Algorithms for Molecular Biology BioMed Central

Research

Open Access

A polynomial time biclustering algorithm for finding approximate expression patterns in gene expression time series

Sara C Madeira*[1,2,3] and Arlindo L Oliveira[1,2]

Address: [1]Knowledge Discovery and Bioinformatics (KDBIO) group, INESC-ID, Lisbon, Portugal, [2]Instituto Superior Técnico, Technical University of Lisbon, Lisbon, Portugal and [3]University of Beira Interior, Covilhã, Portugal

Email: Sara C Madeira* - smadeira@kdbio.inesc-id.pt; Arlindo L Oliveira - aml@inesc-id.pt

* Corresponding author

Published: 4 June 2009

Algorithms for Molecular Biology 2009, **4**:8 doi:10.1186/1748-7188-4-8

Received: 14 July 2008
Accepted: 4 June 2009

This article is available from: http://www.almob.org/content/4/1/8

Abstract

Background: The ability to monitor the change in expression patterns over time, and to observe the emergence of coherent temporal responses using gene expression time series, obtained from microarray experiments, is critical to advance our understanding of complex biological processes. In this context, biclustering algorithms have been recognized as an important tool for the discovery of local expression patterns, which are crucial to unravel potential regulatory mechanisms. Although most formulations of the biclustering problem are NP-hard, when working with time series expression data the interesting biclusters can be restricted to those with contiguous columns. This restriction leads to a tractable problem and enables the design of efficient biclustering algorithms able to identify all maximal contiguous column coherent biclusters.

Methods: In this work, we propose e-CCC-Biclustering, a biclustering algorithm that finds and reports all maximal contiguous column coherent biclusters with approximate expression patterns in time polynomial in the size of the time series gene expression matrix. This polynomial time complexity is achieved by manipulating a discretized version of the original matrix using efficient string processing techniques. We also propose extensions to deal with missing values, discover anticorrelated and scaled expression patterns, and different ways to compute the errors allowed in the expression patterns. We propose a scoring criterion combining the statistical significance of expression patterns with a similarity measure between overlapping biclusters.

Results: We present results in real data showing the effectiveness of e-CCC-Biclustering and its relevance in the discovery of regulatory modules describing the transcriptomic expression patterns occurring in *Saccharomyces cerevisiae* in response to heat stress. In particular, the results show the advantage of considering approximate patterns when compared to state of the art methods that require exact matching of gene expression time series.

Discussion: The identification of co-regulated genes, involved in specific biological processes, remains one of the main avenues open to researchers studying gene regulatory networks. The ability of the proposed methodology to efficiently identify sets of genes with similar expression patterns is shown to be instrumental in the discovery of relevant biological phenomena, leading to more convincing evidence of specific regulatory mechanisms.

Availability: A prototype implementation of the algorithm coded in Java together with the dataset and examples used in the paper is available in http://kdbio.inesc-id.pt/software/e-ccc-biclustering.

Background

Time series gene expression data, obtained from microarray experiments performed in successive instants of time, can be used to study a wide range of biological problems [1], and to unravel the mechanistic drivers characterizing cellular responses [2]. Being able to monitor the change in expression patterns over time, and to observe the emergence of coherent temporal responses of many interacting components, should provide the basis for understanding evolving but complex biological processes, such as disease progression, growth, development, and drug responses [2]. In this context, several machine learning methods have been used in the analysis of gene expression data [3]. Recently, biclustering [4-6], a non-supervised approach that performs simultaneous clustering on the gene and condition dimensions of the gene expression matrix, has been shown to be remarkably effective in a variety of applications. The advantages of biclustering in the discovery of local expression patterns, described by a coherent behavior of a subset of genes in a subset of the conditions under study, have been extensively studied and documented [4-8]. Recently, Androulakis et al. [2] have emphasized the fact that biclustering methods hold a tremendous promise as more systemic perturbations are becoming available and the need to develop consistent representations across multiple conditions is required. Madeira et al. [9] have also described the use of biclustering as critical to identify the dynamics of biological systems as well as the different groups of genes involved in each biological process. However, most formulations of the biclustering problem are NP-hard [10], and almost all the approaches presented to date are heuristic, and for this reason, not guaranteed to find optimal solutions [6]. In a few cases, exhaustive search methods have been used [7,11], but limits are imposed on the size of the biclusters that can be found [7] or on the size of the dataset to be analyzed [11], in order to obtain reasonable runtimes. Furthermore, the inherent difficulty of this problem when dealing with the original real-valued expression matrix and the great interest in finding coherent behaviors regardless of the exact numeric values in the matrix, has led many authors to a formulation based on a discretized version of the expression matrix [7-9,12-23]. Unfortunately, the discretized versions of the biclustering problem remain, in general, NP-hard. Nevertheless, in the case of time series expression data the interesting biclusters can be restricted to those with contiguous columns leading to a tractable problem. The key observation is the fact that biological processes are active in a contiguous period of time, leading to increased (or decreased) activity of sets of genes that can be identified as biclusters with contiguous columns. This fact led several authors to point out the relevance of biclusters with contiguous columns and their importance in the identification of regulatory mechanisms [9,20,22,24].

In this work, we propose *e*-CCC-Biclustering, a biclustering algorithm specifically developed for time series expression data analysis, that finds and reports all maximal contiguous column coherent biclusters with approximate expression patterns in time polynomial in the size of the expression matrix. The polynomial time complexity is obtained by manipulating a discretized version of the original expression matrix and by using efficient string processing techniques based on suffix trees. These approximate patterns allow a given number of errors, per gene, relatively to an expression profile representing the expression pattern in the bicluster. We also propose several extensions to the core *e*-CCC-Biclustering algorithm. These extensions improve the ability of the algorithm to discover other relevant expression patterns by being able to deal with missing values directly in the algorithm and by taking into consideration the possible existence of anti-correlated and scaled expression patterns. Different ways to compute the errors allowed in the approximate patterns (restricted errors, alphabet range weighted errors and pattern length adaptive errors) can also be used. Finally, we propose a statistical test that can be used to score the biclusters discovered (by extending the concept of statistical significance of an expression pattern [9] to cope with approximate expression patterns) and a method to filter highly overlapping, and, therefore, redundant, biclusters. We report results in real data showing the effectiveness of the approach and its relevance in the process of identifying regulatory modules describing the transcriptomic expression patterns occurring in *Saccharomyces cerevisiae* in response to heat stress. We also show the superiority of *e*-CCC-Biclustering when compared with state of the art biclustering algorithms, specially developed for time series gene expression data analysis such as CCC-Biclustering [9,22].

Related Work: Biclustering algorithms for time series gene expression data

Although many algorithms have been proposed to address the general problem of biclustering [5,6], and despite the known importance of discovering local temporal patterns of expression, to our knowledge, only a few recent proposals have addressed this problem in the specific case of time series expression data [9,20,22,24]. These approaches fall into one of the following two classes of algorithms:

1. Exhaustive enumeration: CCC-Biclustering [9,22] and *q*-clustering [20].

2. Greedy iterative search: CC-TSB algorithm [24].

These three biclustering approaches work with a single time series expression matrix and aim at finding biclusters defined as subsets of genes and subsets of **contiguous**

time points with coherent expression patterns. CCC-Biclustering and *q*-clustering work with a discretized version of the expression matrix while the CC-TSB-algorithm works with the original real-valued expression matrix. In additional file 1: related_work we describe in detail these algorithms and identify their strengths and weaknesses. Based on their characteristics, we decided to compare the performance of *e*-CCC-Biclustering with that of CCC-Biclustering, but not with that of the *q*-clustering and CC-TSB algorithms. The decision to exclude the last two algorithms from the comparisons is mainly based on existing analysis of these algorithms [9], and is basically related with complexity issues, in the case of *q*-clustering, and on poor results on real data obtained by the heuristic approach used by the CC-TSB algorithm.

Biclusters in discretized gene expression data

Let A' be an $|R|$ row by $|C|$ column gene expression matrix defined by its set of rows (genes), R, and its set of columns (conditions), C. In this context, A'_{ij} represents the expression level of gene i under condition j. In this work, we address the case where the gene expression levels in matrix A' can be discretized to a set of symbols of interest, Σ, that represent distinctive activation levels. After the discretization process, matrix A' is transformed into matrix A, where $A_{ij} \in \Sigma$ represents the discretized value of the expression level of gene i under condition j (see Figure 1 for an illustrative example).

Given matrix A we define the concept of bicluster and the goal of biclustering as follows:

Definition 1 (Bicluster) *A bicluster is a sub-matrix A_{IJ} defined by $I \subseteq R$, a subset of rows, and $J \subseteq C$, a subset of columns. A bicluster with only one row or one column is called trivial.*

The goal of biclustering algorithms is to identify a set of biclusters $B_k = (I_k, J_k)$ such that each bicluster satisfies specific characteristics of homogeneity. These characteristics vary in different applications [6]. In this work we will deal with biclusters that exhibit coherent evolutions:

Definition 2 (CC-Bicluster) *A column coherent bicluster A_{IJ} is a bicluster such that $A_{ij} = A_{lj}$ for all rows $i, l \in I$ and columns $j \in J$.*

Finding all maximal biclusters satisfying this coherence property is known to be an NP-hard problem [10].

CC-Biclusters in discretized gene expression time series

Since we are interested in the analysis of time series expression data, we can restrict the attention to potentially overlapping biclusters with arbitrary rows and contiguous columns [9,20,22,24]. This fact leads to an important complexity reduction and transforms this particular version of the biclustering problem into a tractable problem. Previous work in this area [9,22] has defined the concept of CC-Biclusters in time series expression data and the important notion of maximality:

Definition 3 (CCC-Bicluster) *A contiguous column coherent bicluster A_{IJ} is a subset of rows $I = \{i_1, ..., i_k\}$ and a subset of **contiguous** columns $J = \{r, r + 1, ..., s - 1, s\}$ such that $A_{ij} = A_{lj}$, for all rows $i, l \in I$ and columns $j \in J$. Each CCC-Bicluster defines a string S that is common to every row in I for the columns in J.*

	C1	C2	C3	C4	C5
G1	0.07	0.73	-0.54	0.45	0.25
G2	-0.34	0.46	-0.38	0.76	-0.44
G3	0.22	0.17	-0.11	0.44	-0.11
G4	0.70	0.71	-0.41	0.33	0.35
G5	0.70	0.17	0.70	- 0.33	0.75

	C1	C2	C3	C4	C5
G1	N	U	D	U	N
G2	D	U	D	U	D
G3	N	N	N	U	N
G4	U	U	D	U	U
G5	U	D	U	D	U

Figure 1

Illustrative example of the discretization process. This figure shows: **(Left)** Original expression matrix A'; and **(Right)** Discretized matrix A obtained by considering a simple discretization technique, which uses a three symbol alphabet $\Sigma = \{D, N, U\}$. The symbols mean down-regulation (D), up-regulation (U) or no-change (N). In this case, the values $A'_{ij} \in]-0.3, 0.3[$ were discretized to N, and the values $A'_{ij} \leq -0.3$ and $A'_{ij} \geq 0.3$ were discretized to D and U, respectively.

Definition 4 (row-maximal CCC-Bicluster) *A CCC-Bicluster A_{IJ} is row-maximal if we cannot add more rows to I and maintain the coherence property referred in Definition 3.*

Definition 5 (left-maximal and right-maximal CCC-Bicluster) *A CCC-Bicluster A_{IJ} is left-maximal/right-maximal if we cannot extend its expression pattern S to the left/right by adding a symbol (contiguous column) at its beginning/end without changing its set of rows I.*

Definition 6 (maximal CCC-Bicluster) *A CCC-Bicluster A_{IJ} is maximal if no other CCC-Bicluster exists that properly contains A_{IJ}, that is, if for all other CCC-Biclusters A_{LM}, $I \subseteq L \wedge J \subseteq M \Rightarrow I = L \wedge J = M$.*

Lemma 1 *Every maximal CCC-Bicluster is right, left and row-maximal.*

Figure 2 shows the maximal CCC-Biclusters with at least two rows (genes) present in the discretized matrix in Figure 1. CCC-Biclusters with only one row, even when maximal, are trivial and uninteresting from a biological point of view and are thus discarded.

Maximal CCC-Biclusters and generalized suffix trees

Consider the discretized matrix *A* obtained from matrix *A'* using the alphabet Σ. Consider also the matrix obtained by preprocessing *A* using a simple alphabet transformation, that appends the column number to each symbol in the matrix (see Figure 3), and considers a new alphabet $\Sigma' = \Sigma \times \{1, ..., |C|\}$, where each element Σ' is obtained by concatenating one symbol in Σ and one number in the range $\{1, ..., |C|\}$. We present below the two Lemmas and the Theorem describing the relation between maximal CCC-Biclusters with at least two rows and nodes in the generalized suffix tree built from the set of strings

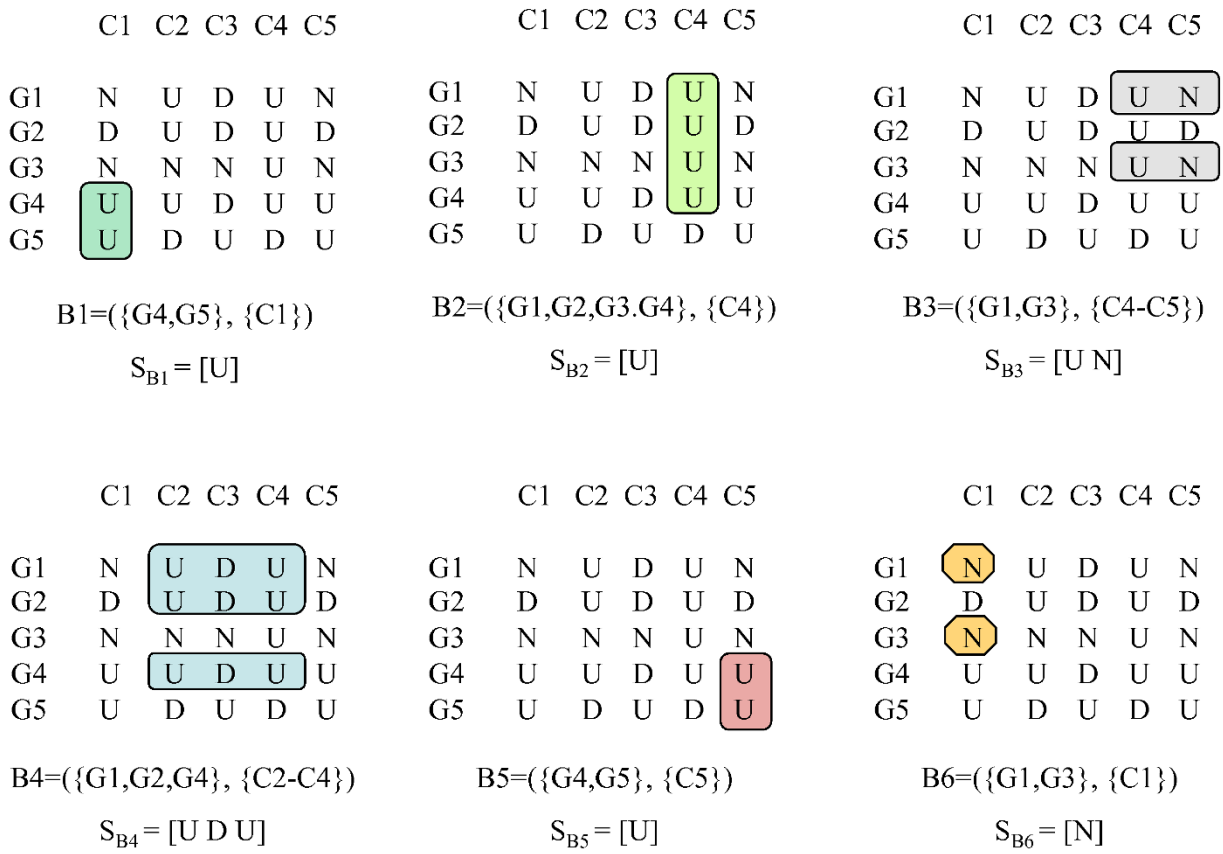


Figure 2

Maximal CCC-Biclusters in a discretized matrix. This figure shows all maximal CCC-Biclusters with at least two rows that can be identified in the discretized matrix in Figure 1. The strings S_{B1} = [*U*], S_{B2} = [*U*], S_{B3} = [*UN*], S_{B4} = [*UDU*], S_{B5} = [*U*] and S_{B6} = [*N*] correspond to the expression patterns of the maximal CCC-Biclusters identified as B1, B2, B3, B4, B5 and B6, respectively.

