering by Tendency in High Dimensional Space," *Proc. Third IEEE Int'l Conf. Data Mining,* pp. 187-194, 2003.

[36] B. Mirkin, "Nonconvex Optimization and its Applications," *Math. Classification and Clustering,* Kluwer Academic Publishers, 1996.

[37] T.M. Murali and S. Kasif, "Extracting Conserved Gene Expression Motifs from Gene Expression Data," *Proc. Pacific Symp. Biocomputing,* vol. 8, pp. 77-88, 2003.

[38] R. Peeters, "The Maximum Edge Biclique Problem is NP-Complete," *Discrete Applied Math.,* vol. 131, no. 3, pp. 651-654, 2003.

[39] S.L. Pomeroy, P. Tamayo, M. Gaasenbeek, L.M. Sturla, M. Angelo, M.E. McLaughlin, J.Y. Kim, L.C. Goumnerova, P.M. Black, C. Lau, J.C. Allen, D. Zagzag, J.M. Olson, T. Curran, C. Wetmore, J.A. Biegel, T. Poggio, S. Mukherjee, R. Rifkin, A. Califano, G. Stolovitzky, D.N. Louis, J.P. Mesirov, E.S. Lander, and T.R. Golub, "Prediction of Central Nervous System Embryonal Tumour Outcome Based on Gene Expression," *Nature,* vol. 415, no. 6870, pp. 436-442, 2002.

[40] E. Segal, A. Battle, and D. Koller, "Decomposing Gene Expression into Cellular Processes," *Proc. Pacific Symp. Biocomputing,* vol. 8, pp. 89-100, 2003.

[41] E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller, "Rich Probabilistic Models for Gene Expression," *Bioinformatics,* vol. 17, pp. S243-S252, 2001.

[42] Q. Sheng, Y. Moreau, and B. De Moor, "Biclustering Microarray Data by Gibbs Sampling," *Bioinformatics,* vol. 19, pp. ii196-ii205, 2003.

[43] P.T. Spellman, G. Sherlock, M.Q. Zhang, V.R. Iyer, K. Anders, M.B. Eisen, P.O. Brown, D. Botstein, and B. Futcher, "Comprehensive Identification of Cell Cycle-Regulated Genes of the Yeast Saccharomyces Cerevisiae by Microarray Hybridization," *Molecular Biology of the Cell,* vol. 9, pp. 3273-3297, 1998.

[44] A. Tanay, R. Sharan, and R. Shamir, "Discovering Statistically Significant Biclusters in Gene Expression Data," *Bioinformatics,* vol. 18, pp. S136-S144, 2002.

[45] C. Tang, L. Zhang, I. Zhang, and M. Ramanathan, "Interrelated Two-Way Clustering: An Unsupervised Approach for Gene Expression Data Analysis," *Proc. Second IEEE Int'l Symp. Bioinformatics and Bioeng.,* pp. 41-48, 2001.

[46] R. Tibshirani, T. Hastie, M. Eisen, D. Ross, D. Botstein, and P. Brown, "Clustering Methods for the Analysis of DNA Microarray Data," technical report, Dept. of Health Research and Policy, Dept. of Genetics, and Dept. of Biochemestry, Stanford Univ., 1999.

[47] L. Ungar and D.P. Foster, "A Formal Statistical Approach to Collaborative Filtering," *Proc. Conf. Automated Learning and Discovery (CONALD '98),* 1998.

[48] H. Wang, W. Wang, J. Yang, and P.S. Yu, "Clustering by Pattern Similarity in Large Data Sets," *Proc. 2002 ACM SIGMOD Int'l Conf. Management of Data,* pp. 394-405, 2002.

[49] J.N. Weinstein, T.G. Myers, P.M. O'Connor, S.H. Friend, A.J. Fornace Jr., K.W. Kohn, T. Fojo, S.E. Bates, L.V. Rubinstein, N.L. Anderson, J.K. Buolamwini, W.W. Van Osdol, A.P. Monks, D.A. Scudiero, E.A. Sausville, D.W. Zaharevitz, B. Bunow, V.N. Viswanadhan, G.S. Johnson, R.E. Wittes, and K.D. Paull, "An Information-Intensive Approach to the Molecular Pharmacology of Cancer," *Science,* vol. 275, pp. 343-349, 1997.

[50] J. Yang, W. Wang, H. Wang, and P. Yu, "δ-Clusters: Capturing Subspace Correlation in a Large Data Set," *Proc. 18th IEEE Int'l Conf. Data Eng.,* pp. 517-528, 2002.

[51] J. Yang, W. Wang, H. Wang, and P. Yu, "Enhanced Biclustering on Expression Data," *Proc. Third IEEE Conf. Bioinformatics and Bioeng.,* pp. 321-327, 2003.

[52] V. Yong, S. Chabot, Q. Stuve, and G. Williams, "Interferon Beta in the Treatment of Multiple Sclerosis: Mechanisms of Action," *Neurology,* vol. 51, pp. 682-689, 1998.

Sara C. Madeira received the BSc degree in computer science from the University of Beira Interior in 2000, and the Msc degree in computer engineering and computer science from Lisbon Technical University in 2002. She is currently a PhD student at IST, Lisbon Technical University, and an assistant lecturer in the Computer Science Department at the University of Beira Interior. She is also an invited researcher at INESC-ID. Her research interests include bioinformatics, data mining, and machine learning.

Arlindo L. Oliveira received the BSc (Eng) and the MSc degrees in electrical and computer engineering from Lisbon Technical University, and the PhD degree in electrical engineering and computer science from the University of California, Berkeley, in 1994. He is currently an associate professor at IST, Lisbon Technical University. He is also a senior researcher at INESC-ID. His research interests include bioinformatics, string processing, combinatorial optimization, machine learning, logic synthesis, and automata theory. He is a member of the IEEE and the IEEE Computer Society.