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 of the application of standard clustering methods to genes are limited. These limited results are 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 gene expression matrix has been proposed to date. This simultaneous clustering, usually designated by biclustering, seeks to find sub-matrices, that is subgroups of genes and subgroups of columns, where the genes exhibit highly correlated activities for every condition. This type of algorithms has also been proposed and used in other 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 and the target applications.

Index Terms

Biclustering, simultaneous clustering, co-clustering, two-way clustering, subspace clustering, bi-dimensional clustering, microarray data analysis, biological data analysis

I. 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). 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 [17].

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. This correspond to the:

- Analysis of expression patterns of genes by comparing rows in the matrix.
- Analysis of expression patterns of samples by comparing 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 its expression and the expression of other genes, with known classification.

*A definitive version of this work was published as:
Sara C. Madeira and Arlindo L. Oliveira, **Biclustering algorithms for biological data analysis: a survey**, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 1, no. 1, pp. 24-45, 2004.

Sara C. Madeira is affiliated with University of Beira Interior, Covilhã, Portugal. She is also a researcher at INESC-ID, Lisbon, Portugal. She can be reached by email at smadeira@di.ubi.pt.

Arlindo Oliveira is affiliated with Instituto Superior Técnico and INESC-ID, Lisbon, Portugal. His email is aml@inesc-id.pt.

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 co-regulated and co-expressed 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 algorithmic approaches capable of discovering local patterns in microarray data [2].

Clustering methods can be applied to either the rows or the columns of the data matrix, separately. Biclustering methods, on the other hand, perform clustering in the two dimensions simultaneously. This means that clustering methods derive a *global model* while biclustering algorithms produce 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. 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 co-active under all conditions.

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

- A cluster of genes should be defined with respect to only a subset of the conditions.
- A cluster of conditions should be defined with respect to only a subset of the genes.
- The clusters should not be exclusive and/or exhaustive: a gene or 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 or genes, respectively.

Additionally, robustness in biclustering algorithms is specially 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.

II. Definitions and Problem Formulation

We will be working with an n by m matrix, where 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 I illustrates the arrangement of a gene expression matrix.

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 elements 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, $X = \{x_1, ..., x_n\}$, and its set of columns, $Y = \{y_1, ..., y_m\}$. We will use (X, Y) to denote the matrix A. If $I \subseteq X$ and $J \subseteq Y$

TABLE I
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}

are subsets of the rows and columns, respectively, $A_{IJ} = (I, J)$ denotes the sub-matrix A_{IJ} of A that contains only the elements a_{ij} belonging to the sub-matrix with set of rows I and set of columns J.

Given the data matrix A a *cluster of rows* is 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 $I = \{i_1, ..., i_k\}$ 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 sub-matrix of the data matrix A. Similarly, a *cluster of columns* is a subset of columns that exhibit similar behavior across the set of all rows. A cluster $A_{XJ} = (X, J)$ is a subset of columns defined over the set of all rows X, where $J = \{j_1, ..., j_s\}$ 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 sub-matrix of the data 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 a subset of rows and a subset of columns where $I = \{i_1, ..., i_k\}$ is a subset of rows ($I \subseteq X$ and $k \leq n$), and $J = \{j_1, ..., j_s\}$ is a subset of columns ($J \subseteq Y$ and $s \leq m$). A bicluster (I, J) can then be defined as a k by s sub-matrix of the data 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 that a bicluster must obey vary from approach to approach, and will be studied in Section III.

A. 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).

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

B. 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 [20].

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.

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

III. Bicluster Type

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

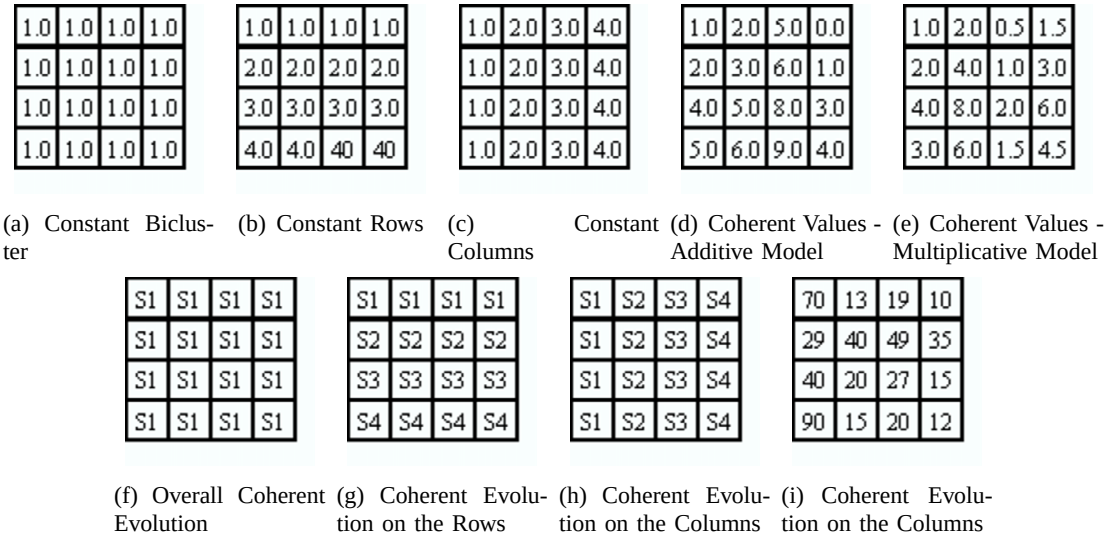
1) Biclusters with constant values.
2) Biclusters with constant values on rows or columns.
3) Biclusters with coherent values.
4) Biclusters with coherent evolutions.

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

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. 1(b) is an example of a bicluster with constant rows, while the bicluster depicted in Fig. 1(c) is an example of a bicluster with constant columns. Section III-C 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 Fig. 1(d) and Fig. 1(e) are examples of this type of bicluster, where each row and column can be obtained by adding a constant to each of the others or by multiplying each of the others by a constant value. These algorithms are studied in Section III-D.

The last type of biclustering approaches we analyzed addresses the problem of finding biclusters with coherent evolutions. These approaches view the elements of the matrix as symbolic values, and try to discover subsets of rows and subsets of columns with coherent behaviors regardless of the exact numeric values in the data matrix. The co-evolution property can be observed on the entire bicluster, that is on both rows and columns of the sub-matrix (see Fig. 1(f)), on the rows of the bicluster (see Fig. 1(g)), or on the columns of the bicluster (see Fig. 1(h) and Fig. 1(i)). These approaches are addressed in Section III-E.



(a) Constant Bicluster (b) Constant Rows (c) Constant Columns (d) Coherent Values - Additive 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

Fig. 1. Examples of Different Types of Biclusters

According to the specific properties of each problem, one or more of these different types of biclusters are 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 at finding.

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 both 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. When this is the case, the quality of the bicluster is not directly evaluated. One-way clustering metrics are used to evaluate the quality of the clustering performed on each of the two dimensions separately and are then combined, in some way, to compute a measure of the quality of the resulting biclustering. The type of biclusters produced by two-way clustering algorithms depends, then, on the distance or similarity measure used by the one-way clustering algorithms. These algorithms will be considered in Sections III-B, III-C, III-D and III-E, depending on the type of bicluster defined by the distance or similarity measure used.