	C1	C2	C3	C4	C5
G1	N	U	D	U	N
G2	D	U	D	U	D
G3	N	N	N	U	N
G4	U	U	D	U	U
G5	U	D	U	D	U

	C1	C2	C3	C4	C5
G1	N1	U2	D3	U4	N5
G2	D1	U2	D3	U4	D5
G3	N1	N2	N3	U4	N5
G4	U1	U2	D3	U4	U5
G5	U1	D2	U3	D4	U5

Figure 3
Illustrative example of the alphabet transformation performed after the discretization process. This figure shows: **(Left)** Discretized matrix *A* in Figure 1; **(Right)** Discretized matrix *A* after alphabet transformation.

obtained after alphabet transformation [9,22]. Figure 4 illustrates this relation using the generalized suffix tree obtained from the rows in the discretized matrix after alphabet transformation in Figure 3 together with the maximal CCC-Biclusters with at least two rows (B1 to B6) already showed in Figure 2.

Lemma 2 *Every right-maximal, row-maximal CCC-Bicluster with at least two rows corresponds to one internal node in T and every internal node in T corresponds to one right-maximal, row-maximal CCC-Bicluster with at least two rows*.

Lemma 3 *An internal node in T corresponds to a left-maximal CCC-Bicluster iff it is a MaxNode*.

Definition 7 (MaxNode) *An internal node v in T is called a MaxNode iff it satisfies one of the following conditions:*

a) It does not have incoming suffix links.

b) It has incoming suffix links only from nodes u_i such that, for every node u_i, the number of leaves in the subtree rooted at u_i is inferior to the number of leaves in the subtree rooted at v.

Theorem 1 *Every maximal CCC-Bicluster with at least two rows corresponds to a MaxNode in the generalized suffix tree T, and each MaxNode defines a maximal CCC-Bicluster with at least two rows*.

Note that this theorem is the base of CCC-Biclustering [9,22], which finds and reports all maximal CCC-Biclusters using three main steps:

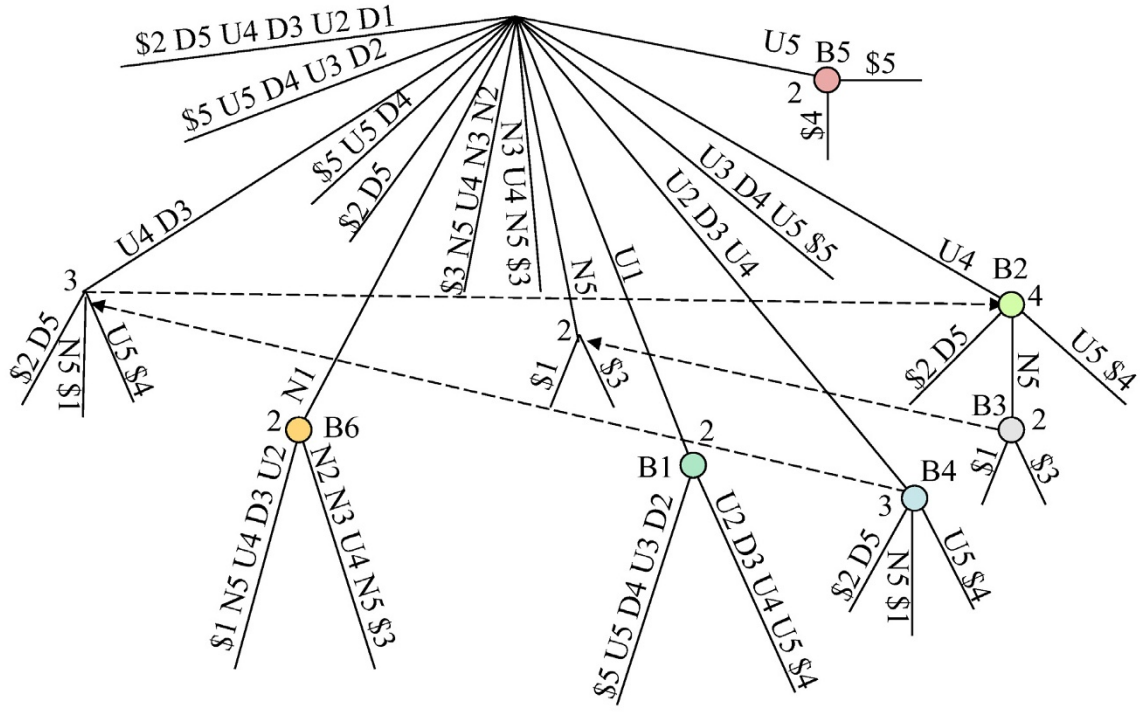
1. All internal nodes in the generalized suffix tree are marked as "*Valid*", meaning each of them identifies a row-maximal, right-maximal CCC-Bicluster with at least two nodes according to Lemma 2.

2. All internal nodes identifying non left-maximal CCC-Biclusters are marked as "*Invalid*" using Theorem 1, discarding all row-maximal, right-maximal CCC-Biclusters which are not left-maximal.

3. All maximal CCC-Biclusters, identified by each node marked as "*Valid*", are reported.

Methods

In this section we propose *e*-CCC-Biclustering, an algorithm designed to find and report all maximal CCC-Biclusters with approximate expression patterns (*e*-CCC-Biclusters) using a discretized matrix *A* and efficient string processing techniques. We first define the concepts of *e*-CCC-Bicluster and maximal *e*-CCC-Bicluster. We then formulate two problems: (1) finding all maximal *e*-CCC-Biclusters and (2) finding all maximal *e*-CCC-Biclusters satisfying row and column quorum constraints. We discuss the relation between maximal *e*-CCC-Biclusters and generalized suffix trees highlighting the differences between this relation and that of maximal CCC-Biclusters and generalized suffix tree, discussed in the previous section. We then discuss and explore the relation between the two problems above and the *Common Motifs Problem* [25,26]. We describe *e*-CCC-Biclustering, a polynomial time algorithm designed to solve both problems and sketch the analysis of its computational complexity. We present extensions to handle missing values, discover anticorrelated and scaled expression patterns, and consider alternative ways to compute approximate expression patterns. Finally, we propose a scoring criterion for *e*-CCC-Biclusters combining the statistical significance of their expression patterns with a similarity measure between overlapping biclusters.



	C1	C2	C3	C4	C5
G1	N1	U2	D3	U4	N5
G2	D1	U2	D3	U4	D5
G3	N1	N2	N3	U4	N5
G4	U1	U2	D3	U4	U5
G5	U1	D2	U3	D4	U5

B1=({G4,G5}, {C1})

S_{B1} = [U]

	C1	C2	C3	C4	C5
G1	N1	U2	D3	U4	N5
G2	D1	U2	D3	U4	D5
G3	N1	N2	N3	U4	N5
G4	U1	U2	D3	U4	U5
G5	U1	D2	U3	D4	U5

B2=({G1,G2,G3.G4}, {C4})

S_{B2} = [U]

	C1	C2	C3	C4	C5
G1	N1	U2	D3	U4	N5
G2	D1	U2	D3	U4	D5
G3	N1	N2	N3	U4	N5
G4	U1	U2	D3	U4	U5
G5	U1	D2	U3	D4	U5

B3=({G1,G3}, {C4-C5})

S_{B3} = [U N]

	C1	C2	C3	C4	C5
G1	N1	U2	D3	U4	N5
G2	D1	U2	D3	U4	D5
G3	N1	N2	N3	U4	N5
G4	U1	U2	D3	U4	U5
G5	U1	D2	U3	D4	U5

B4=({G1,G2,G4}, {C2-C4})

S_{B4} = [U D U]

	C1	C2	C3	C4	C5
G1	N1	U2	D3	U4	N5
G2	D1	U2	D3	U4	D5
G3	N1	N2	N3	U4	N5
G4	U1	U2	D3	U4	U5
G5	U1	D2	U3	D4	U5

B5=({G4,G5}, {C5})

S_{B5} = [U]

	C1	C2	C3	C4	C5
G1	N1	U2	D3	U4	N5
G2	D1	U2	D3	U4	D5
G3	N1	N2	N3	U4	N5
G4	U1	U2	D3	U4	U5
G5	U1	D2	U3	D4	U5

B6=({G1,G3}, {C1})

S_{B6} = [N]

Figure 4 *(see legend on next page)*

Figure 4 *(see previous page)*
Maximal CCC-Biclusters and generalized suffix trees. This figure shows: **(Top)** Generalized suffix tree constructed for the transformed matrix in Figure 3. For clarity, this figure does not contain the leaves that represent string terminators that are direct daughters of the root. Each internal node, other than the root, is labeled with the number of leaves in its subtree. We show the suffix links between nodes although (for clarity) we omit the suffix links pointing to the root. All maximal CCC-Biclusters are identified using a circle. The labels B1 to B6 identify the nodes corresponding to all maximal CCC-Biclusters with at least two rows/genes. Note that the rows in each CCC-Bicluster identified by a given node *v* are obtained from the string terminators in its subtree. The value of the string-depth of *v* and the first symbol in the string-label of *v* provide the information needed to identify the set of contiguous columns. **(Bottom)** Maximal CCC-Biclusters B1 to B6 showed in the discretized matrix as subsets of rows and columns. The strings S_{B1} = [U], S_{B2} = [U], S_{B3} = [U N], S_{B4} = [U D U], S_{B5} = [U] and S_{B6} = [N] correspond to the expression patterns of the maximal CCC-Biclusters identified as B1 to B6, respectively.

CCC-Biclusters with approximate expression patterns

The CCC-Biclusters defined in the previous section are *perfect*, in the sense that they do not allow errors in the expression pattern S that defines the CCC-Bicluster. This means that all genes in I share *exactly* the same expression pattern in the time points in J. Being able to find all maximal CCC-Biclusters using efficient algorithms is useful to identify potentially interesting expression patterns and can be used to discover regulatory modules [9]. However, some genes might not be included in a CCC-Bicluster of interest due to errors. These errors may be measurement errors, inherent to microarray experiments, or discretization errors, introduced by poor choice of discretization thresholds or inadequate number of discretization symbols. In this context, we are interested in CCC-Biclusters with *approximate* expression patterns, that is, biclusters where a certain number of errors is allowed in the expression pattern S that defines the CCC-Bicluster. We introduce here the definitions of *e*-CCC-Bicluster and maximal *e*-CCC-Bicluster preceded by the notion of *e*-neighborhood:

Definition 8 (*e*-Neighborhood) *The e-Neighborhood of a string S of length $|S|$, defined over the alphabet Σ with $|\Sigma|$ symbols, $N(e, S)$, is the set of strings S_i, such that: $|S| = |S_i|$ and Hamming$(S, S_i) \leq e$, where e is an integer such that $e \geq 0$. This means that the Hamming distance between S and S_i is no more than e, that is, we need at most e symbol substitutions to obtain S_i from S.*

Lemma 4 *The e-Neighborhood of a string S, $N(e, S)$, contains $\sum_{j=0}^{e} C_j^{|S|}(|\Sigma|-1)^j \leq |S|^e |\Sigma|^e$ elements.*

Definition 9 (*e*-CCC-Bicluster) *A contiguous column coherent bicluster with e errors **per gene**, e-CCC-Bicluster, is a CCC-Bicluster A_{IJ} where all the strings S_i that define the expression pattern of each of the genes in I are in the e-Neighborhood of an expression pattern S that defines the e-CCC-Bicluster: $S_i \in N(e, S), \forall i \in I$. The definition of 0-CCC-Bicluster is equivalent to that of a CCC-Bicluster.*

Definition 10 (maximal *e*-CCC-Bicluster) *An e-CCC-Bicluster A_{IJ} is maximal if it is row-maximal, left-maximal and right-maximal. This means that no more rows or contiguous columns can be added to I or J, respectively, maintaining the coherence property in Definition 9.*

Given these definitions we can now formulate the problem we solve in this work:

Problem 1 Given a discretized expression matrix A and the integer $e \geq 0$ identify and report all maximal *e*-CCC-Biclusters $B_k = A_{I_k J_k}$.

Similarly to what happened with CCC-Biclusters, *e*-CCC-Biclusters with only one row should be overlooked. A similar problem is that of finding and reporting *only* the maximal *e*-CCC-Biclusters satisfying predefined row and column quorum constraints:

Problem 2 Given a discretized expression matrix A and three integers $e \geq 0$, $q_r \geq 2$ and $q_c \geq 1$, where q_r is the row quorum (minimum number of rows in I_k) and q_c is the column quorum (minimum number of columns in J_k), identify and report all maximal *e*-CCC-Biclusters $B_k = A_{I_k J_k}$ such that, I_k and J_k have at least q_r rows and q_c columns, respectively.

Figure 5 shows all maximal *e*-CCC-Biclusters with at least rows (genes), which are present in the discretized matrix in Figure 1, when one error per gene is allowed ($e = 1$). Figure 6 shows all maximal *e*-CCC-Biclusters identified using row and column constraints. In this case, the maximal 1-CCC-Biclusters having at least three rows and three columns ($q_r = q_c = 3$) are shown. Also clear in these figures is the fact that, when errors are allowed ($e > 0$), different expression patterns S can define the same *e*-CCC-Bicluster. Furthermore, when $e > 0$, an *e*-CCC-Bicluster can be defined by an expression pattern S, which does not occur

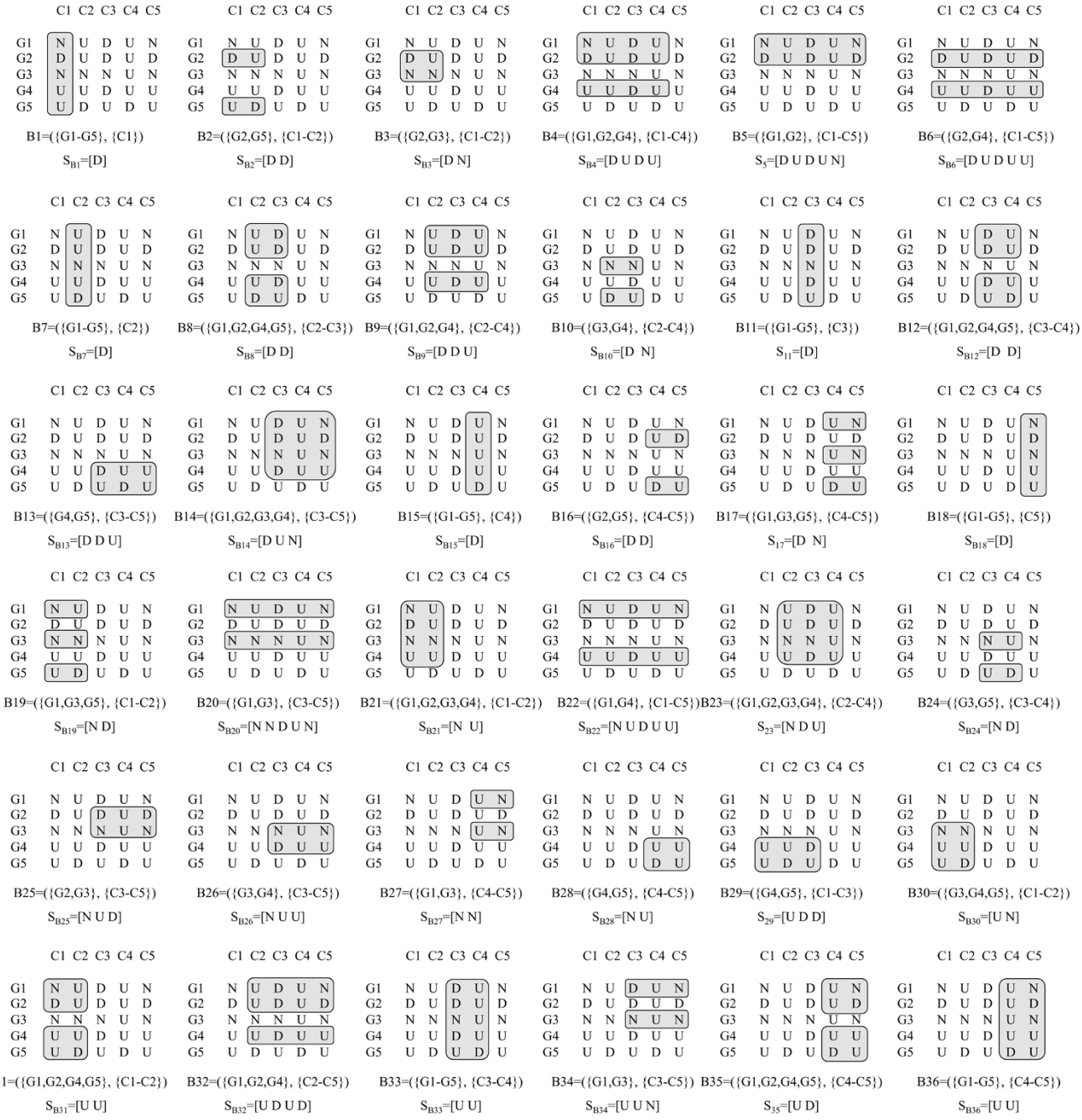


Figure 5

Maximal *e*-CCC-Biclusters in a discretized matrix. This figure shows all maximal 1-CCC-Biclusters with at least two rows that can be identified in the discretized matrix in Figure 1. Note that several of these 1-CCC-Biclusters can be defined by more than one expression pattern. For example, B1 can be defined by S_{B1} = [D], as shown in the figure, but can also be defined by S_{B1} = [N] or S_{B1} = [U]. Other 1-CCC-Biclusters are defined by expression patterns not occurring in the discretized matrix in the contiguous columns identifying the biclusters. This is the case of 1-CCC-Bicluster B2, for example, defined by the pattern S_{B2} = [D D], which does not occur in the columns C1–C2.