A. Notation

We will now introduce some notation used in the remaining of the section. Given the data matrix $A = (X, Y)$, with set of rows X and set of columns Y, a bicluster is a sub-matrix (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 data matrix A corresponding to row 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. These values are defined by:

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

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

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

B. 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 data matrix in order to group together similar rows and similar columns, and discover subsets of rows and subsets of columns (biclusters) with similar values. Since this approach only produces good results when it is performed on non-noisy data, which does not correspond to the great majority of available data, more sophisticated approaches can be used to pursue the goal of finding biclusters with constant values. When gene expression data is used, constant biclusters reveal subsets of genes with similar expression values within a subset of conditions. The bicluster in Fig. 1(a) is an example of a bicluster with constant values.

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

$$a_{ij} = \mu \tag{4}$$

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 or some metric based on it.

Hartigan [13] 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 sub-matrices (biclusters). The *variance* is used to evaluate the quality of each bicluster (I, J):

$$VAR(I, J) = \sum_{i \in I, j \in J} (a_{ij} - a_{IJ})^2 \tag{5}$$

According to this criterion, a perfect bicluster is a sub-matrix 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 since $VAR(I, J) = 0$. 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 $k \in 1, ..., K$. The algorithm stops when the data matrix is partitioned into K biclusters. The quality of the resulting biclustering is computed using the overall variance of the K biclusters:

$$VAR(I, J)_K = \sum_{k=1}^{K} \sum_{i \in I, j \in J} (a_{ij} - a_{IJ})^2 \tag{6}$$

Although Hartigan's goal was to find constant biclusters, he mentioned the possibility to change its merit function in order to make it possible to find biclusters with constant rows, constant columns or coherent values on both rows and columns. He suggested the use of a two-way analysis of the variance within the bicluster, and a possible requirement that the bicluster be of low rank, suggesting the possibility of an ANalysis Of VAriance between groups (ANOVA).

Tibshirani et al. [26] added a backward pruning method to the block splitting algorithm introduced by Hartigan [13] and designed a permutation-based method to induce the optimal number of biclusters, K. The merit function used is however, still the variance and consequently it finds constant biclusters.

Another approach that aims at finding biclusters with constant values is the Double Conjugated Clustering (DCC) introduced by Busygin et al. [4]. DCC is a two-way clustering approach to biclustering that enables the use of any clustering algorithm within its framework. 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. By doing this, they also identify biclusters with constant values.

C. 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 values on the rows or the columns of the data matrix. The biclusters in Fig. 1(b) and Fig. 1(c) are examples of biclusters with constant rows and constant columns, respectively. In the case of gene expression data, 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. The same reasoning can be applied to identify 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.

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

$$a_{ij} = \mu + \alpha_i \tag{7}$$

$$a_{ij} = \mu \times \alpha_i \tag{8}$$

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 (7) or multiplicative way (8).

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

$$a_{ij} = \mu + \beta_j \tag{9}$$

$$a_{ij} = \mu \times \beta_j \tag{10}$$

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 or by computing similarities between the rows and columns of the data matrix as we have seen in Section III-B.

The straightforward approach to identify non-constant 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 Fig. 1(b) and Fig. 1(c), would both be transformed into the bicluster presented in Fig. 1(a), which is a constant bicluster. This means that the row and column normalization allows 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.

This approach was followed by Getz et al. [11], who 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. By doing this preprocessing step, they manage to transform the biclusters of the type presented in Fig. 1(c) into biclusters of the type shown in Fig. 1(a), making it possible to discover a set of biclusters with constant values on their columns.

Califano et al. [5] 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 $i \in I$:

$$\max(a_{ij}) - \min(a_{ij}) < \delta, \forall j \in J \tag{11}$$

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. In particular, Califano et al. want 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. [23] 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 allow the identification of biclusters with constant values on the columns of the data matrix. 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. [21] [22] 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.

D. 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. In the case of gene expression data, we can be interested in identifying more complex biclusters where a subset of genes and a subset of conditions have coherent values on both rows and columns. The biclusters in Fig. 1(d) and Fig. 1(e) 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 (7), (8), (9) and (10). More sophisticated approaches perform an analysis of variance between groups and use a particular form of co-variance 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 III-C, 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} are predicted using the following expression:

$$a_{ij} = \mu + \alpha_i + \beta_j \tag{12}$$

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. 1(d) 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 Fig. 1(b) and Fig. 1(c) can be considered special cases of this general additive model where the coherence of values can be observed on the rows and on the columns of the bicluster, respectively. This means that (7) and (9) are special cases of the model represented by (12) 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:

$$a_{ij} = \mu' \times \alpha'_i \times \beta'_j \tag{13}$$

These approaches are effectively equivalent to the additive model in (12), when $\mu = \log \mu'$, $\alpha_i = \alpha'_i$ and $\beta_j = \beta'_j$. In this model each element a_{ij} in the data matrix 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. 1(e) 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 Fig. 1(b) and Fig. 1(c) can also be considered special cases of this multiplicative model, since (8) and (10) are special cases of (13) when $\alpha'_i = 0$ and $\beta'_j = 0$, respectively.

Several biclustering algorithms attempt to discover biclusters with coherent values assuming either additive or multiplicative models.

Cheng and Church [6] 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 sub-matrix (I, J) having a low mean squared residue score. A sub-matrix (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 Fig. 1(b), Fig. 1(c) and Fig. 1(d) 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:

$$a_{ij} = a_{iJ} + a_{Ij} - a_{IJ} \tag{14}$$

Note that this corresponds to considering the additive model in (12) and using $\mu = a_{IJ}$, $\alpha_i = a_{iJ} - a_{IJ}$ and $\beta_j = a_{Ij} - a_{IJ}$.

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) was defined as follows:

$$r(a_{ij}) = a_{ij} - a_{iJ} - a_{Ij} + a_{IJ} \tag{15}$$

Assuming the possible existence of residue, the value of a_{ij} in a non-perfect bicluster is then defined as:

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

where the value of the residue is an indicator of the coherence of a value relatively to the remaining values in the bicluster given the biases of the relevant rows and the relevant columns. The lower the residue, the stronger the coherence.

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. The mean squared residue score is given by:

$$H(I,J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} r(a_{ij})^2 \tag{17}$$

Using this merit function makes it possible to find biclusters with coherent values across both rows and columns since a score $H(I, J) = 0$ indicates that the values in the data matrix fluctuate in unison. This includes, as a particular case, biclusters with constant values, which were addressed in Section III-B.

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. [29] [30] generalized the definition of a δ-bicluster to cope with missing values and avoid the interference caused by the random fillins 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 (non-missing) values of the rows and columns considered. The FLOC (FLexible Overlapped biClustering) algorithm introduced an *occupancy* threshold, ϑ, and defined a δ-bicluster of ϑ occupancy as a sub-matrix (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} and $r(a_{ij})$ and $H(I, J)$, in (1), (2), (3) and (15), respectively, so that their calculus does not take into account missing values:

$$a_{iJ} = \frac{1}{|J'_i|} \sum_{j \in J'_i} a_{ij} \tag{18}$$

$$a_{Ij} = \frac{1}{|I'_j|} \sum_{i \in I'_j} a_{ij} \tag{19}$$

$$a_{IJ} = \frac{1}{\upsilon_{IJ}} \sum_{i \in I'_i, j \in J'_j} a_{ij} \tag{20}$$

$$r(a_{ij}) = \begin{cases} a_{ij} - a_{iJ} - a_{Ij} + a_{IJ} & \text{, if } a_{ij} \text{ is specified} \\ 0 & \text{, otherwise} \end{cases} \tag{21}$$

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 square mean. The arithmetic mean was used in [29]:

$$H(I,J) = \frac{1}{\upsilon_{IJ}} \sum_{i \in I'_j, j \in J'_i} |r(a_{ij})| \tag{22}$$

The square mean was used in [30] and redefines Cheng and Church's score, which was defined in (17), as follows:

$$H(I,J) = \frac{1}{\upsilon_{IJ}} \sum_{i \in I'_j, j \in J'_i} r(a_{ij})^2 \tag{23}$$

Wang el al. [28] also assumed the additive model in (12) and seek to discover δ-pClusters. Given a sub-matrix (I, J) of A, they consider each 2×2 sub-matrix $M = (I_{i_1 i_2}, J_{j_1 j_2})$ 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:

$$pscore(M) = |(a_{i_1 j_1} - a_{i_1 j_2}) - (a_{i_2 j_1} - a_{i_2 j_2})| \tag{24}$$

They consider that the sub-matrix (I, J) is a δ-pCluster if for any 2×2 sub-matrix $M \subset (I, J)$, $pscore(M) < \delta$. 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. 1(d). 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 (13). An example of a perfect δ-pCluster modeled using a multiplicative model is the one presented in Fig. 1(e).