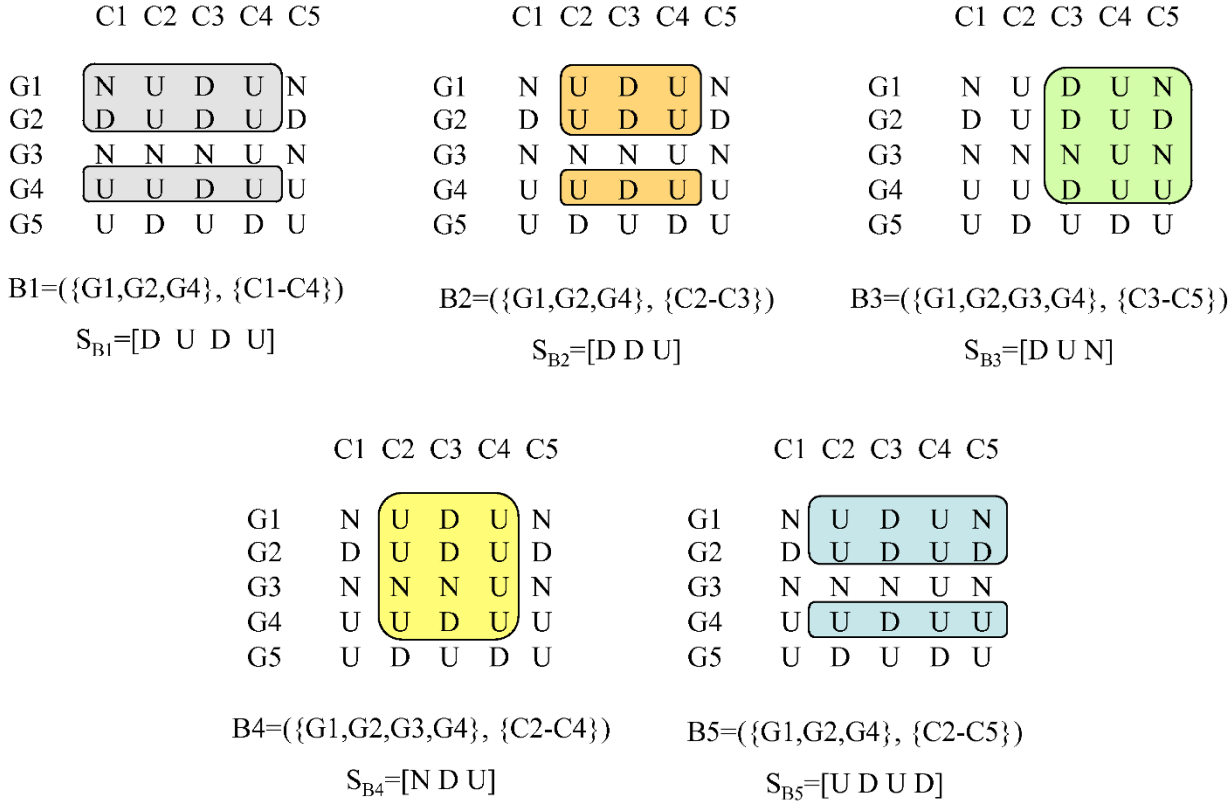


Figure 6

Maximal e-CCC-Biclusters with row and column quorum constraints in a discretized matrix. This figure shows the five maximal 1-CCC-Biclusters with at least 3 rows/columns ($q_r = q_c = 3$) that can be identified in the discretized matrix in Figure 1. These 1-CCC-Biclusters are defined, respectively, by the following patterns: S_{B1} = [D U D U], S_{B2} = [D D U], S_{B3} = [D U N], S_{B4} = [N D U] and S_{B5} = [U D U D]. Also clear from this figure is the fact that the same e-CCC-Bicluster can be defined by several patterns. For example, 1-CCC-Bicluster B1 can also be identified by the patterns [N U D U] and [U U D U]. An interesting example is the case of 1-CCC-Bicluster B2, which can also be defined by the patterns [N D U], [U N U], [U U U], [U D D] and [U D N]. Note however, that B2 cannot be identified by the pattern [U D U]. If this was the case, B2 would not be right maximal, since the pattern [U D N] can be extended to the right by allowing one error at column 5. In fact, this leads to the discovery of the maximal 1-CCC-Bicluster B5. Moreover, e-CCC-Biclusters can be defined by expression patterns not occurring in the discretized matrix. This is the case of 1-CCC-Biclusters B2 and B4, defined respectively by the patterns S_{B2} = [D D U] and S_{B4} = [N D U], which do not occur in the matrix in the contiguous columns defining B2 and B4 (C2–C3 and C2–C4, respectively).

in the discretized matrix in the set of contiguous columns in the *e*-CCC-Bicluster.

Maximal e-CCC-Biclusters and generalized suffix trees

In the previous section we showed that each internal node in the generalized suffix tree, constructed for the set of strings corresponding to the rows in the discretized matrix after alphabet transformation, identifies exactly one CCC-Bicluster with at least two rows (maximal or not) (see Lemma 2). We also showed that each internal node corresponding to a *MaxNode* (see Definition 7) in the generalized suffix tree identifies exactly *one* maximal CCC-Bicluster and that each maximal CCC-Bicluster is identified by exactly *one MaxNode* (see Lemma 3 and Theorem 1). This also implies that a maximal CCC-Bicluster is identified by *one* expression pattern, which is common to all genes in the CCC-Bicluster within the contiguous columns in the bicluster. Moreover, all expression patterns identifying maximal CCC-Biclusters *always occur* in the discretized matrix and thus correspond to a node in the generalized suffix tree (see Figure 4).

When errors are allowed, *one e*-CCC-Bicluster ($e > 0$) can be identified (and usually is) by *several* nodes in the gen-

eralized suffix tree, constructed for the set of strings corresponding to the rows in the discretized matrix after alphabet transformation, and *one* node in the generalized suffix tree may be related with *multiple* *e*-CCC-Biclusters (maximal or not) (see Figure 7). Moreover, a maximal *e*-CCC-Bicluster can be defined by *several* expression patterns (see Figure 5 and Figure 6). Upon all this, a maximal *e*-CCC-Bicluster can be defined by an expression pattern *not occurring* in the expression matrix and thus not appearing in the generalized suffix tree (see Figure 6 and Figure 7).

Furthermore we cannot obtain *all* maximal *e*-CCC-Biclusters using the set of maximal CCC-Biclusters by: 1) extending them with genes by looking for their approximate patterns in the generalized suffix tree, or 2) extending them with *e* contiguous columns (see Figure 5 and Figure 8). It is also clear from Figure 8 that extending maximal CCC-Biclusters can in fact lead to the discovery of non maximal *e*-CCC-Biclusters. For the reasons stated above we cannot use the same searching strategy used to find maximal CCC-Biclusters when looking for maximal *e*-CCC-Biclusters ($e > 0$). We therefore need to explore the relation between finding *e*-CCC-Biclusters and the *Common Motifs Problem*, as explained below.

Finding e-CCC-Biclusters and the common motifs problem

There is an interesting relation between the problem of finding all maximal *e*-CCC-Biclusters, discussed in this work, and the well known problem of finding common motifs (patterns) in a set of sequences (strings). For the first problem, and to our knowledge, no efficient algorithm has been proposed to date. For the latter problem (*Common Motifs Problem*), several efficient algorithms based on string processing techniques have been proposed to date [25,26]. The *Common Motifs Problem* is as follows [26]:

Common Motifs Problem Given a set of N sequences S_i ($1 \le i \le N$) and two integers $e \ge 0$ and $2 \le q \le N$, where e is the number of errors allowed and q is the required quorum, find all models m that appear in at least q distinct sequences of S_i.

During the design of *e*-CCC-Biclustering, we used the ideas proposed in SPELLER [26], an algorithm to find *common motifs* in a set of N sequences using a generalized suffix tree T. The motifs searched by SPELLER correspond to *words*, over an alphabet Σ, which must occur with at most e mismatches in $2 \le q \le N$ distinct sequences. Since these words representing the motifs may not be present exactly in the sequences (see SPELLER for details), a motif is seen as an "external" object and called *model*. In order to be considered a *valid model*, a given model m of length $|m|$ has to verify the *quorum constraint*: m must belong to the e-neighborhood of a word w in at least q distinct sequences.

In order to solve the *Common Motifs Problem*, SPELLER builds a generalized suffix tree T for the set of sequences S_i and then, after some further preprocessing, uses this tree to "spell" the valid models. Valid models verify two properties [26]:

1. All the prefixes of a valid model are also valid models.

2. When $e = 0$, spelling a model leads to one node v in T such that $L(v) \ge q$, where $L(v)$ denotes the number of leaves in the subtree rooted at v.

When $e > 0$, spelling a model leads to a set of nodes v_1, ..., v_k in T for which $\sum_{j=1}^{k} L(v_j) \ge q$, where $L(v_j)$ denotes the number of leaves in the subtree rooted at v_j.

In these settings, and since the occurrences of a model are in fact nodes of the generalized suffix tree T, these occurrences are called *node-occurrences* [26]. The goal of SPELLER is thus to identify all valid models by extending them in the generalized suffix tree and to report them together with their set of node-occurrences. We present here an adaptation of the definition of node-occurrence used in SPELLER. In SPELLER, a node-occurrence is defined by a pair (v, v_{err}) and not by a triple (v, v_{err}, p), as in this work. For clarity, SPELLER was originally exemplified [26] in an uncompacted version of the generalized suffix tree, that is, a trie (although it was proposed to work with a generalized suffix tree). However, and as pointed out by the authors, when using a generalized suffix tree, as in our case, we need to know at any given step in the algorithm whether we are at a node or in an edge between nodes v and v'. We use p to provide this information, and redefine node-occurrence as follows:

Definition 11 (node-occurrence) *A node-occurrence of a model m is a triple (v, v_{err}, p), where v is a node in the generalized suffix tree T and v_{err} is the number of mismatches between m and the string-label of v computed using Hamming(m, string-label(v)). The integer $p \ge 0$ identifies a position/point in T such that:*

1. If $p = 0$: we are exactly at node v.

2. If $p > 0$: we are in $E(v)$, the edge between $father_v$ and v, in a point p between two symbols in label$(E(v))$ such that $1 \le p < |label(E(v))|$.

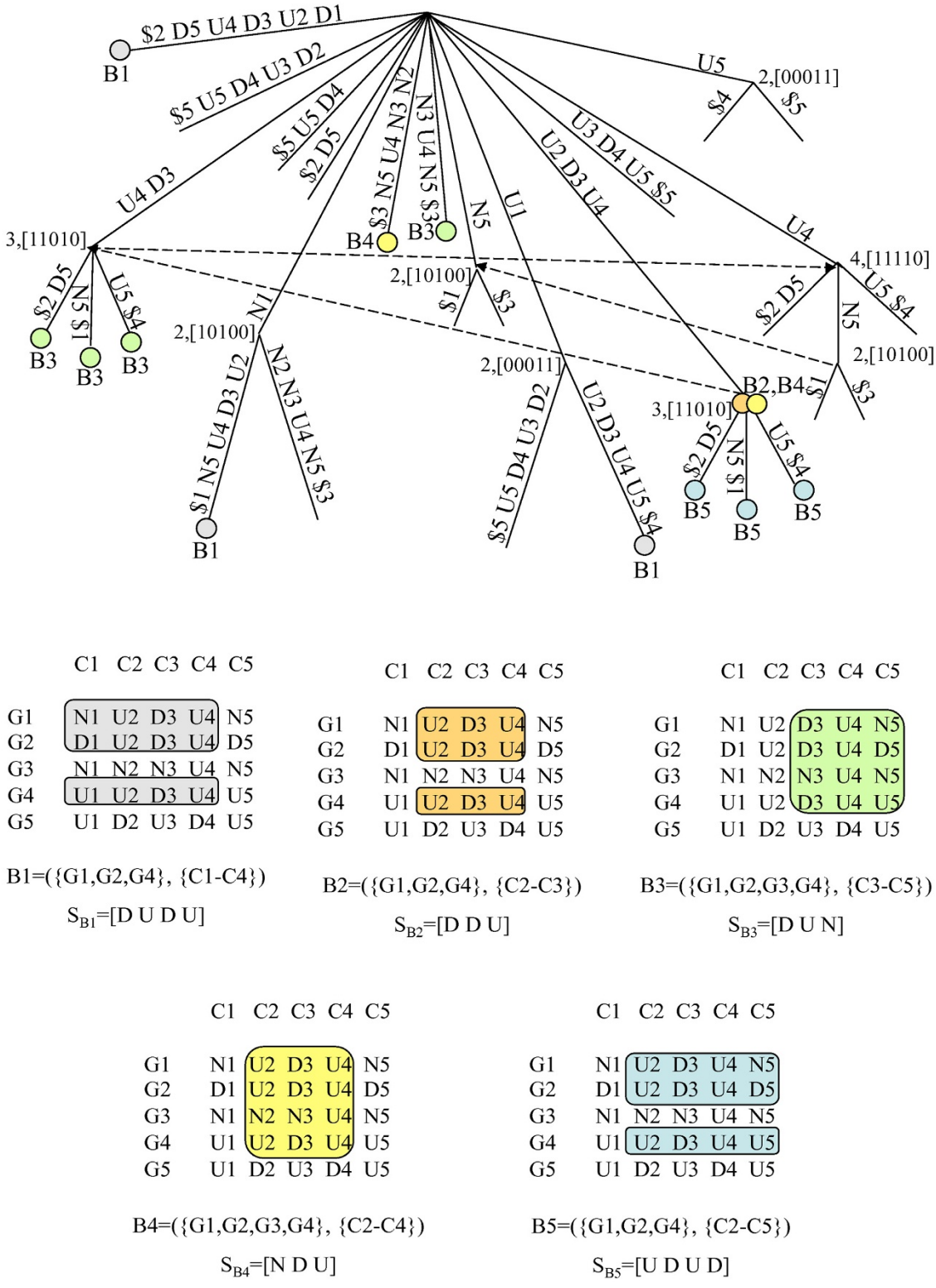


Figure 7 *(see legend on next page)*

Figure 7 *(see previous page)*
e-CCC-Biclusters (e > 0) and generalized suffix trees. This figure shows: **(Top)** Generalized suffix tree constructed for the transformed matrix in Figure 3 (the information stored in the nodes correspond to the number of leaves and row identifiers in their subtree and is used by e-CCC-Biclustering). The circles labeled with B1, B2, B3, B4 and B5 identify the nodes related with the five maximal 1-CCC-Biclusters discovered when e = 1 and $q_e = q_c = 3$, shown in Figure 6; **(Bottom)** Maximal 1-CCC-Biclusters B1 to B5 showed in the matrix as subsets of rows and columns. The strings S_{B1} = [D U D U], S_{B2} = [D D U], S_{B3} = [D U N], S_{B4} = [N D U] and S_{B5} = [U D U D] correspond to the expression patterns defining the maximal 1-CCC-Biclusters identified as B1 to B5, respectively. Note that e-CCC-Biclusters can now be identified (and generally are) by more than one node in the generalized suffix tree. This is the case of 1-CCC-Biclusters B1, B3, B4 and B5. In fact only B2 is identified by a single node in this example. Moreover, a node in the generalized suffix tree might be related with more than one maximal e-CCC-Bicluster. Look for example at the node identifying approximate patterns occurring in both 1-CCC-Biclusters B2 and B4.