Kluger et al. [16] 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 (13). 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. [25] 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 pre-defined 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, it also permits the identification of biclusters with coherent values represented by the multiplicative model in (13).

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 [17] 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:

$$a_{ij} = \sum_{k=0}^{K} \theta_{ijk} \rho_{ik} \kappa_{jk} \tag{25}$$

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.

Lazzeroni and Owen [17] want to obtain a plaid model, which describes the interactions between the several biclusters on the data matrix and minimizes the following merit function:

$$\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} \left(a_{ij} - \theta_{ij0} - \sum_{k=1}^{K} \theta_{ijk} \rho_{ik} \kappa_{jk} \right)^2 \tag{26}$$

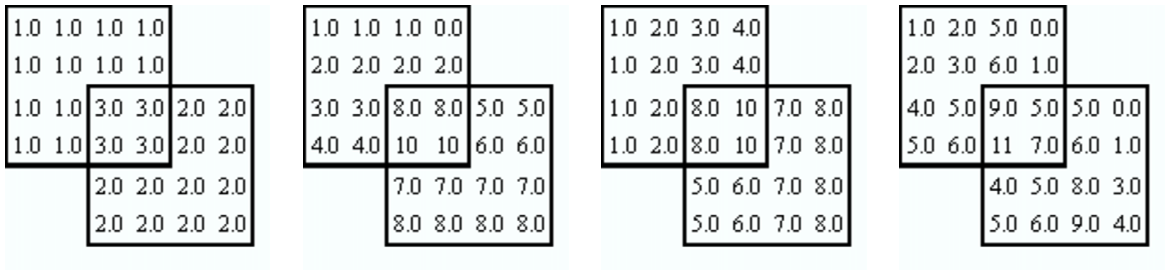
where the term θ_{ij0} considers the possible existence of a single bicluster that covers the whole matrix and that explains away some variability that is not particular to any specific bicluster.

The plaid model described in (25) can be seen as a generalization of the additive model presented in (12). We will call this model the *general additive model*. For every element a_{ij} it represents a sum of

additive models each representing the contribution of the bicluster $(I, J)_k$ to the value of a_{ij} in case $i \in I$ and $j \in J$.

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 (4) in Section III-B). When $\theta_{ijk} = \mu_k + \alpha_{ik}$ the plaid model identifies a set of biclusters with constant rows (see (7) in Section III-C). Similarly, when $\theta_{ijk} = \mu_k + \beta_{jk}$ biclusters with constant columns are found (see (9) in Section III-C). Finally, when $\theta_{ijk} = \mu_k + \alpha_{ik} + \beta_{jk}$ the plaid model identifies biclusters with coherent values across a set of rows and columns by assuming the additive model in (12) for every bicluster k to whom the row i and the column j belong.

Fig. 2(a) to Fig. 2(d) show examples of different types of overlapping biclusters described by a general additive model where the values in the data matrix are seen as a sum of the contributions of the different biclusters they belong to.



(a) Constant Biclusters (b) Constant Rows (c) Constant Columns (d) Coherent Values

Fig. 2. Overlapping Biclusters with General Additive Model

Segal et al. [21] [22] also assumed the additive model in (12), 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 (25)). 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 (12), $\theta_{ijk} = \mu_k + \beta_{jk}$ and $\kappa_{jk} = 1$, for all columns j and all biclusters k in (25). This was in fact the reason why this approach was classified as producing biclusters with constant columns and addressed in Section III-C. 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 the following expression:

$$\sum_{k=1}^{K} \frac{(a_{ijk} - \theta_{ijk}\rho_{ik})^2}{2\sigma_k^2} \tag{27}$$

$$a_{ij} = \sum_{k=1}^{K} a_{ijk} \tag{28}$$

where a_{ijk} is the sum of the predicted value for the element a_{ij} in each bicluster k, which is computed using (25) 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.

Following this reasoning, an obvious extension to (26) 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 most general form of the expression to be minimized is, therefore:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} \frac{(a_{ij} - \theta_{ij0} - \sum_{k=1}^{K} \theta_{ijk}\rho_{ik}\kappa_{jk})^2}{2(\sigma_{iJ}^2 + \sigma_{Ij}^2 + \sigma_{IJ}^2)} \tag{29}$$

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,

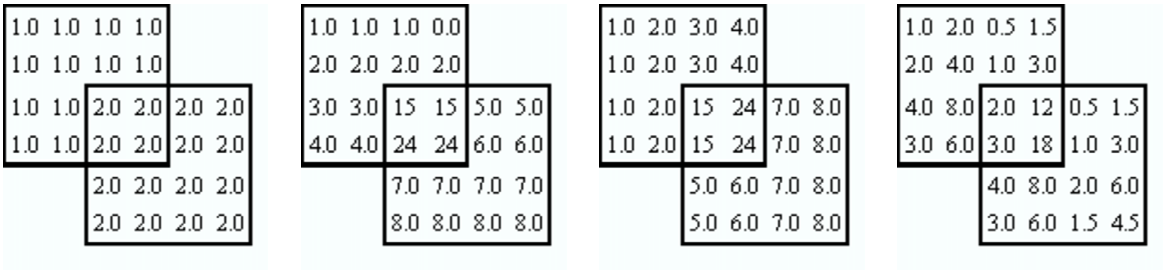
that are know to exhibit a higher degree of variability.

Other 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 data matrix is given by the product of the contributions of the different biclusters to whom the row i and the column j belong, instead of a sum of contributions as it is considered by the plaid model. In this approach, which we will call *general multiplicative model*, the value of each element a_{ij} in the data matrix is given by the following expression:

$$a_{ij} = \prod_{k=0}^{K} \theta_{ijk}\rho_{ik}\kappa_{jk} \tag{30}$$

Similarly to the plaid model that sees a bicluster as a sum of layers (biclusters), (30) 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 (13) instead of being defined by the additive model in (12), as the plaid model was. Fig. 3(a) to Fig. 3(d) 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 (30) with θ_{ijk} given by the additive model in (12). 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 (25) with θ_{ijk} given by the multiplicative model in (13). This corresponds to considering 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 ((12) and (25)) and the general multiplicative model ((13) and (30)).



(a) Constant Biclusters (b) Constant Rows (c) Constant Columns (d) Coherent Values

Fig. 3. Overlapping Biclusters with General Multiplicative Model

The previous biclustering algorithms used either an additive or multiplicative model to produce biclusters with coherent values and can for this reason be put in the same framework. In these bicluster models, a background value is used together with the row and column effects to predict the values within the bicluster and find bicluster that satisfy a certain coherence criterion regarding their values. The last approaches we analyzed consider that the value of a given element in the data matrix can be seen as a sum of the contributions of the different biclusters to whom its rows and columns belong, while the other consider the contribution of a bicluster at a time. We also looked at the possibility to consider the values in the data matrix as a product of the contributions of several biclusters. Nevertheless, all the previously surveyed biclustering algorithms try to discover sets of biclusters by analyzing directly the values a_{ij} in the data matrix A.