Consider a model m, a symbol α in the alphabet Σ, a node v in T, its father $father_v$, the edge between $father_v$ and v, $E(v)$, the edge-label of $E(v)$, $label(E(v))$ and its edge-length, $|label(E(v))|$. The modified version of SPELLER described below is based on the following Lemmas (adapted from SPELLER):

Lemma 5 $(v, v_{err}, 0)$ *is a node-occurrence of a model* $m' = m\alpha$, *if, and only if:*

1. ***Match:***

   $(father_v, v_{err}, 0)$ *is a node-occurrence of* m *and* $label(E(v)) = \alpha$.

   The edge-label of $E(v)$ *has only one symbol and this symbol is* α.

   or

   $(v, v_{err}, |label(E(v))| - 1)$ *is a node-occurrence of* m *and* $label(E(v))\ [|label(E(v))|] = \alpha$.

   The last symbol in $label(E(v))$ *is* α.

2. ***Substitution:***

   $(father_v, v_{err} - 1, 0)$ *is a node-occurrence of* m *and* $label(E(v)) = \beta \neq \alpha$.

   The edge-label of $E(v)$ *has only one symbol and this symbol is not* α.

   or

   $(v, v_{err} - 1, |label(E(v))| - 1)$ *is a node-occurrence of* m *and* $label(E(v))\ [|label(E(v))|] = \beta \neq \alpha$.

   The last symbol in $label(E(v))$ *is not* α.

Lemma 6 $(v, v_{err}, 1)$ *is a node-occurrence of a model* $m' = m\alpha$, *if, and only if:*

1. ***Match:***

   $(father_v, v_{err}, 0)$ *is a node-occurrence of* m *and* $label(E(v))[1] = \alpha$.

2. ***Substitution:***

   $(father_v, v_{err} - 1, 0)$ *is a node-occurrence of* m *and* $label(E(v))[1] = \beta \neq \alpha$.

Lemma 7 (v, v_{err}, p), $2 \leq p < |label(E(v)|$ *is a node-occurrence of a model* $m' = m\alpha$, *if, and only if:*

1. ***Match:***

   $(v, v_{err}, p - 1)$ *is a node-occurrence of* m *and* $label(E(v)[p] = \alpha$.

2. ***Substitution:***

   $(v, v_{err} - 1, p - 1)$ *is a node-occurrence of* m *and* $label(E(v))\ [p] = \beta \neq \alpha$.

Consider now the discretized matrix A obtained from matrix A' using the alphabet Σ. We preprocess A using the same alphabet transformation used in CCC-Biclustering. Remember that we append the column number to each symbol in the matrix and consider a new alphabet $\Sigma' = \Sigma \times \{1, ..., |C|\}$ (see Figure 3). We will now show that SPELLER can be adapted to extract all right-maximal *e*-CCC-Biclusters from this transformed matrix A by building a generalized suffix tree for the set of $|R|$ strings S_i obtained from each row in A and use it to "spell" the valid models using the symbols in the new alphabet Σ'.

Given the set of $|R|$ strings S_i, the number of allowed errors $e \geq 0$ and the quorum constraint $2 \leq q \leq |R|$, the goal is now to find the set of all *right-maximal* valid models m, identifying expression patterns that are present in at least q distinct rows *starting and ending at the same columns*. Note that the valid models identified by the original SPELLER algorithm are already row-maximal. However they may be

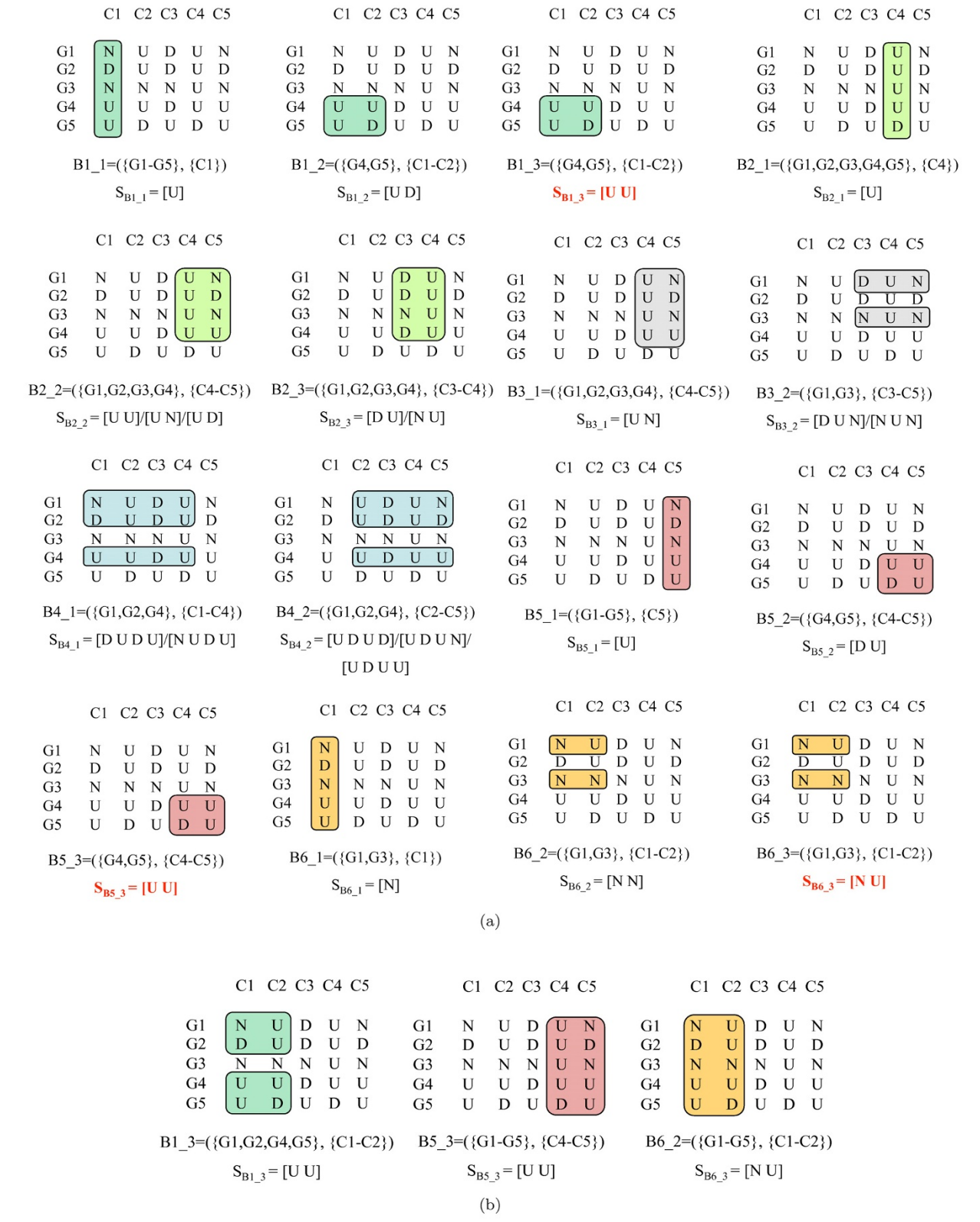


Figure 8 *(see legend on next page)*

Figure 8 *(see previous page)*
Maximal CCC-Biclusters and maximal e-CCC-Biclusters. This figure shows: **(Top)** 1-CCC-Biclusters obtained from the maximal CCC-Biclusters in Figure 2 by extending them with genes by looking for their approximate patterns in the generalized suffix tree (1-CCC-Biclusters B1_1, B2_1, B3_1, B5_1 and B6_1) or extending them with e = 1 contiguous columns at right (1-CCC-Biclusters B1_2, B1_3, B2_2, B4_2, B6_2 and B6_3) or at left (1-CCC-Biclusters B2_3, B3_2, B4_1, B5_2 and B5_3). Note that several of these 1-Biclusters can be defined by more than one expression pattern. This is the case of 1-CCC-Biclusters B2_1, B2_3, B3_2, B4_1 and B4_2, which in fact correspond to maximal 1-CCC-Biclusters (see Figure 5). Other 1-CCC-Biclusters are identified by a single expression pattern. This is the case of 1-CCC-Biclusters B1_1, B1_2, B2_1, B3_1, B5_1, B5_2, B6_1 and B6 2, and also correspond to maximal 1-CCC-Biclusters (see Figure 5). However, the 1-CCC-Biclusters B1_3, B5_3 and B6_3 do not correspond to maximal 1-CCC-Biclusters since they are not row-maximal. **(Bottom)** Maximal 1-CCC-Biclusters B1_3, B5_3 and B6_3 obtained not only by extending maximal CCC-Biclusters B1, B5 and B6 with one contiguous column to the right, left and right, respectively, but also by looking for the patterns in the 1-neighborhood of the patterns S_{B1_3} = [U U] (columns C1–C2), S_{B5_3} = [U U] (columns C4–C5) and S_{B6_3} = [N U] (columns C1–C2). Note however, that even if we replaced the non maximal 1-CCC-Biclusters B1_3, B5_3 and B6_3 (in the top) by the truly maximal 1-CCC-Biclusters (in the bottom) we could only find 16 of the 36 maximal 1-CCC-Biclusters with at least two rows shown in Figure 5 that can be found in the discretized matrix in Figure 1.

non right-maximal, non left-maximal, and start at different positions in the sequences. Under these settings, the set of node-occurrences of each valid model m and the model itself in our modified version of SPELLER identifies one row-maximal, right-maximal e-CCC-Bicluster with q rows and a maximum of $|C|$ contiguous columns. Furthermore, it is possible to find all right-maximal e-CCC-Biclusters by fixing the quorum constraint, used to specify the number of rows/genes necessary to identify a model as valid, to the value $q = 2$. In this context, and in order to be able to solve not only Problem 1 but also Problem 2, we adapted SPELLER to consider not only a *row constraint*, $2 \leq q_r \leq |R|$, but also an additional *column constraint*, $1 \leq q_c \leq |C|$.

Figure 7 shows the generalized suffix tree used by our modified version of SPELLER when it is applied to the discretized matrix after alphabet transformation in Figure 3. We can also see in this figure the five maximal 1-CCC-Biclusters B1, B2, B3, B4 and B5, already shown in Figure 6, identified by five valid models, when $e = 1$ and the values q_r and q_c, specifying the row and column constraints, respectively, are set to 3. The maximal 1-CCC-Biclusters B1 to B5 are defined, respectively, by the following valid models: m = [D1 U2 D3 U4 N5] (three node-occurrences labeled with B1); m = [D2 D3 U4] (three node-occurrences labeled with B2), m = [D3 U4 N5] (four node-occurrences labeled with B3), m = [N2 D3 U4] (four node-occurrences labeled with B4) and m = [U2 D3 U4 D5] (four node-occurrences labeled with B5). It is also possible to observe in this figure that, when $e > 0$, *a model can be valid without being right/left-maximal* and that *several valid models may identify the same e-CCC-Bicluster*. For example, m = [D1 U2 D3] is valid but it is not right-maximal, m = [D3 U4 D5] is also valid but it is not left-maximal, and finally the models m = [D1 U2 D3 U4 N5] and m = [N1 U2 D3 U4 D5] are both valid but identify the same 1-CCC-Bicluster B1. Figure 4 shows the generalized suffix tree used when $e = 0$, $q_r = 2$ and $q_c = 1$. Since no errors are allowed the generalized suffix tree is the same as the one used by CCC-Biclustering and the maximal 0-CCC-Biclusters identified correspond in fact to the maximal CCC-Biclusters in Figure 2.

In the next section we describe the details of the modified version of SPELLER that we used to identify all right-maximal e-CCC-Biclusters. However, and for clarity, we summarize here the main differences between the original version of SPELLER and the modified version (procedure `computeRightMaximalBiclusters` in the next section), which we use as the first step of the e-CCC-Biclustering algorithm. While reading the differences listed below have in mind that in order to be maximal, an e-CCC-Bicluster must be row-maximal, right-maximal and left-maximal. Moreover, all the approximate patterns identifying genes in an e-CCC-Bicluster must start and end at the same columns.

1. In SPELLER a node-occurrence is defined by a pair (v, v_{err}) since (for clarity) the algorithm was exemplified using a trie and not a generalized suffix tree, as explained above. As such we redefined the original concept of node-occurrence to use the triple (v, v_{err}, p) (see Definition 11), adapted the three original Lemmas in SPELLER to use the new definition of node-occurrence (see Lemma 5, Lemma 6 and Lemma 7), and rewrote SPELLER to use a generalized suffix tree.

2. In SPELLER a model can be valid without being right/left-maximal. As such all models satisfying the quorum constraint are stored for further reporting. This means that the valid models reported by SPELLER are only row-maximal. We only store valid models that cannot be extended to the right without loosing

genes, that is valid models which are both row-maximal are right-maximal. This implied modifying the original procedure `storeModel` in SPELLER in order to include the procedure `checkRightMaximality` (see procedure `spellModels` in the next section, for details).

3. In SPELLER the node-occurrences of a valid model can start in any position in the sequences. In our modified version of this algorithm all node-occurrences of a valid model must start in the same position (same column in the discretized matrix) in order to guarantee that they belong to an *e*-CCC-Bicluster. As such we modified the construction of the generalized suffix tree used in SPELLER in order to be constructed using the set of strings corresponding to the set of rows in the discretized matrix after alphabet transformation. We also modified all the procedures used in SPELLER for model extension. Note that it is not possible to modify SPELLER in order to check if a valid model that is right-maximal is also left-maximal. This is so since we can only guarantee that a model is/is not left-maximal once we have computed all valid models corresponding to right-maximal *e*-CCC-Biclusters. This justifies why we need to discard valid models which are not left-maximal in the next step of the algorithm and did not integrate this step in our modified version of SPELLER.

In this context, we also show in the next section that the proposed *e*-CCC-Biclustering algorithm will need *three steps* to identify all maximal *e*-CCC-Biclusters without repetitions: a first step to identify all right-maximal *e*-CCC-Biclusters (for this we use the modified version of SPELLER), a second step to discard all right-maximal *e*-CCC-Biclusters which are not left-maximal, and finally a third step to discard repetitions, that is maximal valid models identifying the same maximal *e*-CCC-Bicluster.

Note that the original SPELLER algorithm does not eliminate repetitions (different valid models with the same set of node-occurrences). Furthermore, we also cannot integrate the elimination of valid models corresponding to the same right-maximal *e*-CCC-Biclusters in our modified version of SPELLER since we need the set of all valid models corresponding to right-maximal *e*-CCC-Biclusters in order to discard valid models which are not left-maximal in the second step of *e*-CCC-Biclustering.

e-CCC-Biclustering: Finding and reporting all maximal e-CCC-Biclusters in polynomial time

This section presents *e*-CCC-Biclustering, a polynomial time biclustering algorithm for finding and reporting all maximal CCC-Biclusters with approximate patterns (*e*-CCC-Biclusters), and describes its main steps. Algorithm 1 is designed to solve Problem 2: identify and report all maximal *e*-CCC-Biclusters $B_k = A_{I_k J_k}$ such that I_k and J_k have at least q_r rows and q_c columns, respectively. The proposed algorithm is easily adapted to solve problem 1 (identify and report all maximal *e*-CCC-Biclusters $B_k = A_{I_k J_k}$ without quorum constraints) by fixing the values of q_r and q_c to the values two and one, respectively. The proposed algorithm is based on the following steps (described in detail below):

[Step 1] Computes all valid models corresponding to right-maximal *e*-CCC-Biclusters. Uses the discretized matrix *A* after alphabet transformation, the quorum constraints q_r and q_c, a generalized suffix tree and a modified version of SPELLER.

[Step 2] Deletes all valid models not corresponding to left-maximal *e*-CCC-Biclusters. Uses all valid models computed in Step 1 and a trie.

[Step 3] Deletes all valid models representing the same *e*-CCC-Biclusters. Uses all valid models corresponding to maximal *e*-CCC-Biclusters (both left and right) computed in Step 2 and a hash table. Note that this step is only needed when $e > 0$.

[Step 4] Reports all maximal *e*-CCC-Biclusters.

Algorithm 1: *e*-CCC-Biclustering

Input : A, Σ, e, q_r, q_c

Output: Maximal *e*-CCC-Biclusters.

1 $\{S_1, ..., S_{|R|}\} \leftarrow$ `alphabetTransformation`(A, Σ)

2 *modelsOcc* $\leftarrow \{\}$

3 `computeRightMaximalBiclusters`$(\Sigma, e, q_r, q_c, \{S_1, ..., S_{|R|}\}, modelsOcc)$

4 `deleteNonLeftMaximalBiclusters`$(modelsOcc)$

5 if $e > 0$ **then**

6 `deleteRepeatedBiclusters`$(modelsOcc)$

7 `reportMaximalBiclusters`$(modelsOcc)$

Detailed discussions can be found in additional file 2: **algorithmic_complexity_details**.

Computing valid models corresponding to right-maximal e*-CCC-Biclusters*

In step 1 of *e*-CCC-Biclustering we compute all valid models *m* together with their node-occurrences Occ_m corresponding to right-maximal *e*-CCC-Biclusters. The details are shown in the procedure `computeRightMaximalBiclusters` below, which corresponds to a modified version of SPELLER.

Procedure `computeRightMaximalBiclusters`

Input: Σ, *e*, q_r, q_c, $\{S_1, ..., S_{|R|}\}$, *modelsOcc*

/* The value of *modelsOcc* is updated. */

1 $T_{right} \leftarrow$ `constructGeneralizedSuffixTree(`$\{S_1, ..., S_{|R|}\}$`)`

2 `addNumberOfLeaves(`T_{right}`)` /* Adds $L(v)$ to each node v in T_{right}. */

3 if $e \neq 0$ **then**

4 `addColorArray(`T_{right}`)`

/* Adds $colors_v$ to every node v in T_{right}: $colors_v[i] = 1$, if there is a leaf in the subtree rooted at v that is a suffix of S_i; $colors_v[i] = 0$, otherwise. */

5 $m \leftarrow$ "" /* model m is a string $[m[1] ... m[length_m - 1]]$ */

6 $length_m \leftarrow 0$

7 $father_m \leftarrow$ "" /* $father_m$ is a string $[m[1] ... m[length_m - 1]]$ */

8 $numberOfGenesOcc_{father_m} \leftarrow 0$

9 $Occ_m \leftarrow \{\}$ /* List of node-occurrences (v, v_{err}, p) */

10 `addNodeOccurrence(`Occ_m, (root(T_{right}), 0, 0))

11 $Ext_m \leftarrow \{\}$ /* Ext_m is the set of possible symbols α to extend the model m. */

12 if $e = 0$ **then**

13 forall *edges* $E(v_i)$ *leaving from node* root(T_{right}) *to a node* v_i **do**

14 if $label(E(v_i))[1]$ *is not a string terminator* **then**

15 `addSymbol(`Ext_m, $label(E(v_i))[1]$)

16 else

17 forall *symbols in* Σ **do**

/* Σ' must be in lexicographic order. */

18 `addSymbol(`Ext_m, Σ [*i*])

19 $length_m \leftarrow 0$

20 `spellModels(`Σ, *e*, q_r, q_c, *modelsOcc*, T_{right}, *m*, $length_m$, Occ_m, Ext_m, $father_m$, $numberOfGenesOcc_{father_m}$)

In this procedure we use the transformed matrix *A* as input and store the results in the list *modelsOcc*, which stores triples with the following information (*m*, $genesOcc_m$, $numberOfGenesOcc_m$), where *m* is the model, $genesOcc_m$ is a bit vector containing the distinct genes in the node-occurrences of *m*, Occ_m, and $numberOfGenesOcc_m$ is the number of bits set to 1 in $genesOcc_m$ and, therefore, the number of genes where the model occurs. This information is computed using the procedure `spellModels` described below, which corresponds to a modified version of the procedure with the same name used in SPELLER).

Procedure `spellModels`

/* Called recursively. Stores right-maximal *e*-CCC-Biclusters in *modelsOcc*. */

Input : Σ, *e*, q_r, q_c, *modelsOcc*, T_{right}, *m*, $length_m$, Occ_m, Ext_m, $father_m$, $numberOfGenesOcc_{father_m}$

/* The value of *modelsOcc* is updated. */

1 `keepModel(`q_r, q_c, *modelsOcc*, T_{right}, *m*, $length_m$, Occ_m, $father_m$, $numberOfGenesOcc_{father_m}$

2 if $length_m \leq |C|$ **then**

/* $|C|$ is the length of the longest model */

3 forall *symbols* α *in* Ext_m **do**

4 if α *is not a string terminator* **then**

5 maxGenes ← 0/* Sum of $L(v)$ for all node-occurrences (v, v_{err}, p) in $Occ_{m\alpha}$ */

6 minGenes← ∞/* Minimum $L(v)$ in all node-occurrences (v, v_{err}, p) in $Occ_{m\alpha}$ */

7 $Colors_{m\alpha} \leftarrow \{\}$

8 **if** $e > 0$ **then**

9 $Colors_{m\alpha}[i] \leftarrow 0,\ 1 \leq i \leq |R|$

/* $colors_{m\alpha}[i] = 1$, if there is a node-occurrence of m in S_i; */

/* $colors_{m\alpha}[i] = 0$, otherwise */

10 $Ext_{m\alpha} \leftarrow \{\}$

11 $Occ_{m\alpha} \leftarrow \{\}$

12 **forall** *node-occurrences* (v, v_{err}, p) *in* Occ_m **do**

/* If p = 0 we are at node v. Otherwise, we are at edge $E(v)$ between nodes $father(v)$ and v at point $p > 0$. */

13 **if** $p = 0$ **then**

14 extendFromNodeWithoutErrors(Σ, e, T_{right}, (v, v_{err}, p), m, α, $Occ_{m\alpha}$, $Colors_{m\alpha}$, $Ext_{m\alpha}$, *maxGenes*, *minGenes*)

15 **if** $(v_{err} < e)$ **then**

16 extendFromNodeWithErrors(Σ, e, T_{right}, (v, v_{err}, p), m, α, $Occ_{m\alpha}$, $Colors_{m\alpha}$, $Ext_{m\alpha}$, *maxGenes*, *minGenes*)

17 **else**

18 extendFromEdgeWithoutErrors(T_{right}, Σ, e, (v, v_{err}, p), m, α, m, $Occ_{m\alpha}$, $Colors_{m\alpha}$, $Ext_{m\alpha}$, *maxGenes*, *minGenes*)

19 **if** $x_{err} < e$ **then**

20 extendFromEdgeWithErrors(Σ, e, T_{right}, (v, v_{err}, p), m, α, $Occ_{m\alpha}$, $Colors_{m\alpha}$, $Ext_{m\alpha}$, *maxGenes*, *minGenes*)

21 **if** modelHasQuorum(*maxGenes*, *minGenes*, $Colors_{m\alpha}$, q_r) **then**

22 spellModels(Σ, e, q_r, q_c, *modelsOcc*, T_{right}, $m\alpha$, $length_m + 1$, $Occ_{m\alpha}$, $Ext_{m\alpha}$, $father_{m\alpha}$, $numberOfGenesOcc_m$)

The recursive procedure spellModels (modified to extract valid models corresponding to right-maximal *e*-CCC-Biclusters) is now able to:

1. Use a generalized suffix tree T_{right} and define node-occurrences as triples (v, v_{err}, p), where p is used throughout the algorithm to find out whether we are at node v ($p = 0$) or in an edge $E(v)$ between nodes v and $father_v$ ($p > 0$).

2. Check if a valid model m corresponds to a right-maximal *e*-CCC-Bicluster. This is performed using the procedure checkRightMaximality inside the procedure keepModel. This procedure deletes from the list of stored models, *modelsOcc*, a valid model m when the result of its extension with a symbol α, $m\alpha$, is also a valid model and the set of node-occurrences of $m\alpha$, $Occ_{m\alpha}$, has as many genes as the set of node-occurrences of its father m, Occ_m. When this is the case, m no longer corresponds to a right-maximal *e*-CCC-Bicluster since its expression pattern can be extended to the right with the symbol α without losing genes.

3. Restrict the extensions of a given model m, Ext_m, to the level of the model in the generalized suffix tree (column of the last symbol in m). When we are extending a model m with a symbol α (eventually extracting a valid model $m\alpha$), the column number of the last symbol in m, $m\,[length_m]$, is $C(m\,[length_m])$, where $C(m\,[length_m]) \in \{1, ..., |C|\}$, and errors are still allowed, α can only be one of the symbols in the set $\Sigma'_{C(m[length_m])+1}$, where $\Sigma'_{C(m[length_m])+1}$ corresponds to the subset of elements in Σ' whose column is equal to $C(m\,[length_m])) + 1$. For example, if $\Sigma = \{D, N, U\}$ and the model m = [D1] is being extended, the possible symbols α with which m can be extended to $m\alpha$ must be in $\Sigma'_2 = \{D2.\ N\ 2,\ U\ 2\}$. In the same way, if m = [D2 U3], the possible symbols α with which m can be extended to $m\alpha$ are in $\Sigma'_4 = \{D4, N\ 4, U\ 4\}$.

The algorithmic details of the procedures and functions called in the recursive procedure spellModels are described in additional file 2: **algorithmic_complexity_details**.

Deleting valid models not corresponding to left-maximal e*-CCC-Biclusters*

In step 2 of *e*-CCC-Biclustering (details in procedure `deleteNonLeftMaximalBiclusters` below), we remove from the valid models stored in *modelsOcc* (identifying right-maximal *e*-CCC-Biclusters) those not corresponding to left-maximal *e*-CCC-Biclusters. These models are removed from *modelsOcc* by first building a trie with the reverse patterns of all (right-maximal) models *m* and storing the number of genes in $numberOfGenesOcc_m$ in its corresponding node in the trie. After this, it is sufficient to mark as "non left-maximal" any node in the trie that has at least one child with as many genes as itself. This is easily achieved by performing a depth-first search (*dfs*) of the trie and computing, for each node, the maximum value amongst the values of $numberOfGenesOcc_m$ stored in its children. The models whose corresponding node in the trie is marked as "non left-maximal" are then removed from *modelsOcc*.

Procedure `deleteNonLeftMaximalBiclusters`

Input: *modelsOcc*

`/* The value of` *modelsOcc* `is updated. */`

1 $T_{left} \leftarrow$ `createTrie ()`

`/* Array which will store references to nodes in` T_{left} `*/`

2 $R_{nodes} \leftarrow \{\}$

3 foreach *model and occurrences* (m, $genesOcc_m$, $numberOfGenesOcc_m$) *in modelsOcc* **do**

4 $m_r \leftarrow$ `ReverseModel(`m`)`

5 *nodeRepresentingModel* $\leftarrow$ `addReverseModelToTrie(`T_{left}, m_r`)`

`/* Each node in` T_{left} `stores two integers1) the number of genes in the model it represents,` $genes_v$ `(0 if it does not represent the end of a model); and 2) the maximum number of genes in the subtree rooted at`v, $maxGenes_{subtree_v}$ `(computed later). Both these values are initialized with 0. */`

6 `addNumberOfGenes(`*nodeRepresentingModel*,$numberOfGenesOcc_m$`)`

7 `addReferenceToNode(`R_{nodes}, *nodeRepresentingModel*`)`

8 forall *nodes v in* T_{left} **do**

`/* Performed using a depth-first search (dfs) */`

9 if $genes_v > 0$ **then**

`/* Node` v `represents a model and is potentially left-maximal. */`

10 Mark v as "left-maximal"

11 else

12 Mark v as "non left-maximal"

13 Compute the maximum number of genes in the subtree rooted at v

14 foreach *node v in* T_{left} **do**

`/* Performed using a depth-first search (dfs) */`

15 if $genes_v > 0$ *and* $genes_v = maxGenes_{subtree_v}$ **then**

16 Mark v as "non left-maximal"

17 $p_{modelsOcc} \leftarrow 0$

18 foreach *model and occurrences* (m, $genesOcc_m$, $numberOfGenesOcc_m$) *in modelsOcc* **do**

19 if $R_{nodes}[p_{modelsOcc}]$ *is marked as "non-left maximal"* **then**

20 `deleteModelAndOccurrences(`*modelsOcc*, m`)`

21 $p_{modelsOcc} \leftarrow p_{modelsOcc} + 1$

Deleting valid models representing the same e*-CCC-Biclusters*

When errors are allowed, different valid models may identify the same *e*-CCC-Bicluster. Step 3 of *e*-CCC-Biclustering, described in detail in procedure `deleteRepeatedBiclusters` below, uses a hash table to remove from *modelsOcc* all the valid models that, although maximal (left and right), identify repeated *e*-CCC-Biclusters. This is needed because all valid models *m* with the same first and last columns and the same set of genes represent the same maximal *e*-CCC-Bicluster.

Procedure `deleteRepeatedBiclusters`

Input: *modelsOcc*

`/* The value of` *modelsOcc* `is updated. */`

1 $H \leftarrow$ `createHashTable()`

2 foreach *model and occurrences (m, genesOcc*$_m$*, numberOfGenesOcc*$_m$*) in modelsOcc* **do**

3 *firstColumn*$_m$ = $C(m\ [1])$

4 *lastColumn*$_m$ = $C(m\ [length_m])$

5 *key* $\leftarrow$ `createKey`(*firstColumn, lastColumn, genesOcc*$_m$)

6 *value* $\leftarrow$ (*firstColumn, lastColumn, genesOcc*$_m$)

7 if `containsKey`(*H, key*) **then**

8 *value*$_{key}$ $\leftarrow$ `getValue`(*H, key*)

9 if *value* = *value*$_{key}$ **then**

/* *H* already has a value representing the same *e*-CCC-Bicluster */

10 `deleteModelAndOccurrences`(*modelsOcc, m*)

11 else

12 `insertKeyValue`(*key, value*)

13 else

14 insertKeyValue(*key, value*)

Reporting all maximal e*-CCC-Biclusters*

After the three main steps of *e*-CCC-Biclustering the list *modelsOcc* stores all valid models corresponding to maximal *e*-CCC-Biclusters satisfying the quorum constraints q_r and q_c. In this context, the reporting procedure `reportMaximalBiclusters`, described below, lists these *e*-CCC-Biclusters using the information stored in the model *m* (needed to identify the expression pattern and the columns in each *e*-CCC-Bicluster) and the bit vector *genesOcc* (needed to identify the genes in the *e*-CCC-Bicluster).

Procedure `reportMaximalBiclusters`

Input: *modelsOcc*

1 foreach *model and occurrences (m, genesOcc*$_m$*, numberOfGenesOcc*$_m$*) in modelsOcc* **do**

2 *firstColumn*$_m$ = $C(m\ [1])$

3 *lastColumn*$_m$ = $C(m\ [length_m])$

4 print(*m, firstColumn*$_m$*, lastColumn*$_m$*, genesOcc*$_m$)

***e*-CCC-Biclustering: Complexity analysis**

In this section we sketch an analysis of the complexity of *e*-CCC-Biclustering. For a detailed complexity analysis see additional file 2: **algorithmic_complexity_details**.

Given a discretized matrix *A* with $|R|$ rows and $|C|$ columns, the alphabet transformation performed using the procedure `alphabetTransformation` takes $O(|R||C|)$ time.

The complexity of computing all valid models corresponding to right-maximal *e*-CCC-Biclusters using procedure `computeRightMaximalBiclusters` takes $O(|R|^2|C|^{1+e}|\Sigma|^e)$ operations. The construction of T_{right} and the computation of $L(v)$ for all its nodes takes $O(|R||C|)$ time each, using Ukkonen's algorithm with appropriate data structures, and a *dfs*, respectively. The increase in the alphabet size from $|\Sigma|$ to $|C||\Sigma|$ due to the alphabet transformation does not affect the $O(|R||C|)$ construction and manipulation of the generalized suffix tree [9]. When $e > 0$, adding the color array to all nodes in T_{right} takes $O(|R|^2|C|)$ time. Initializing Ext_m takes $O(|C||\Sigma|)$ and `spellModels` is $O(|R|^2|C|^{1+e}|\Sigma|^e)$. The complexity of this step of the algorithm is bounded by the complexity of `spellModels` and is thus $O(|R|^2|C|^{1+e}|\Sigma|^e)$. The complexity of deleting from *modelsOcc* all valid models that are not left-maximal using procedure `deleteNonLeftMaximalBiclusters` is $O(|R||C|^{2+e}|\Sigma|^e)$. Since the number of models in *modelsOcc* is $O(|R||C|^{1+e}|\Sigma|^e)$ and the size of the models is $O(|C|)$, the trie T_{left} can be constructed and manipulated in $O(|R||C|^{2+e}|\Sigma|^e)$.