E. 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. In the case of gene expression data, we may be interested in looking for evidence that a subset of genes is up-regulated or down-regulated across a subset of conditions without taking into account their actual expression values in the data matrix. The co-evolution property can be observed on both rows and columns of the biclusters, as it is shown in Fig. 1(f), on the rows of the bicluster or on its columns. The biclusters presented in Fig. 1(h) and Fig. 1(i) are examples of biclusters with coherent evolutions on the columns, while Fig. 1(g) shows a bicluster with co-evolution on the rows.

Ben-Dor et al. [2] defined a bicluster as an order-preserving sub-matrix (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 focus 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 [17] did. More specifically, they want to identify large OPSMs. A sub-matrix 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. 1(i) is an example of an OPSM, where $a_{i4} \leq a_{i2} \leq a_{i3} \leq a_{i1}$, and represents a bicluster with coherent evolution 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 $\pi = (j_1, j_2, ..., j_s)$ 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. The statistical significance of a given OPSM is thus computed using an upper-bound on the probability that a random data matrix of size n-by-m will contain a complete model of size s with k or more rows supporting it. In the case of gene expression data such a sub-matrix 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 co-expressed patterns for large sets of genes. They also considered that, in many cases, data contains more than one such pattern.

Following the same idea, Liu and Wang [18] 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. 1(i) is an example of an OPSM and also of an OP-Cluster.

Murali and Kasif [19] 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 up-regulated and down-regulated, when only two states are considered. An example of a perfect bicluster in this approach is the one presented in Fig. 1(g), where Si is the symbol representing the preserved state of the row (gene) i.

Murali and Kasif assumed that 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. Together with this conservation condition, 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. Note that this approach is similar to the one followed by Ben-Dor et al. [2]. Ben-Dor et al. considered that rows (genes) have only two states (up-regulated and down-regulated) and looked for a group of rows whose states induce some linear order across a subset of the columns (conditions). This means that the

expression level of the genes in the bicluster increased or decreased from condition to condition. Murali and Kasif [19] consider that rows (genes) can have a given number of states and look for a group of columns (conditions) within which a subset of the rows is in the same state.

Tanay et al. [24] 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. SAMBA's goal is to discover biclusters (sub-graphs) with an overall coherent evolution. In order to do that it is assumed that all the genes in a given bicluster are up-regulated in the subset of conditions that form the bicluster and the goal is then to find the largest biclusters with this co-evolution property. As such, SAMBA does not try to find any kind of coherence on the values, a_{ij}, in the bicluster. It assumes that regardless of its true value, a_{ij} is either 0 or 1, where 1 is up-regulation and 0 is down-regulation. A large bicluster is thus one with a maximum number of genes (rows) whose value a_{ij} is expected to be 1 (up-regulation). The bicluster presented in Fig. 1(f) is an example of the type of bicluster SAMBA produces, if we say that $S1$ is the symbol that represents a coherent overall up-regulation evolution. The merit function used to evaluate the quality of a computed bicluster using SAMBA is the weight of the sub-graph 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 sub-graph is defined as the sum of the weights of gene-condition (row-column) pairs in it including edges and non-edges, the goal is thus to assign weights to the vertex pairs of the bipartite sub-graph so that heavy sub-graphs correspond to significant biclusters.

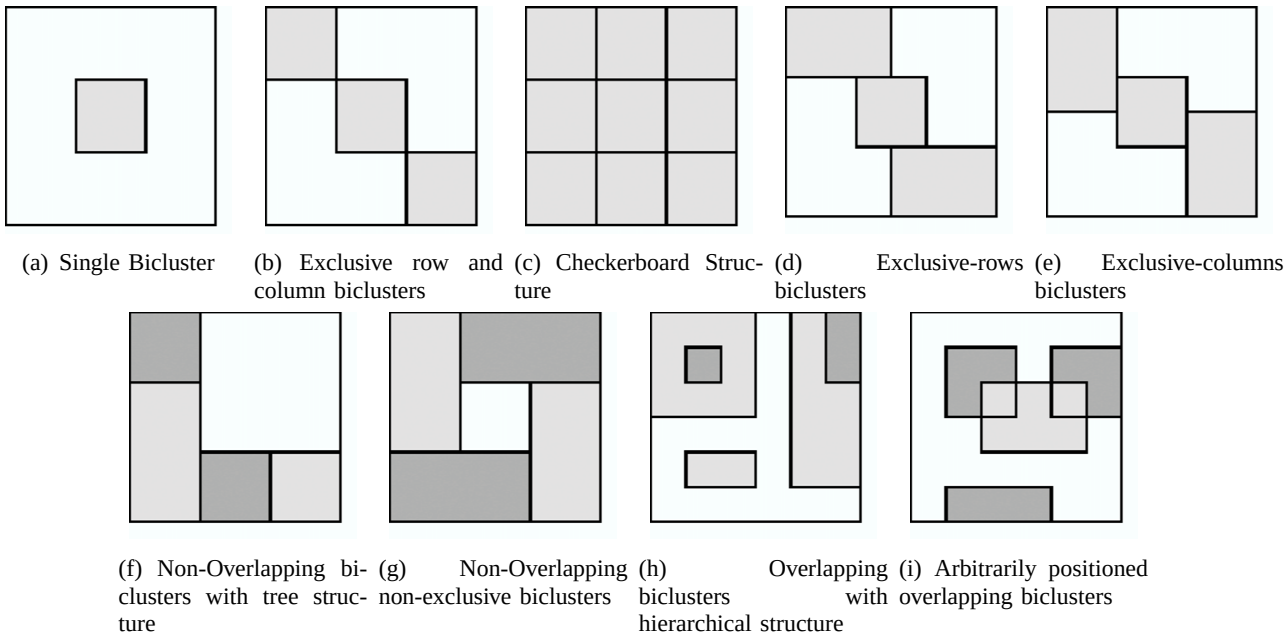
IV. Bicluster Structure

Biclustering algorithms assume one of the following situations: either there is only *one bicluster* in the data matrix (see Fig. 4(a)), or the data 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 in the data matrix [13] [6] [11] [5] [17] [21] [25] [29] [4] [24] [30] [16] [23] [22] [18], 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 one considered the best according to some criterion [2] [19].

When the biclustering algorithm assumes the existence of several biclusters in the data matrix, the following bicluster structures can be obtained (see Fig. 4(b) to Fig. 4(i)):

1) Exclusive row and column biclusters (rectangular diagonal blocks after row and column reorder).
2) Non-Overlapping biclusters with checkerboard structure.
3) Exclusive-rows biclusters.
4) Exclusive-columns biclusters.
5) Non-Overlapping biclusters with tree structure.
6) Non-Overlapping non-exclusive 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 data matrix would produce an image with some number K of rectangular blocks on the diagonal (see Fig. 4(b)). Each block



(a) Single Bicluster (b) Exclusive row and column biclusters (c) Checkerboard Structure (d) Exclusive-rows biclusters (e) Exclusive-columns biclusters

(f) Non-Overlapping biclusters with tree structure (g) Non-Overlapping non-exclusive biclusters (h) Overlapping biclusters with hierarchical structure (i) Arbitrarily positioned overlapping biclusters

Fig. 4. Bicluster Structure

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 biclustering 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 data matrix belongs exclusively to one of the K biclusters considered (see Fig. 4(b)). 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, which would lead to such a bicluster structure, will seldom exist in real data [17].