The complexity of deleting from *modelsOcc* all models representing the same *e*-CCC-Biclusters with procedure `deleteRepeatedBiclusters` takes $O(|R|^2|C|^{1+e}|\Sigma|^e)$. Since computing the hash key for each of the $O(|R||C|^{1+e}|\Sigma|^e)$ models in *modelsOcc* takes $O(|R|)$ time, the overall complexity of this step is $O(|R|^2|C|^{1+e}|\Sigma|^e)$.

Since the number of genes in *genesOcc*$_m$ is $O(|R|)$ and computing the first and last column of the valid model *m* takes constant time, reporting all maximal *e*-CCC-Biclusters using procedure `reportMaximalBiclusters` is $O(|R|^2|C|^{1+e}|\Sigma|^e)$.

Therefore, the asymptotic complexity of the proposed *e*-CCC-Biclustering algorithm is $O(\max(|R|^2|C|^{1+e}|\Sigma|^e, |R||C|^{2+e}|\Sigma|^e))$. However, in most cases of interest $|R| >> |C|$ and the complexity becomes $O(|R|^2|C|^{1+e}|\Sigma|^e)$. Moreover, when $e = 0$, CCC-Biclustering [9,22] can be used to obtain $O(|R||C|)$.

Extensions to handle missing values, anticorrelated and scaled expression patterns

In this section we present extensions to *e*-CCC-Biclustering able to handle missing values and discover anticorrelated (opposite patterns) and scaled (patterns with different expression rates) expression patterns. In the subsections below we consider the illustrative example in Figure 9, corresponding to a modified version of the example in Figure 1. We now assume that some expression values are missing.

Handling missing values

Since *e*-CCC-Biclustering cannot deal with missing values directly, genes with missing values have to be removed, or missing values have to be filled, as a preprocessing step. In this section we present extensions that enable direct processing of the expression matrix with missing values. Our goal is to consider all available time points and thus always include the expression pattern of a gene as input to the extended version of the algorithm. Nevertheless genes with more than a predefined percentage of missing values can still be discarded in a preprocessing step.

Dealing with missing values in *e*-CCC-Biclustering is straightforward and can be performed in two ways:

1. Considering missing values as valid errors.

2. "Jumping over" missing values.

In order to consider missing values as valid errors we modify *e*-CCC-Biclustering as follows:

- The initialization of Ext_m in procedure `computeRightMaximalBiclusters` must include the symbol used for missing value, when $e > 0$, and ignore all edges descending from the root starting with this symbol, when $e = 0$.

- The extension of a model m with a symbol α in `spellModels` must take into account the following: α can either be, or not be, the symbol used for missing value, depending on whether we are performing an *extension without errors* or performing an *extension with errors*, respectively.

  For details, see procedures `extendFromNodeWithoutErrors` and `extendFromEdgeWithoutErrors`, in case of extensions without errors, or procedures `extendFromNodeWithErrors` and `extendFromEdgeWithErrors`, in case of extensions with errors. These procedures are called in `spellModels` and described in additional file 2: **algorithmic_complexity_details**.

Consider the illustrative example in Figure 9, where some gene expression values are missing.

Figure 10 shows the generalized suffix tree T_{right} and the two maximal 1-CCC-Biclusters (B1 and B2) identified by two valid models when $e = 1$, $q_r = q_c = 3$ and missing values are considered as valid errors.

In order to "jump over" missing values we modify *e*-CCC-Biclustering as follows:

- After alphabet transformation, we construct the generalized suffix tree T_{right}, used in procedure `computeRightMaximalBiclusters`, using the set of strings without missing values $\{S_{1_1}, ..., S_{1_{r_1}}, ..., S_{i_1}, ..., S_{i_{r_i}}, ..., S_{|R|_1}, ..., S_{|R|_{r_{|R|}}}\}$, where r_i is the number of contiguous sets of symbols without missing values in row i. The set of substrings of each string S_i (gene i), $\{S_{i_1}, ..., S_{i_{r_i}}\}$, is inserted in T using the *same* terminator $\$i$.

  Consider, for example, the string corresponding to the expression pattern of gene G2 in the illustrative example in Figure 9. In this case, and in order to "jump over" the missing value in the time points C3 and C5, we insert in T_{right} two strings corresponding to each of

	C1	C2	C3	C4	C5
G1	_	0.73	-0.54	0.45	0.25
G2	-0.34	0.46	_	0.76	_
G3	_	_	_	0.44	-0.11
G4	0.70	_	-0.41	0.33	0.35
G5	0.70	_	0.70	-0.33	0.75

	C1	C2	C3	C4	C5
G1	_	U	D	U	N
G2	D	U	_	U	_
G3	_	_	_	U	N
G4	U	_	D	U	U
G5	U	_	U	D	U

	C1	C2	C3	C4	C5
G1	_1	U2	D3	U4	N5
G2	D1	U2	_3	U4	_5
G3	_1	_2	_3	U4	N5
G4	U1	_2	D3	U4	U5
G5	U1	_2	U3	D4	U5

Figure 9
Illustrative example with missing values. This figure shows: **(Left)** Original expression matrix, **(Middle)** Discretized matrix and **(Right)** Discretized matrix after alphabet transformation.

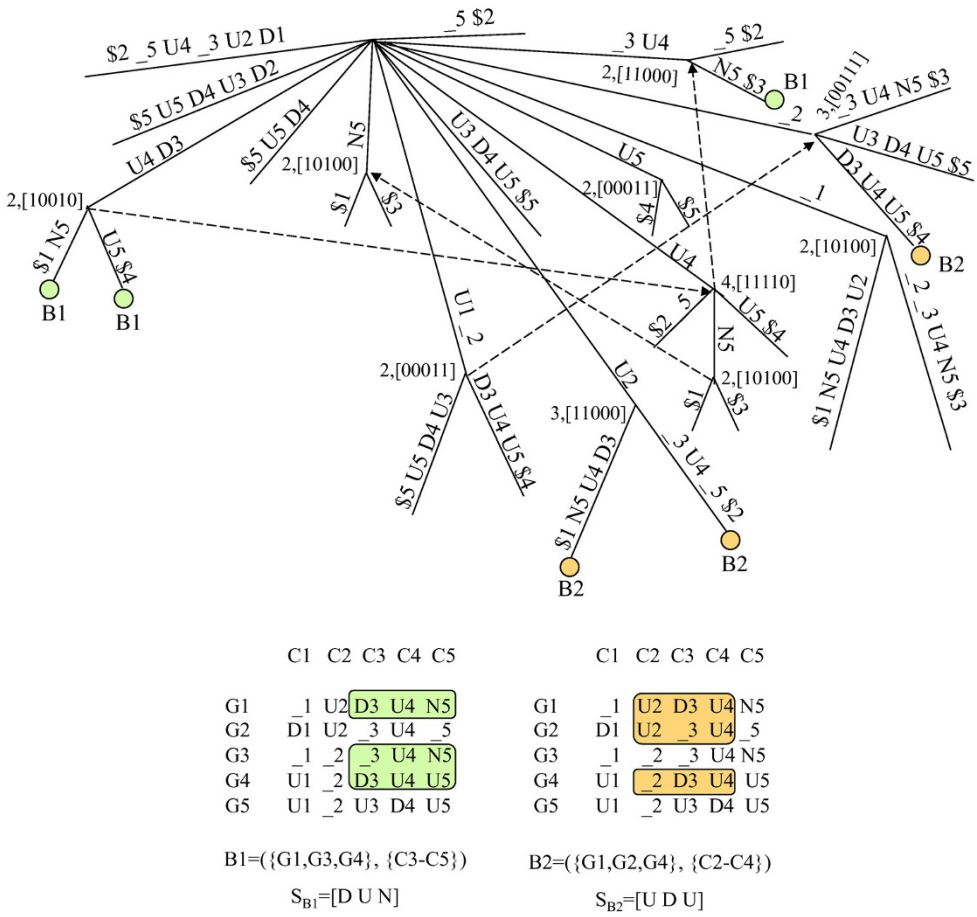


Figure 10

***e*-CCC-Biclusters extended to consider missing values as valid errors**. This figure shows: **(Top)** Generalized suffix tree used by e-CCC-Biclustering extended to consider missing values as valid errors when applied to the transformed matrix in Figure 9. The circles labeled with B1 and B2 identify the node-occurrences of the two maximal 1-CCC-Biclusters discovered when $e = 1$ and $q_e = q_c = 3$; **(Bottom)** Maximal 1-CCC-Biclusters corresponding, respectively, to the valid models m = [D3 U4 N5] (three node-occurrences labeled with B1) and m = [U2 D3 U4] (three node-occurrences labeled with B2).

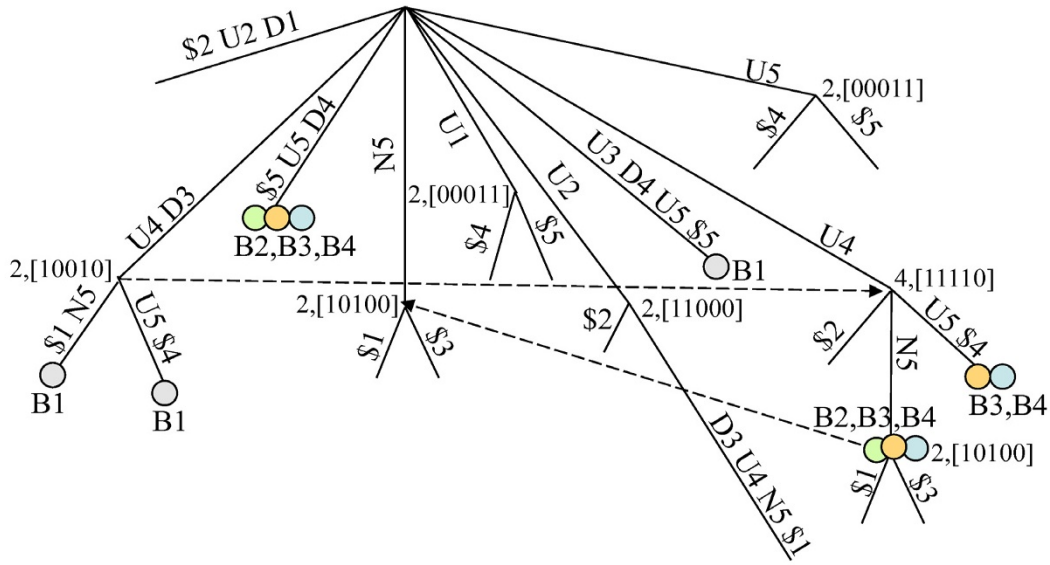
the two contiguous sets of symbols without missing values in the expression pattern of G2: S_{2_1} = [D1 U2 $2] and S_{2_2} = [U4 $2]. Note that the same terminator $2 is used for all the substrings of row i: S_{2_1} and S_{2_2}.

Figure 11 shows the generalized suffix tree T_{right} constructed for the matrix after alphabet transformation in Figure 9 together with the four maximal 1-CCC-Biclusters (B1, B2, B3 and B4), identified by four valid models, when $e = 1$, $q_r = 3$, $q_c = 2$ and the algorithm "jumps over" missing values.

The asymptotic complexity of both versions of this extended version of e-CCC-Biclustering remains $O(\max(|R|^2|C|^{1+e}|\Sigma|^e, |R||C|^{2+e}|\Sigma|^e))$. When $e = 0$, a modified version of CCC-Biclustering [27] can be used to achieve the linear time complexity $O(|R||C|)$, if repeated CCC-Biclusters are not filtered. In order to eliminate repetitions, the asymptotic complexity is now $O(|R|^2|C|)$.

Handling anticorrelated expression patterns

Given the importance of anticorrelation relationships in the study of transcription regulation using time series expression data we present here the extension of e-CCC-Biclustering to extract maximal e-CCC-Biclusters with sign-changes, that is maximal e-CCC-Biclusters allowing genes with opposite expression patterns. We first define formally the concepts of opposite expression pattern, e-



	C1	C2	C3	C4	C5
G1	_1	U2	D3	U4	N5
G2	D1	U2	_3	U4	_5
G3	_1	_2	_3	U4	N5
G4	U1	_2	D3	U4	U5
G5	U1	_2	U3	D4	U5

B1=({G1,G4,G5}, {C3-C4})

S_{B1}=[D D]

	C1	C2	C3	C4	C5
G1	_1	U2	D3	U4	N5
G2	D1	U2	_3	U4	_5
G3	_1	_2	_3	U4	N5
G4	U1	_2	D3	U4	U5
G5	U1	_2	U3	D4	U5

B2=({G1,G3,G5}, {C4-C5})

S_{B2}=[D N]

	C1	C2	C3	C4	C5
G1	_1	U2	D3	U4	N5
G2	D1	U2	_3	U4	_5
G3	_1	_2	_3	U4	N5
G4	U1	_2	D3	U4	U5
G5	U1	_2	U3	D4	U5

B3=({G1,G3,G4}, {C4-C5})

S_{B3}=[U D]

	C1	C2	C3	C4	C5
G1	_1	U2	D3	U4	N5
G2	D1	U2	_3	U4	_5
G3	_1	_2	_3	U4	N5
G4	U1	_2	D3	U4	U5
G5	U1	_2	U3	D4	U5

B4=({G1,G3,G4,G5}, {C4-C5})

S_{B4}=[U U]

Figure 11

e-CCC-Biclusters extended to "jump over" missing values. This figure shows: **(Top)** Generalized suffix tree used by e-CCC-Biclustering extended to "jump over" missing values when applied to the transformed matrix in Figure 9. The circles labeled with B1, B2, B3 and B4 identify the node-occurrences of the four maximal 1-CCC-Biclusters discovered when e = 1, q_e = 3 and q_c = 2; **(Bottom)** Maximal 1-CCC-Biclusters corresponding, respectively, to the valid models m = [D2 D3] (three node-occurrences labeled with B1), m = [D4 N5] (three node-occurrences labeled with B2), m = [U4 D5] (three node-occurrences labeled with B3) and m = [U4 U5] (three node-occurrences labeled with B4).

CCC-Bicluster with sign-changes, and maximal *e*-CCC-Bicluster with sign-changes:

Definition 12 (*e*-CCC-Bicluster with Sign-Changes) *An e-CCC-Bicluster with sign-changes A_{IJ} is an e-CCC-Bicluster where all the strings S_i that define the expression pattern of each of the genes in I are either in the e-Neighborhood of the expression pattern S that defines the e-CCC-Bicluster, or in the e-neighborhood of its opposite expression pattern S^{-1}: $S_i \in N(e; S)$ or $S_i \in N(e, S^{-1})$, $\forall i \in I$.*

Definition 13 (Maximal *e*-CCC-Bicluster with Sign-Changes) *An e-CCC-Bicluster with sign-changes A_{IJ} is maximal if it is row-maximal, left-maximal and right-maximal. This means that no more rows or **contiguous** columns can be added to I or J, respectively, maintaining the coherence property in Definition 12.*

In order to discover maximal *e*-CCC-Biclusters with sign-changes we modify *e*-CCC-Biclustering as follows:

- We construct the generalized suffix tree $T_{right'}$ used in procedure `computeRight MaximalBiclusters`, for the set of strings $S_i \in \{S_1, ..., S_{|R|}\}$ obtained after alphabet transformation and insert in T_{right} the set of opposite patterns of these strings $S_i^{-1} \in \{S_1^{-1}, ..., S_{|R|}^{-1}\}$. Since we use string terminators $\{\$1, ..., \$|R|\}$ for the expression patterns S_i and $\{\$(|R| + 1, ..., \$(2|R|)\}$ for their opposite patterns S_i^{-1} it is easy to compute the color arrays in T_{right} in $O(|R|)$ time and space. Notet hat we still use a color array with a maximum of $|R|$ bits and not $2|R|$ bits.

- When the extension to "jump over" missing values is considered, we construct T_{right} for the set of strings $\{S_{1_1}, ..., S_{1_{r_1}}, ..., S_{i_1}, ..., S_{i_{r_i}}, ..., S_{|R|_1}, ..., S_{|R|_{r_{|R|}}}\}$ and their opposite patterns $\{S_{1_1}^{-1}, ..., S_{1_{r_1}}^{-2}, ..., S_{i_1}^{-1}, ..., S_{i_{r_i}}^{-1}, ..., S_{|R|_1}^{-1}, ..., S_{|R|_{r_{|R|}}}^{-1}\}$.

Figure 12 shows the generalized suffix tree T_{right} and the three maximal 1-CCC-Biclusters (B1, B2 and B3), identified by three valid models, when $e = 1$, $q_r = 3$ and $q_c = 2$. In this example, the extension "jump over" missing values was used to handle missing values.

The asymptotic complexity of this extended version of *e*-CCC-Biclustering remains $O(\max(|R|^2|C|^{1+e}|\Sigma|^e, |R||C|^{2+e}|\Sigma|^e))$. Note however that, although the asymptotic complexity does not change the constant of proportionality is higher. When $e = 0$, a modified version of CCC-Biclustering [27] can again be used to achieve the linear time complexity $O(|R||C|)$, if repeated CCC-Biclusters are not filtered. However, removing repeated CCC-Biclusters takes $O(|R|^2|C|)$.

Handling scaled expression patterns

Since different genes can have different expression rates, we propose *e*-CCC-Biclustering with scaled expression patterns. These extensions allow the shifting of gene expression patterns up to *K* symbols up and down, in order to potentially find maximal *e*-CCC-Biclusters that would not be found due to different gene expression rates. The value of *K* is an integer between 1 and $(|\Sigma| - 1)$, where Σ is the set of symbols used to discretize the original expression matrix, in lexicographic order.

In the general case, and in order to shift the expression pattern of the genes *K* symbols up and down we consider a pair of *K* symbol alphabets: $\Sigma^{\uparrow}$ and $\Sigma^{\downarrow}$. These alphabets make it possible to shift all the symbols in $|\Sigma|$ the desired *K* symbols up and down. Assuming the three alphabets Σ, $\Sigma^{\uparrow}$ and $\Sigma^{\downarrow}$ are in lexicographic order and thus their symbols respect the ordering $\Sigma^{\downarrow}[1] < ... < \Sigma^{\downarrow}[K] < \Sigma[1] < ... < \Sigma[|\Sigma|] < \Sigma^{\uparrow}[1] < ... < \Sigma^{\uparrow}[K]$, the alphabet $\Sigma^{K_{shifts}} = \Sigma^{\downarrow} \cup \Sigma \cup \Sigma^{\uparrow} = \Sigma^{\downarrow} \cup \Sigma \cup \Sigma^{\uparrow}$ is also in lexicographic order.

For illustration purposes, consider $\Sigma = \{D, N, U\}$, $K = (|\Sigma| - 1) = 2$, and the illustrative example in Figure 9. In this case, and in order to shift the expression pattern of the genes $K = 2$ symbols up and down, we need to consider, for example, the $K = 2$ symbol alphabets $\Sigma^{\uparrow} = \{V, W\}$ and $\Sigma^{\downarrow} = \{B, C\}$. The three symbols in Σ are then shifted $K = 2$ symbols up and down using the following three pairs of alphabets: $\Sigma^{D^{\uparrow}} = \{N, U\}$ and $\Sigma^{D^{\downarrow}} = \{B, C\}$; $\Sigma^{N^{\uparrow}} = \{U, V\}$ and $\Sigma^{N^{\downarrow}} = \{C, D\}$; and $\Sigma^{U^{\uparrow}} = \{V, W\}$ and $\Sigma^{U^{\downarrow}} = \{D, N\}$. Thus, $\Sigma^{2_{shifts}} = \{B, C, D, N, U, V, W\}$, in this specific case.

We define *e*-CCC-Bicluster with scaled patterns and the notion of maximality as follows:

Definition 14 (*e*-CCC-Bicluster with Scaled Patterns) *An e-CCC-Bicluster with scaled patterns*

A_{IJ} is an e-CCC-Bicluster where all the strings S_i that define the expression pattern of each of the genes in I are either in the e-Neighborhood of the expression pattern S, that defines the e-CCC-Bicluster, or in the e-neighborhood of the patterns resulting from shifting its expression pattern S K symbols up,

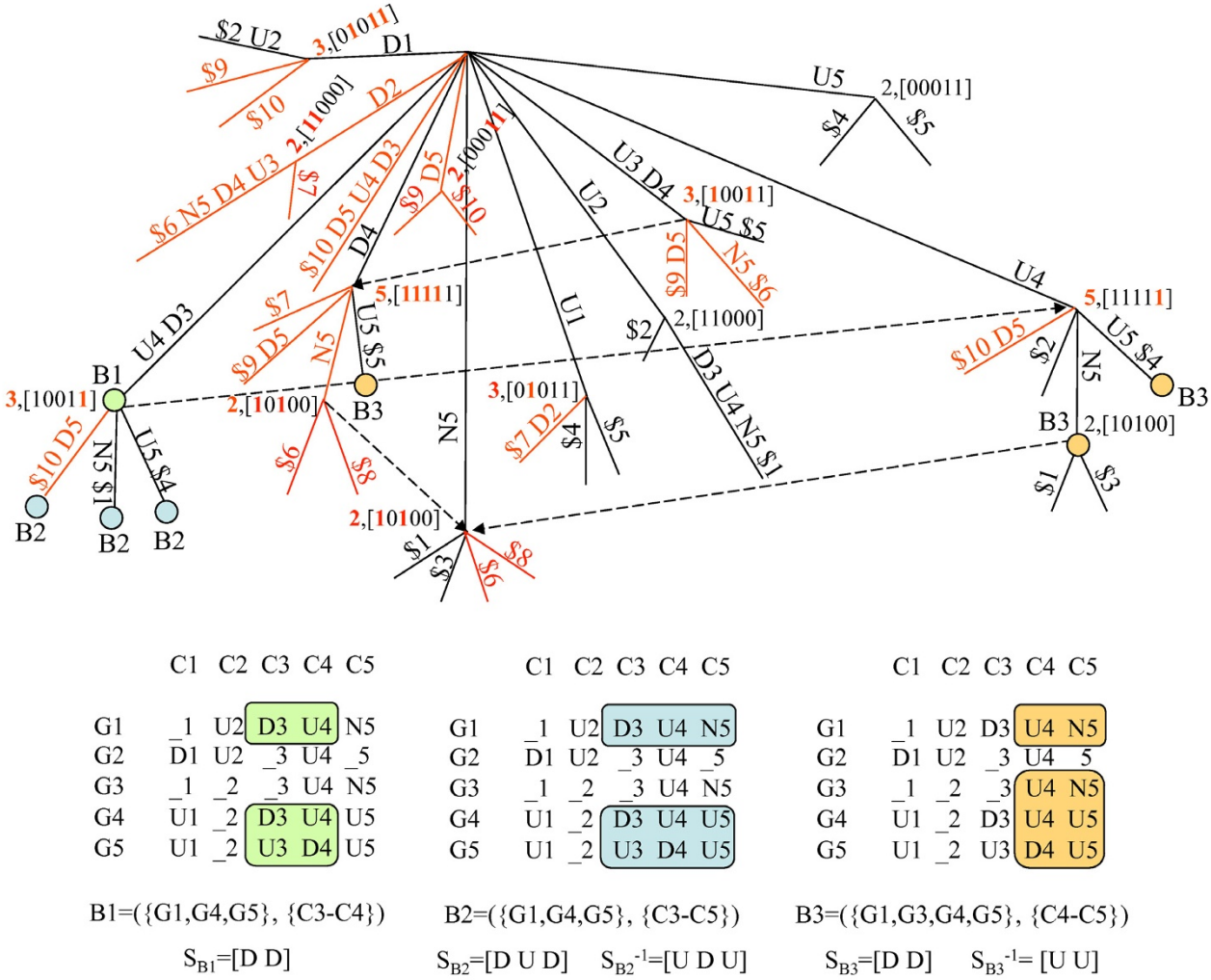


Figure 12

e-CCC-Biclusters extended to "jump over" missing values and allow anticorrelation. This figure shows: **(Top)** Generalized suffix tree used by e-CCC-Biclustering extended to "jump over" missing values and extract e-CCC-Biclusters with sign-changes when applied to the transformed matrix in Figure 9. The circles labeled with B1, B2 and B3 identify the node-occurrences of the three maximal 1-CCC-Biclusters discovered when e = 1, q_e = 3 and q_c = 2; **(Bottom)** Maximal 1-CCC-Biclusters corresponding, respectively, to the valid models *m* = [D3 D4] (B1), *m* = [D3 U4 D5] and m^{-1} = [U3 D4 U5] (B2), and *m* = [U4 U5] and m^{-1} = [D4 D5] (B3).

$S^{\uparrow} = \{S^{\uparrow_1}, ..., S^{\uparrow_K}\}$, *or K symbols down,* $S^{\uparrow} = \{S^{\downarrow_1}, ..., S^{\downarrow_K}\}$, *where K is an integer and* $K \in [1, ..., |\Sigma| - 1]$. *This means* $S_i \in <N(e, S) \vee S_i \in N(e, S^{\uparrow}) \vee S_i \in N(e, S^{\downarrow}), \forall i \in I$.

Definition 15 (Maximal *e*-CCC-Bicluster with Scaled Patterns) *An e-CCC-Bicluster with scaled patterns* A_{IJ} *is maximal if it is row-maximal, left-maximal and right-maximal. This means that no more rows can be added to the set of rows I and no* ***contiguous*** *columns can be added to the set of columns J while maintaining the coherence property in Definition 14.*

In order to discover *e*-CCC-Biclusters with scaled patterns we modify *e*-CCC-Biclustering as follows:

- We construct the generalized suffix tree T_{right}, used in procedure `computeRight MaximalBiclusters`, for the set of strings $S_i = \{S_1, ..., S_{|R|}\}$ and insert in T_{right} the patterns resulting from shifting the expression pattern S_i *K* symbols up and down.

  Since we use string terminators \$1, ..., \$|*R*| for the expression patterns S_i and \$(|*R*| + 1), ..., \$(|*R*| + 2 × *K*

× $|R|$) for shifted patterns it is easy to compute the colors arrays in T_{right} in $O(|R|)$ time and space.

• When the extension to "jump over" missing values is considered, we construct T_{right} for the set of strings $S_i = \{S_{1_1}, ..., S_{1_{r_1}}, ..., S_{|R|_1}, ..., S_{|R|_{r_{|R|}}}\}$ together with their corresponding set of shifted patterns K symbols up and down.

The asymptotic complexity of e-CCC-Biclustering with scaled patterns is $O(K|R|^2|C|^{1+e}|\Sigma|^e)$. When $e = 0$, a modified version of CCC-Biclustering [27] can be used to obtain $O(K|R||C|)$, or $O(K|R|^2|C|)$ if repetitions are discarded.

Alternative ways to compute approximate expression patterns

In this section we describe alternative ways to compute the errors allowed in the approximate patterns, which can reveal to be more suitable depending on the specific problem under study. The proposed e-CCC-Biclustering algorithm can be modified in order to cope with the three different kinds of errors described below: *restricted errors*, *alphabet range weighted errors*, and *pattern length adaptive errors*.

Restricted errors

The e-CCC-Biclustering algorithm allows *general errors*, that is, substitutions of the symbols A_{ij} in the e-CCC-Bicluster A_{IJ} by any symbol in the alphabet Σ'_j but A_{ij}. Considering approximate expression patterns having this kind of errors is specially relevant to minimize the negative effect of *measurement errors*, generally occurring during the microarray experiments, in the ability of the algorithm to identify relevant expression patterns. However, if we are specially interested in minimizing the also problematic effects of potential *discretization errors*, introduced due to poor choice of discretization thresholds or number of symbols, we can consider *restricted errors*, that is, substitutions of the symbols A_{ij} by the lexicographically closer symbols (*neighbors*) in $\Sigma'_j = \{\Sigma'_j[1], ..., \Sigma'_j[|\Sigma'_j|]\}$.

In general, when restricted errors are considered, the allowed substitutions for any symbol A_{ij} are in the set $\{\Sigma'_j[p-z], ..., \Sigma'_j[p-1], \Sigma'_j[p+1], ..., \Sigma'_j[p+z]\}$, where $p \in \{1, ..., |\Sigma'_j|\}$ is the position of A_{ij} in Σ'_j and z is a value in $\{1, ..., |\Sigma'_j| - 1\}$ that specifies the number of neighbors both to the left and to the right of $\Sigma'_j[p] = A_{ij}$ that are considered valid errors. Note that this set with the allowed symbols to substitute the symbol in A_{ij} has a maximum of $(2z)$ elements. Furthermore, the exact number of elements depends both on the number of considered neighbors, z, and on the position of A_{ij} in the alphabet Σ'_j, p. If $z = |\Sigma'_j| - 1$ then the errors are not restricted. For example, when general errors are allowed, $\Sigma = \{D, N, U\}$, and m = [U2 D3 U4 D5], D5 can be substituted by N5 and U5 in $\Sigma'_5 = \{D5, N5, U5\}$ leading to the 1-CCC-Bicluster B5 = ({G1, G2, G4},{C2–C5}) in Figure 7. However, if only restricted errors with $z = 1$ are allowed, D5 can only be substituted by $\{N5\}$ leading to 1-CCC-Bicluster B = ({G1, G2},{C2–C5}).

Alphabet range weighted errors

When the alphabet Σ used to discretize the data has many symbols, we can either restrict the errors allowed in the approximate patterns to a neighborhood around the symbol, or to consider alphabet range weighted errors. In the last case, we weight the errors according to the percentage of the total alphabet range they correspond to. For example, if Σ has 10 symbols, an error consisting of a substitution between symbols Σ[1] and Σ[3] should get a weight of 2/9 ~ 0.22 and not a weight of 1 (as happens to all errors in the definition of e-CCC-Bicluster). This means that in general an error from symbol Σ[i] to symbol Σ [j], considering that Σ is in lexicographic order and $i < j$, is weighted as $W_{e_{(\Sigma[i]-\Sigma[j])}} = (j-i)/(|\Sigma| - 1)$, where $W_{e_{(\Sigma[i]-\Sigma[j])}} \in [0, ..., 1]$. Since |Σ| - 1 is the maximum amplitude error, $W_{e_{(\Sigma[i]-\Sigma[j])}} = 1$, when i = 1 and j = |Σ|. Furthermore, $W_{e_{(\Sigma[i]-\Sigma[j])}} = 0$, each time $i = j$ and no error occurred. In these settings, a node-occurrence can be extended with errors if the *weighted sum of the errors* already found is less than e.

Pattern length adaptive errors

The definition of an e-CCC-Bicluster A_{IJ} states that the expression pattern S_i of each gene in I must be in the e-Neighborhood of an expression pattern S that defines the e-CCC-Bicluster. This implies that the maximum number of errors e is fixed, and, as such, it does not take into account the length of the expression pattern S_{B_k} of each individual e-CCC-Bicluster B_k. Since allowing e errors in an expression pattern of a few columns is not the same as allowing e errors in longer expression patterns, we pro-

pose the use of *pattern length adaptive errors* (the longer the pattern the more errors can be considered during the model extension process). In this context, we believe that a valuable extension to *e*-CCC-Biclustering is to allow not a fixed number of errors *e* per gene in I_{B_k}, but an adaptive number of errors, per gene and per *e*-CCC-Bicluster B_k, which is computed using a percentage of the number of columns in J_{B_k}. As such the goal is to modify the algorithm in order to find and report all maximal δ-CCC-Biclusters (defined below) instead of all maximal *e*-CCC-Biclusters using Lemma 8 and Lemma 9 (described below).