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. 4(c)). By doing this we allow the existence of K non-overlapping and non-exclusive biclusters where each row in the data matrix belongs to exactly K biclusters. The same applies to columns. Kluger et al. [16] assumed this structure on cancer data. The Double Conjugated Clustering (DCC) approach introduced by Busygin et al. [4] also makes it possible to identify this biclustering structure. However, DCC tends to produce the structure in Fig. 4(b).

Other biclustering 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. 4(d), assumes exclusive-rows biclusters and was used by Sheng et al. [23] and Tang et al. [25]. However, these approaches can also produce exclusive-columns biclusters when the algorithm is used using the opposite orientation of the data matrix. This means that the columns of the data matrix can only belong to one bicluster while the rows can belong to one or more biclusters (see Fig. 4(e)).

The structures presented in Fig. 4(b) to Fig. 4(e) assume that the biclusters are exhaustive, that is, every row and every column in the data matrix belongs at least to one bicluster. However, we can consider non-exhaustive variations of these structures that make it possible that some rows and columns do not belong to any bicluster. Other exhaustive bicluster structures, include the tree structure considered by Hartigan [13] and Tibshirani et al. [26] and that is depicted in Fig. 4(f), and the structure in Fig. 4(g). A non-exhaustive variation of the structure presented in Fig. 4(g) was assumed by Wang et al. [28]. None of these structures allow overlapping, that is, none of these structures 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, 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 is assumed. This structure, depicted in Fig. 4(h), requires that either the biclusters are disjoint or one includes the other. Two specializations of this structure, are the tree structure presented in Fig. 4(f), where the biclusters form a tree, and the checkerboard structure depicted in Fig. 4(c), where the biclusters, the row clusters and the column clusters are all trees.

A more general bicluster structure permits the existence of K possibly overlapping biclusters without taking into account their direct observation in the data matrix with a common reordering of its rows and columns. Furthermore, these non-exclusive biclusters can also be non-exhaustive, which means that some rows or columns may not belong to any bicluster. Several biclustering algorithms [6] [17] [11] [5] [25] [21] [24] [2] [19] [22] [18] allow this more general structure, which is presented in Fig. 4(i).

The plaid model [17], defined in (25), can be used to describe most of these different biclusters structures. The restriction that every row and every column are in exactly one bicluster correspond to the conditions $\sum_k \rho_{ik} = 1$, for all i, and $\sum_k \kappa_{jk} = 1$ for all j. To allow overlapping it is necessary that $\sum_k \rho_{ik} \geq 2$, for some i, and $\sum_k \kappa_{jk} \geq 2$, for some j. Similarly, allowing that some rows or columns do not belong to any bicluster corresponds to the restrictions $\sum_k \rho_{ik} = 0$, for some i, and $\sum_k \kappa_{jk} = 0$, for some j. This means that without any of these constrains, the plaid model represents the data matrix as a sum of possibly overlapping biclusters as presented in Fig. 4(i).

V. 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 [6] and Sheng et al. [23], for instance, identify a bicluster at a time, mask it with random numbers, and repeat the procedure in order to eventually find other biclusters. Lazzeroni and Owen [17] also attempt to discover one bicluster at a time in an iterative process where a plaid model is obtained. Ben-Dor et al. [2] also identify one bicluster at a time.

Other biclustering approaches discover *one set of biclusters at a time*. Hartigan [13] identifies two biclusters at the time by splicing each existing bicluster into two pieces at each iteration. CTWC [11] 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 [25].

We also analyzed algorithms that perform *simultaneous bicluster identification*, which means that the biclusters are discovered all at the same time. FLOC [29], [30] 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. Muraly and Kasif [19] 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. [24] use SAMBA to performs simultaneous bicluster identification using exhaustive bicluster enumeration, but restricting the number of rows the biclusters may have. Liu and Yand [18] and Yang et al. [28] also used exhaustive bicluster enumeration to perform simultaneous biclustering identification. The approaches followed by Busygin et al. [4], Kluget et al. [16] and Califano et al. [5] 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, studied in the following five subsections:

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 perform bicluster identification 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 V-A.

Other approaches 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 these solutions to create a solution to the original problem [7]. These biclustering approaches are described in Section V-B.

A large number of methods, studied in section V-C, 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 [7].

Some authors propose methods that perform *exhaustive bicluster enumeration*. A number of methods have been used to speed up exhaustive search, in some cases assuming restrictions on the size of the biclusters that should be listed. These algorithms are revised in Section V-D.

The last type of approach we identified performs *distribution parameter identification*. These approaches 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. Section V-E describes these approaches.

A. 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) [11] 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 sub-clusters. 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) [25] is an iterative biclustering 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 data matrix. The task in this step is to cluster n_1 rows into K groups, denoted as $I_i, i = 1, ..., K$, each of which is an exclusive subset of the set of all rows X. The clustering technique used can be any method that receives the number of clusters. 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 $I_i, i = 1, ..., k$, 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, $i = 1, ..., 4$, 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), $s, t = 1, ..., 4$. Heterogeneous pair 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 of the matrix 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 sequence I' for each heterogeneous group. In order to select the row set I' that should be chosen for the next iteration of the algorithm they use cross-validation. After this final step the number of rows are reduced from n_1 to n_2 and the above five steps can be repeated using the n_2 selected rows until the termination conditions of the algorithm are satisfied.

The Double Conjugated Clustering (DCC) [4] 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, respectively, 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.

B. Divide-and-Conquer

Divide and conquer algorithms have the significant advantage of being potentially very fast. However, they 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 [13] who called it direct clustering (see Section III-B). 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 splicing 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 [10] suggested the use of permutation tests to determine when a given block split is not significant. Following this direction, Tibshirani et al. [26] added a backward pruning method to the block splitting algorithm introduced by Hartigan [13] 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 a pruning criterion is used at the end.

C. 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. They may make wrong decisions and loose good biclusters, but they have the potential to be very fast.

Cheng and Church [6] were the first to apply biclustering to gene expression data. Given a data matrix A and a maximum acceptable mean squared residue score (see (17)), $\delta > 0$, 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 III-D). 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 starts 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 [29], [30] addresses this limitation (see Section III-D). 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 non-missing elements) when computing the score (see (23)). This means that missing values are not used when computing the row mean, the column mean and the bicluster mean needed to compute the score (see (18), (19) and (20)). FLOC avoids the introduction of random interference and discovers K possibly overlapping biclusters simultaneously.

FLOC 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 towards reducing the overall mean score residue. 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.

Ben-Dor et al. [2] addressed the identification of large order-preserving submatrices (OPSMs) with maximum statistical significance (see Section III-E). 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 $\pi = (j_1, j_2, ..., j_s)$ 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 $< j_1, ..., j_a >$ and the indices of the b "largest" elements $< j_{s-(b-1)}, ..., j_s >$ 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 [19] introduced a biclustering algorithm that aims at finding xMOTIFs. An xMOTIF is a bicluster with coherent evolutions on its rows (see Section III-E). The data is discretized to a set of symbols by using a list of statistical significant intervals, for each row (gene, in the case of expression data), representing the states in which the gene is expressed within the set of conditions (columns).

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 motif. 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. If one knows a condition c that matches the largest xMOTIF in the data matrix, the goal of the algorithm is thus to compute the rows that belong to this largest xMOTIF and the states they are in.

In order to pursue the goal of finding the largest xMOTIF, Murali and Kasif define a *discriminating set* as a set of conditions, D, with the following two properties: (i) for every column c' in D and for every row in the largest motif, there is one state such that the row is in that state in columns c and c'; and (ii) for every row r that is not in the largest motif, there exists a column c' in D such that row r is not in the same state in columns c and c'. The key property of a discriminating set is that given the seed column c and such a set D, it is possible to compute the largest xMOTIF by including exactly those row-states that satisfy these properties and exactly those columns that agree with c on all these row-states.

The xMOTIF algorithm works by assuming that, for each row, there are a set of intervals corresponding to gene states, which are computed in the first step of the algorithm. This corresponds to a step of discretization of the continuous data to a set of discrete symbols. The algorithm then proceeds by selecting n_s columns uniformly at random from the set of all columns. These columns act as seeds. Having done this, for each previously selected seed, c, the algorithm selects n_d sets of columns uniformly at random from the set of all columns. These sets have s_d elements each and serve as candidates for the discriminating set, D. For each pair (seed, discriminating set) the corresponding xMOTIF is computed as explained above. The computed motif is discarded if less than an α-fraction of the columns match it. After all the seeds have been used to produce xMOTIFs, the largest xMOTIF, that is, the one that contains the largest number of rows, is returned.

Califano et al. [5] introduced an algorithm that addresses the problem of finding δ-valid ks-patterns (see Section III-D). 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 candidate to become statistically significant biclusters (other candidates generated by the pattern discovery 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 III-D).

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 pre-aligned 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 δ-valid (see Section III-D) are selected. Non-maximal subsets that are completely contained within another subset are removed. Each subset is considered a potential super-patterns of a maximal pattern. All possible maximal combinations of these super-pattern are then created iteratively. As a result, all patterns that exists in the data matrix are generated hierarchically by pattern combination.

D. 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 data 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.

Tanay et al. [24] 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 II-A, and define as their objective the identification of a maximum weight sub-graph, assuming that the weight of a sub-graph will correspond to its statistical significance.

Discovering the most significant biclusters in the data matrix under this weighting schemes is equivalent to the selection of the heaviest sub-graphs in the model bipartite graph. SAMBA assumes that row vertices have d-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.

The first phase of the SAMBA algorithm 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 the algorithm finds the K heaviest bicliques in the graph. This is done by looking at a pre-computed 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 of this procedure, 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. [28] also proposed an algorithm that performs exhaustive bicluster enumeration, subject to a restriction that they should posses 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 MDS 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 [12] 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 post-order 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 post-order traversal of this tree will generate all the existing biclusters in the matrix. The number of biclusters and, therefore, the execution time, can be exponential on the number of columns in the matrix, however.

Liu and Wang [18] 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 objective of finding all 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 [14].

The structure they use to perform efficient enumeration of all common patterns in the rows uses an OPC-tree. An OPC tree 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 matrix 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 sub-trees of n_c in the child of n that has a label that matches the symbol that is in the root of the sub-tree. This procedure, complemented by appropriate pruning operations performed when there is not enough quorum to reach the target minimum bicluster dimension, generates all possible order preserving biclusters.

E. Distribution Parameter Identification

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

Lazzeroni and Owen [17] want to obtain a plaid model that minimizes the merit function defined in (26). 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:

$$Q = \tfrac{1}{2}\textstyle\sum_{i=1}^{n}\sum_{j=1}^{m}\left(Z_{ij} - \theta_{ijK}\rho_{iK}\kappa_{jK}\right)^2 \tag{31}$$

$$Z_{ij} = a_{ij} - \theta_{ij0} - \textstyle\sum_{k=1}^{K-1}\theta_{ijk}\rho_{ik}\kappa_{jk} \tag{32}$$

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 quite similar to the Expectation-Maximization (EM) algorithm. Lagrange Multipliers are used to estimate the parameters and improve the objective 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 $s = 1, ..., S$ 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 (31) is minimized subject to the restrictions that every row and column has zero mean. The same reasoning is applied to estimate the remaining parameters. Furthermore, given a set of K layers, the θ_{ijk} values can be re-estimated, by cycling through $k = 1, ..., K$ in turn.

Segal et al. [21] [22] attempt to estimate the activity of each column (condition) in each process (bicluster) by minimizing the expression in (27). This is performed in an iterative process where the model parameters used to generate each bicluster are updated all at the same time. The EM algorithm is use to estimate the parameters.

Sheng et al. [23] introduced a biclustering approach based on Gibbs sampling. The row-column (gene-condition) orientation of the data matrix is assumed although 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 they do 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, $i = 1, ..., n$, 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, the goal of step three of the algorithm is to fix the labels of the columns. In order to do that, for every column j, $j = 1, ..., m$, the labels for all the other columns are fixed while the Bernoulli distribution for the given column j is computed. A label is then assigned to column j from the computed distribution. The parameters for both row and column distributions are estimated using Gibbs sampling. The algorithm iteratively goes back to the second step during 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. This means that this approach discovers one bicluster at a time. Moreover, the rows retrieved for previous biclusters are no longer selected as candidate rows for any future bicluster, while the background model will still be calculated over all possible columns in the whole data matrix including the positions of the masked rows or columns. Note that this choice does allow the unmasked dimension of the bicluster to be selected multiple times. Furthermore, in the case of row-column orientation, a column can be relevant to multiple biclusters. In this way, the algorithm is iterated on the data matrix until no bicluster can be found for the unmasked part of the data matrix.

Kluger et al. [16] used a spectral approach to biclustering by assuming that the data matrix contains a checkerboard structure after normalization. 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 $A' A'^T y = \lambda^2 y$, 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 re-scaling of rows and columns) assumes the non-normalized 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 (13)). 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 (bi-stochastization) 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 (12) and (13)). 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 (15).

VI. Overall Comparison of the Biclustering Algorithms

Table II presents a summary of the different biclustering algorithms in accordance with the different dimensions of analysis considered. The second column classifies the algorithms according to the type of biclusters they aim at finding (see Section III). Column three lists the biclustering approaches according to the bicluster structure they can produce. The notation used is the one in Fig. IV in Section IV. The last two

columns summarize Section V by classifying the different algorithms according to the way they discover the biclusters and the approach they use to achieve their goal. 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).

TABLE II
OVERALL COMPARISON OF THE BICLUSTERING ALGORITHMS

	Type	Structure	Discovery	Approach
Block Clustering [13]	Constant	4(f)	One Set at a Time	Div-and-Conq
δ-biclusters [6]	Coherent Values	4(i)	One at a Time	Greedy
FLOC [29], [30]	Coherent Values	4(i)	Simultaneous	Greedy
pClusters [28]	Coherent Values	4(g)	Simultaneous	Exh-Enum
Plaid Models [17]	Coherent Values	4(i)	One at a Time	Dist-Ident
PRMs [21], [22]	Coherent Values	4(i)	Simultaneous	Dist-Ident
CTWC [11]	Constant Columns	4(i)	One Set at a Time	Clust-Comb
ITWC [25]	Coherent Values	4(d)/4(e)	One Set at a Time	Clust-Comb
DCC [4]	Constant	4(b)/4(c)	Simultaneous	Clust-Comb
δ-Patterns [5]	Constant Rows	4(i)	Simultaneous	Greedy
Spectral [16]	Coherent Values	4(c)	Simultaneous	Greedy
Gibbs [23]	Constant Columns	4(d)/4(e)	One at a Time	Dist-Ident
OPSMs [2]	Coherent Evolution	4(a)/4(i)	One at a Time	Greedy
SAMBA [24]	Coherent Evolution	4(i)	Simultaneous	Exh-Enum
xMOTIFs [19]	Coherent Evolution	4(a)/4(i)	Simultaneous	Greedy
OP-Clusters [18]	Coherent Evolution	4(i)	Simultaneous	Exh-Enum

VII. 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. An example of this kind of data are the gene expression matrices. Moreover, it can be applied when the data can be modeled as a weighted bipartite graph as we explained in Section II-A. Furthermore, biclustering can be used when the goal is to identify sub-matrices described by a subset of rows and a subset of columns with certain coherence properties.

Large datasets of clinical samples are an ideal target for biclustering [24]. As such, many 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. In this application domain, we can use biclusters to associate genes with specific clinical classes or for classifying samples, among other possible interesting applications. The applications of biclustering to biological data analysis are discussed in Section VII-A.

However, and even though most recent applications of biclustering are in biological data analysis, there exist many other possible applications in very different application domains. Examples of these application areas are: information retrieval and text mining; collaborative filtering, recommendation systems, and target marketing; database research and data mining; and even analysis of electoral data. Some non-biological applications of biclustering are presented in Section VII-B.

A. Biological Applications

Cheng and Church [6] applied biclustering to two gene expression data matrices, specifically to the Yeast Saccharomyces Cerevisiae cell cycle expression data with 2884 genes and 17 conditions and the

human B-cells expression data with 4026 genes and 96 conditions. Yang et al. [29], [30] also used these two datasets. Wang et al. [28] and Liu and Wang [18] also used the Yeast data.

Lazzeroni et al. [17] also used biclustering to identify biclusters in Yeast gene expression data: the rows of the data matrix represented 2467 genes and the columns were time points within each of 10 experimental conditions. Furthermore, experiments one to three examined the mitotic cell cycle; experiments four to six tracked different strains of Yeast during sporulation; experiments seven to nine tracked expression following exposure to different types of shocks and experiment ten studied the diauxic shift.

Segal et al. [21], [22] used two gene expression data matrices. They first analyzed the Yeast stress data, which characterizes the expression patterns of yeast genes under different experimental conditions by selecting 954 genes with significant changes in gene expression and the full set of 92 conditions. Their model identifies groupings based on similarities of gene expression, the presence of known transcription factor binding sites within the gene promoters and functional annotation of genes. They identify expected gene clusters, that display similar gene expression patterns and are known to function in the same metabolic processes. They also discover new groupings of genes based on both expression levels and transcription factor binding sites. Secondly, they used the Yeast Compendium data, which observed the genomic expression programs triggered by specific gene mutations. The goal of these experiments is to assign hypothetical functions to uncharacterized genes by their deletion to known expression programs. They selected 528 genes and 207 conditions, focusing on genes and mutations that had some functional annotations in the MIPS database.

Getz et al. [11] applied biclustering to two gene expression data matrices containing cancer data. The first data matrix was constituted by 72 samples collected from acute Leukemia patients at the time of diagnosis using RNA prepared from the bone marrow mononuclear cells of 6817 human genes: 47 cases were diagnosed as ALL (Acute Lymphoblastic Leukemia) and the other 25 as AML (Acute Myeloid Leukemia). They identified a possible diagnosis to leukemia by identifying different responses to treatment, and the groups of genes to be used as the appropriate probe. Busygin et al. [4] and Kluger et al. [16] also used these Leukemia data. The second gene expression matrix used by Getz et al. contained 40 colon tumor samples and 22 normal colon samples and 6500 human genes from which they choosed the 2000 of greatest minimal expression over the samples. Muraly and Kasif [19] also used these two datasets.

Sheng et al. [23] also used leukemia expression data. The data matrix was this time constituted by 72 samples collected from acute Leukemia patients which were now classified into three types of Leukemia: 28 cases were diagnosed as ALL (Acute Lymphoblastic Leukemia), 24 as AML (Acute Myeloid Leukemia) and the remaining 20 as MLL (Mixed-Linkage Leukemia). The expression level of 12600 human genes was available.

Tang et al. [25] applied ITWC to a gene expression matrix with 4132 genes and 48 samples of Multiple Sclerosis patients and Ben-Dor et al. [2] used a breast tumor dataset with gene expression data from 3226 genes under 22 experimental conditions.

Tanay et al. [24] used SAMBA to perform functional annotation in Yeast data using an expression matrix with 6200 genes and 515 experimental conditions. They also applied biclustering to human cancer data. The Lymphoma dataset they used is characterized by well defined expression patterns differentiating three types of lymphoma: Chronic Lymphocytic Leucemia (CLL), Diffuse Large B-Cell Lymphoma (DLBCL) and Follicular Lymphoma (FL).

Kluger et al. [16] also used the Lymphoma expression data used by Tanay et al. but also applied biclustering to two extra gene expression matrices: a breast tumor dataset and a central nervous system embryonal tumor dataset.

All the previous applications of biclustering analyzed biological data from gene expression matrices obtained from microarray experiments. However, biclustering can also reveal to be interesting to analyze other kind of biological data. For example, Liu and Wang used a dataset with drug activity data: a matrix with 10000 rows and 30 columns where each row corresponds to a chemical compound and each column represents a descriptor/feature of the compound. The values in the data matrix ranged from 0 to 1000.

B. Other Applications

Biclustering techniques can be used in collaborative filtering to identify subgroups of customers with similar preferences or behaviors towards a subset of products with the goal of performing 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. In these applications the goal is thus to identify sets of customers with similar behavior so that we can predict the customers' interest and make proper recommendations.

Yang et al. [29], [30] used the MovieLens dataset collected by the GroupLens Research Project at the University of Minnesota. This dataset consists of a data matrix where the rows represent 943 customers and the columns represent 1682 movies. The values a_{ij} in the data matrix are integers from 1 to 10 representing the rate that customer i assigned to the movie j. Since a customer only rated less than 10% of the movies on average, the data matrix is only partially filled with 100000 ratings. Wang et al. [28] also used the MovieLens data.

Hoffman and Puzicha [15] also applied biclustering to collaborative filtering using the EachMovie dataset, which consisted of data collected on the Internet with almost three million preference votes on a 0-5 scales. Ungar and Foster [27] also used a movie dataset where the value a_{ij} was 1 if the person j watched the movie i, and 0 otherwise. Both Hoffman and Puzicha [15] and Ungar and Foster [27] used biclustering approaches similar to the one presented by Sheng et al. [23]. While Ungar and Foster used the Expectation-Maximization (EM) algorithm, Hoffman and Puzicha used Gibbs sampling.

In information retrieval and text mining, biclustering can be applied to identify subgroups of documents with similar properties relatively to subgroups of attributes, such as words or images. These information can be very important in query and indexing in the domain of search engines.

Dhillon [8] used biclustering to perform simultaneous clustering of documents and words by considering a word-by-document matrix A where the rows correspond to words, the columns to documents, and a non-zero element a_{ij} indicates the presence of word i in document j: $a_{ij} = o_{ij} \times log(n/n_i)$, where o_{ij} is the number of occurrences of word i in document j, n is the number of documents and n_i is the number of documents (rows) that contain the word i. This type of matrix is called *incidence matrix* in this context and the term co-clustering is generally used instead of biclustering. Both document clustering and word clustering are well studied problems in the are of information retrieval and text mining. However, most existing algorithms cluster documents and words separately and not simultaneously. One one hand, given a collection of unlabeled documents, document clustering can help in organizing the collection thereby facilitating future navigation and search. On the other hand, words may be clustered on the basis of the documents where they co-occur. Clusters of words have been used in applications such as the automatic construction of statistical thesaurus, the enhancement of queries and the automatic classification of documents. Dhillon wanted to identify subsets of words and subsets of documents strongly related to each other by modeling the data matrix as a bipartite graph as Tanay et al. [24] did and using a spectral approach similar to the one used by Kluger et al. [16]. He used three document collections: Medline (1033 medical abstracts), Cranfield (1400 aeronautical systems abstracts) and Cisi (1460 information retrieval abstracts). Other biclustering application with this kind of data matrices was presented by Dhillon et al. [9] and Berkhin and Becher [3].

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 is addressed by Agrawal et al. [1].

More exotic applications of biclustering use data matrices with electoral data and try to identify biclusters to discover subgroups of rows with the same political ideas and electoral behaviors among a subset of the attributes considered.

Hartigan [13] applied biclustering to two data sets: voting data consisting of the percentage of Republican vote for the President of the United States, in the southern states, over the years 1900–1968; and voting data consisting of the UN votes in 1969–1970. In the first case, the data matrix consisted of a set of rows representing states and a set of columns represented years. Each value a_{ij} represented the percentage of votes of state i in year j. The goal was to detect clusters of rows, that is, groups of states that vote similarly, and clusters of columns, that is, years for which the votes are similar. A bicluster in this case is a subset of states with similar votes in a subset of years. In the case of the second data matrix, the rows represented countries and the columns propositions about discussed issues. The goal was to identify clusters of countries with similar interests or political systems, and clusters of propositions, identifying series of propositions about the same underlying issues. A bicluster is this time a subset of countries with similar votes within a subset of discussed prepositions.

We can also think of many other applications of biclustering using other datasets. Lazzeroni et al. [17] applied biclustering to nutritional data and to a foreign exchange example. The nutritional data consisted of a data matrix with 961 rows representing different foods and a set of columns representing the following food attributes: grams of fat, calories of food energy, grams of carbohydrate, grams of protein, milligrams of cholesterol, grams of saturated fat, and the weight of the food item in grams. The goal was to identify subsets of foods with similar properties on a subset of the attributes considered. The foreign exchange data matrix consisted of monthly foreign exchange: the rows in the data matrix were 18 currencies corresponding to 18 countries and the columns in the data matrix represented 277 months from January 1977 to January 2000. The values a_{ij} in the data matrix correspond to the number of units of currency i that one US dollar purchased in month j. The goal was to identify subsets of currencies (rows) where the US dollar had similar behavior within a subset of months (columns).

VIII. 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 III to VI, 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 the general 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 that are 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 biclustering applications presented, long as it is already, represents, on our view, only a small fraction of the potential applications of this type of techniques. Many other domains of applications in biological data analysis, gene network identification, data mining, text 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. Other 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.

ACKNOWLEDGEMENT

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] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulus, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the ACM/SIGMOD International Conference on Management of Data*, pages 94–105, 1998.

[2] Amir Ben-Dor, Benny Chor, Richard Karp, and Zohar Yakhini. Discovering local structure in gene expression data: The order–preserving submatrix problem. In *Proceedings of the 6th International Conference on Computacional Biology (RECOMB'02)*, pages 49–57, 2002.

[3] Pavel Berkhin and Jonathan Becher. Learning simple relations: theory and applications. In *Proceedings of the 2nd SIAM International Conference on Data Mining*, pages 420–436, 2002.

[4] Stanislav Busygin, Gerrit Jacobsen, and Ewald Kramer. Double conjugated clustering applied o leukemia microarray data. In *Proceedings of the 2nd SIAM International Conference on Data Mining, Workshop on Clustering High Dimensional Data*, 2002.

[5] Andrea Califano, Gustavo Stolovitzky, and Yunai Tu. Analysis of gene expression microarays for phenotype classification. In *Proceedings of the International Conference on Computacional Molecular Biology*, pages 75–85, 2000.

[6] Yizong Cheng and George M. Church. Biclustering of expression data. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB'00)*, pages 93–103, 2000.

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Cliffoord Stein. *Introduction to Algorithms*. The MIT Electrical Engineering and Computer Science Series. The MIT Press, 2nd edition, 2001.

[8] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'01)*, pages 269–274, 2001.

[9] Inderjit S. Dhillon, Subramanyam Mallela, and Dharmendra S. Modha. Information-theoretical co-clustering. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*, pages 89–98, 2003.

[10] D. Duffy and A. Quiroz. A permutation based algorithm for block clustering. *Journal of Classification*, 8:65–91, 1991.

[11] G. Getz, E. Levine, and E. Domany. Coupled two-way clustering analysis of gene microarray data. In *Proceedings of the Natural Academy of Sciences USA*, pages 12079–12084, 2000.

[12] Dan Gusfield. *Algorithms on strings, trees, and sequences*. Computer Science and Computational Biology Series. Cambridge University Press, 1997.

[13] J. A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association (JASA)*, 67(337):123–129, 1972.

[14] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. Algorithms for association rule mining – a general survey and comparison. *SIGKDD Explorations*, 2(1):58–64, July 2000.

[15] Thomas Hofmann and Jaz Puzicha. Latent class models for collaborative filtering. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 668–693, 1999.

[16] Yuval Klugar, Ronen Basri, Joseph T. Chang, and Mark Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. In *Genome Research*, volume 13, pages 703–716, 2003.

[17] Laura Lazzeroni and Art Owen. Plaid models for gene expression data. Technical report, Stanford University, 2000.

[18] Jinze Liu and Wei Wang. Op-cluster: Clustering by tendency in high dimensional space. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 187–194, 2003.

[19] T. M. Murali and Simon Kasif. Extracting conserved gene expression motifs from gene expression data. In *Proceedings of the Pacific Symposium on Biocomputing*, volume 8, pages 77–88, 2003.

[20] Ren Peeters. The maximum edge biclique problem is NP-complete. *Discrete Applied Mathematics*, 131(3):651–654, 2003.

[21] Eran Segal, Ben Taskar, Audrey Gasch, Nir Friedman, and Daphne Koller. Rich probabilistic models for gene expression. In *Bioinformatics*, volume 17 (Suppl. 1), pages S243–S252, 2001.

[22] Eran Segal, Ben Taskar, Audrey Gasch, Nir Friedman, and Daphne Koller. Decomposing gene expression into cellular processes. In *Proceedings of the Pacific Symposium on Biocomputing*, volume 8, pages 89–100, 2003.

[23] Qizheng Sheng, Yves Moreau, and Bart De Moor. Biclustering micrarray data by gibbs sampling. In *Bioinformatics*, volume 19 (Suppl. 2), pages ii196–ii205, 2003.

[24] Amos Tanay, Roded Sharan, and Ron Shamir. Discovering statistically significant biclusters in gene expression data. In *Bioinformatics*, volume 18 (Suppl. 1), pages S136–S144, 2002.

[25] Chun Tang, Li Zhang, Idon Zhang, and Murali Ramanathan. Interrelated two-way clustering: an unsupervised approach for gene expression data analysis. In *Proceedings of the 2nd IEEE International Symposium on Bioinformatics and Bioengineering*, pages 41–48, 2001.

[26] R. Tibshirani, T. Hastie, M. Eisen, D. Ross, D. Botstein, and P. Brown. Clustering methods for the analysis of DNA microarray data. Technical report, Department of Health Research and Policy, Department of Genetics and Department of Biochemestry, Stanford University, 1999.

[27] Lyle Ungar and Dean P. Foster. A formal statistical approach to collaborative filtering. In *Proceedings of the Conference on Automated Learning and Discovery (CONALD'98)*, 1998.

[28] Haixun Wang, Wei Wang, Jiong Yang, and Philip S. Yu. Clustering by pattern similarity in large data sets. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 394–405, 2002.

[29] Jiong Yang, Wei Wang, Haixun Wang, and Philip Yu. δ-clusters: Capturing subspace correlation in a large data set. In *Proceedings of the 18th IEEE International Conference on Data Engineering*, pages 517–528, 2002.

[30] Jiong Yang, Wei Wang, Haixun Wang, and Philip Yu. Enhanced biclustering on expression data. In *Proceedings of the 3rd IEEE Conference on Bioinformatics and Bioengineering*, pages 321–327, 2003.