Definition 16 (δ-CCC-Bicluster) *A contiguous column coherent bicluster A_{IJ} with pattern length adaptive errors, δ-CCC-Bicluster, is a CCC-Bicluster where all the strings S_i that define the expression pattern of each of the genes in I are in the δ-Neighborhood of an expression pattern S that defines the δ-CCC-Bicluster: $S_i \in N(\delta, S)$, $\forall i \in I$. The value of δ is computed using a percentage of the number of columns in J: $\delta = \alpha|J|$, where $\alpha \in [0, ..., 1]$.*

Lemma 8 *A δ-CCC-Bicluster A_{IJ} is an e-CCC-Bicluster with $e = \delta = \alpha|J|$ errors per gene in I.*

Lemma 9 *The δ-Neighborhood of a string S, $N(\delta, S)$, where $\delta = \alpha|J|$, $\alpha \in [0, ..., 1]$ and $|J| \in [1, ..., |C|]$, contains $\sum_{j=0}^{\delta} C_j^{|S|}(|\Sigma|-1)^j \leq |S|^{\delta}|\Sigma|^{\delta} \leq |S|^{\dagger}|\Sigma|^{\dagger}$ elements, where $= \alpha|C|$.*

Scoring e-CCC-Biclusters using statistical significance and similarity measures

Since applying biclustering to real gene expression matrices can produce hundreds or even thousands of biclusters, an objective evaluation of the quality of the biclusters discovered is crucial. In fact, the inspection of biclustering results can be prohibitive without an efficient scoring approach which enables sorting and filtering the results according to a statistical scoring criterion. The statistical significance of the results can then be combined with measures of biological significance in order to produce a set of interesting and potentially useful biclusters, both from the statistical and biological point of view. For *e*-CCC-Biclusters, we propose the use of a scoring criterion, which combines two criteria:

1. Statistical significance of expression patterns.

2. Similarity measure between overlapping *e*-CCC-Biclusters.

In this work, we extend the concept of statistical significance of perfect expression patterns, proposed for CCC-Biclusters [9], in order to compute the statistical significance of approximate expression patterns. Based on this scoring criterion, the *p*-value of each *e*-CCC-Bicluster is computed and those not passing a Bonferroni corrected statistical significance test at a predefined level are discarded. Biclusters are then sorted by increasing order of their *p*-value and, when several of them overlap more than a predefined threshold, only the most significant are kept.

Note that although we use the stringent Bonferroni correction for multiple testing at the 1% level to guarantee that only the highly significant *e*-CCC-Biclusters are considered for further analysis, other less conservative statistical corrections can be used, thus considering more *e*-CCC-Biclusters as highly significant. Other significance levels can also be used. Moreover, even though we also use a stringent threshold in the overlapping filter (only *e*-CCC-Biclusters overlapping less than 25% are further analyzed), other overlapping thresholds can be used, thus allowing the analysis of a larger number of *e*-CCC-Biclusters, although potentially increasing the number of redundant biclusters.

In what follows we describe how to compute the statistical significance of an *e*-CCC-Bicluster using its expression pattern and describe the similarity score used to compare *e*-CCC-Biclusters.

Statistical significance

We proposed to measure the statistical significance of an *e*-CCC-Bicluster *B* of size $|I| \times |J|$, where *I* is the set of genes and *J* is the set of contiguous time-points, and expression pattern p_B, against the null hypothesis, H_0, that assumes that the expression values of genes evolve independently.

Under the null hypothesis, it is possible to compute, using reasonable simplifying assumptions, the probability of an *e*-CCC-Bicluster of the considered size and expression pattern occurring by chance in an expression matrix with $|R|$ genes and $|C|$ time points. The value of this probability is obtained by computing the *tail of the binomial distribution*, which gives the probability of an event with probability *p* occurring *k* or more times in *n* independent trials: $\sum_{j=k}^{n} C_j^n p^j (1-p)^{n-j}$.

The statistical significance of an *e*-CCC-Bicluster *B* is thus the value of *p*-value(*B*), which is computed by obtaining the probability of a random occurrence under H_0 of the expression patterns in the *e*-Neighborhood of the expression pattern p_B defining the *e*-CCC-Bicluster, $N(e, p_B)$, $k = |I| - 1$ times in $n = |R| - 1$ independent trials, where *I* is the

number of genes in B and $|R|$ is the total number of genes in the gene expression matrix. This is performed using the simplifying assumption that the probability of occurrence of a specific expression pattern in the e-Neighborhood of the pattern p_B, $N(e, p_B)$, is adequately modeled by a first order Markov Chain, with state transition probabilities obtained from the values in the corresponding columns in the matrix. In the general case,

$$P(N(e, p_B)) = \sum_{i=1}^{|N(e,p_B)|} P(N(e, p_B)[i])$$

where $|N(e, p_B)|$ and $N(e, p_B)[i]$ are, respectively, the number of patterns and the ith pattern in the e-Neighborhood of the pattern p_B. As an example, consider the computation of $P(N(e, p_B))$ when B is the e-CCC-Bicluster B1 = ({G1, G2, G4}, {C1 - C4}) in Figure 7 with p_B = [D1 U2 D3 U4]. Since, in this case, e-CCC-Biclustering was applied using e = 1, we have to compute $P(N(1, [D1\ U2\ D3\ U4]))$, which has, in this case, the following set of elements: {[D1 U2 D3 U4], [N1 U2 D3 U4], [U1 U2 D3 U4], [D1 D2 D3 U4], [D1 N2 D3 U4], [D1 U2 N3 U4], [D1 U2 U3 U4], [D1 U2 D3 D4], [D1 U2 D3 N4]}.

In this context, the value of $P(N(1, [D1\ U2\ D3\ U4]))$ is computed as follows: $P(N(1, [D1\ U2\ D3\ U4])) = P([D1\ U2\ D3\ U4]) + P([N1\ U2\ D3\ U4]) + P([U1\ U2\ D3\ U4]) + P([D1\ D2\ D3\ U4]) + P([D1\ N2\ D3\ U4]) + P([D1\ U2\ N3\ U4]) + P([D1\ U2\ U3\ U4]) + P([D1\ U2\ D3\ D4]) + P([D1\ U2\ D3\ N4])$.

By using a first order Markov Chain, $P([D1\ U2\ D3\ U4])$, for example, is computed as follows:

$$P([D1\ U2\ D3\ U4]) = P(D1)P(U2 \mid D1)P(D3 \mid U2)P(U4 \mid D3)$$

where $P(D1) = \frac{|D1|}{|G|}$, $P(U2 \mid D1) = \frac{P(D1U2)}{P(D1)} = \frac{|D1U2|}{|D1|}$, $P(D3 \mid U2) = \frac{P(U2D3)}{P(U2)} = \frac{|U2D3|}{|U2|}$ and $P(U4 \mid D3) = \frac{P(D3U4)}{P(D3)} = \frac{|D3U4|}{|D3|}$. The values $|D1|$, $|D1U2|$, $|U2|$, $|U2D3|$, $|D3|$, $|D3U4|$ correspond, respectively, to the number of occurrences of symbol D1, the number of transitions from D1 to U2, the number of occurrences of symbol U2, the number of transitions from D1 to U2, the number of occurrences of symbol D3 and the number of transitions from D3 to U4. The remainder conditional probabilities needed to compute $P(N(1, p_B))$ are computed in a similar way.

When *missing values are considered as valid errors*, $N(e, p_B)$ is computed using the alphabet $\Sigma' \cup mv'$, where mv is the symbol used for missing value and each element mv' is obtained by concatenating m and one number in the range $\{1, ..., |C|\}$, that is, $mv' = \{mv\} \times \{1, ..., |C|\}$.

When only *restricted errors* are allowed, $N(e, p_B)$ is not computed using all the symbols in Σ'. The allowed substitutions for each symbol in p_B are the z neighbors, both to the left and to the right of $\Sigma'[p]$ that are considered as valid errors, where p is the position of the symbol $p_B[k]$ in Σ'.

In the case of *e-CCC-Biclusters with sign-changes* we compute the statistical significance of B, using the p-value(B), by obtaining the probability of a random occurrence under H_0 of the expression patterns in the e-Neighborhoods of the patterns in the p_B and p_B^{-1}, $K = |I|$-1 times in $n = |R|$ -1 independent trials. We compute $P(N(e, p_B) \cup N(e, p_B^{-1}))$ as follows:

$$P(N(e, p_B) \bigcup N(e, p_B^{-1})) = P(N(e, p_B)) + P(N(e, p_B^{-1})),$$

where

$$P(N(e, p_B)) + P(N(e, p_B^{-1})) = \sum_{i=1}^{|N(e,p_B)|} P(N(e, p_B)[i]) + \sum_{i=1}^{|N(e,p_B^{-1})|} P(N(e, p_B^{-1})[i])$$

and $|N(e, p_B)|$, $|N(e, p_B^{-1})|$, $N(e, p_B)[i]$ and $N(e, p_B^{-1})[i]$ are, respectively, the number of patterns and the ith pattern in the e-Neighborhood of the pattern p_B and p_B^{-1}, respectively.

In the case of *e-CCC-Biclusters with scaled patterns* we compute the statistical significance of B, using the p-value(B), computed by obtaining the probability of a random occurrence under H_0 of the patterns in the e-Neighborhood of the pattern p_B, and its scaled patterns, $(p_B)^{\uparrow}$ and $(p_B)^{\downarrow}$, $k = |I|$ - 1 times in $n = |R|$ - 1 independent trials, where I is the number of genes in B and $|R|$ is the total number of genes in the matrix.

We compute $P(N(e, p_B) \cup N(e, (p_B)^{\uparrow})) \cup N(e, (p_B)^{\downarrow}))$ as follows:

$$P(N(e, p_B) \bigcup N(e, (p_B)^{\uparrow})) \bigcup N(e, (p_B)^{\downarrow})) =$$
$$P(N(e, p_B)) + \sum_{shift} P(N(e, p_B)^{\uparrow_{shift}})) + \sum_{shift} P(N(e, p_B)^{\downarrow_{shift}})),$$

where $shift \in \{1, ..., K\}$, and K is the value used in e-CCC-Biclustering with scaled patterns to shift the expression patterns K symbols up and down.

Similarity measure

In order to compute the similarity measure between two e-CCC-Biclusters, $B_1 = (I_1, J_1)$ and $B_2 = (I_2, J_2)$, we use the Jaccard Index. In this work, this score is used to measure

the overlap between two *e*-CCC-Biclusters both in terms of genes and conditions and is defined as follows:

$$J(B_1, B_2) = J((I_1, J_1), (I_2, J_2)) = \frac{|B_1 \cap B_2|}{|B_1 \cup B_2|} = \frac{|B_{11}|}{|B_{01}| + |B_{10}| - |B_{11}|},$$

where $B_{11} = \{(i, j): (i, j) \in B_1 \wedge (i, j) \in B_2\}$, $B_{10} = \{(i, j): (i, j) \in B_1 \wedge (i, j) \notin B_2\}$, and $B_{01} = \{(i, j): (i, j) \notin B_1 \wedge (i, j) \in B_2\}$, for the genes $i \in I_1 \cup I_2$ and the conditions $j \in J_1 \cup J_2$.

Similarly, the gene similarity and condition similarity can be computed, respectively, as follows:

$$J(I_1, I_2) = \frac{|I_1 \cap I_2|}{|I_1 \cup I_2|} \quad \text{and} \quad J(J_1, J_2) = \frac{|J_1 \cap J_2|}{|J_1 \cup J_2|}.$$

Note that, in practice, and since $|B_1| = |I_1| \times |J_1|$ and $|B_2| = |I_2| \times |J_2|$, the similarity score as defined above can be computed easily using the fact that $|B_1 \cap B_2| = |I_1 \cap I_2| \times |J_1 \cap J_2|$ and $|B_1 \cup B_2| = |B_1| + |B_2| - |B_1 \cap B_2|$.

Results and discussion

In this section we present and discuss the results obtained when applying the proposed *e*-CCC-Biclustering algorithm to real time series gene expression data. We also compare the performance of the proposed algorithm to that of CCC-Biclustering [9]. We first describe the dataset used to test the ability of the algorithm to find biologically relevant expression patterns in real data and to perform the comparison with CCC-Biclustering. This dataset describes the transcriptional responses of *Saccharomyces cerevisiae* to heat stress. We then show an application of *e*-CCC-Biclustering to the discovery of transcriptional regulatory modules. Finally, we present the comparison with CCC-Biclustering. All the results presented are based on the analysis of Gene Ontology annotations obtained using the GOToolbox database [28], together with information about transcriptional regulations available in the YEASTRACT database [29].

Dataset

We used a dataset from Gasch et al. [30], concerning the *Saccharomyces cerevisiae* response to heat shock. This dataset comprises seven different time points along the first hour of exposure to 37°C (0, 0, 0, 5, 15, 30 and 60 minutes) and corresponds to the experiment identified as "*heat shock 2*" in the original group of datasets described by the authors. Since the first three time points are replicates of the steady state, we computed an average of three replicates of time zero and used a dataset with five time points. From the original 6152 ORFS we removed those with missing values and the ones that no longer existed in SGD (Saccharomyces Genome Database). For the remaining 6142 genes we obtained the correspondence between ORFS and gene names using the YEASTRACT database [29]. Since both *e*-CCC-Biclustering and CCC-Biclustering work with a discretized matrix we have then discretized this dataset using the discretization technique proposed by Ji and Tan [20,31]. The discretized matrix A is obtained in two steps. In the first step, A' is transformed into an A'' = $|R| \times (|C| - 1)$ matrix of variations (see Equation 1). Once matrix A'' is generated, the final discretized matrix A, also with $|R|$ rows and $|C| - 1$ columns, is obtained in a second step by binning the values of the transformed matrix considering a threshold $t > 0$.

$$A''_{ij} = \begin{cases} \frac{A'_{i(j+1)} - A'_{ij}}{|A'_{ij}|} & \text{if } A'_{ij} \neq 0 \\ -1 & \text{if } A'_{ij} = 0 \text{ and } A'_{i(j+1)} < 0 \\ 1 & \text{if } A'_{ij} = 0 \text{ and } A'_{i(j+1)} > 0 \\ 0 & \text{if } A'_{ij} = 0 \text{ and } A'_{i(j+1)} = 0 \end{cases} \quad A_{ij} = \begin{cases} D & \text{if } A''_{ij} \leq -t \\ U & \text{if } A''_{ij} \geq t \\ N & \text{otherwise} \end{cases} \tag{1}$$

The expression matrix A' was standardized to zero mean and unit standard deviation, gene by gene, before the discretization process, and the discretization threshold t was set to the value of the standard deviation ($t = 1$). We refer to this preprocessed and discretized dataset as **DiscretizedHeatShock**.

Application of e-CCC-Biclustering to the identification of transcriptional regulatory modules

To assess the biological relevance of *e*-CCC-Biclusters in real data we applied *e*-CCC-Biclustering to the **DiscretizedHeatShock** dataset. We allowed only one error ($e = 1$) and considered only errors in the 1-neighborhood of the symbols in the alphabet $\Sigma = \{D, N, U\}$. Note that this corresponds to applying one of the *e*-CCC-Biclustering extensions we propose in this work (*e*-CCC-Biclustering with restricted errors). By restricting the errors to the 1-neighborhood of the symbols in the alphabet $\Sigma = \{D, N, U\}$, our goal is to avoid the impact of a poor choice of the discretization thresholds in the ability of the algorithm to find all genes with coherent expression patterns. As such, the errors $D <-> N$ and $N <-> U$ are allowed but the error $D <-> U$ is not permitted.

With this settings, 1-CCC-Biclustering found 170 maximal non-trivial 1-CCC-Biclusters. For these 170 1-CCC-Biclusters we computed the *p*-value using the method described in the previous section. Only 47 1-CCC-Biclusters were considered as statistically significant, at the 1% level, after applying the Bonferroni correction for multiple testing. All the 1-CCC-Biclusters not passing this statistical test were discarded. The remainder 47 were then sorted in ascending order of the statistical *p*-value previously computed. See additional file 3: **1_ccc_biclusters** for a summary of these 47 *e*-CCC-Biclusters.

In order to avoid the analysis of highly overlapping 1-CCC-Biclusters, we computed the similarities between the sorted 1-CCC-Biclusters using the Jaccard similarity score and filtered the 1-CCC-Biclusters with similarity above 25%. The filtering process removed 35 of the 47 1-CCC-Biclusters originally selected. Figure 13 shows the expression patterns of the 12 1-CCC-Biclusters that remain.

These 12 highly significant and non-redundant 1-CCC-Biclusters were then analyzed using the Gene Ontology annotations using the GOToolbox database [28], together with information about transcriptional regulations available in the YEASTRACT database [29].

Figure 14 shows a summary of these top 12 1-CCC-Biclusters (expression patterns, number of genes and contiguous time points) together with information about functional enrichment relatively to terms in the Gene Ontology. To perform the analysis for functional enrichment, we considered only the "*Biological Process*" *ontology* and *terms above level 2*. We used the *p*-values obtained using the hypergeometric distribution to assess the over-representation of a specific GO term. In order to consider an *e*-CCC-Bicluster to be *highly significant*, we require its genes to show highly significant enrichment in one or more of the "Biological Process" ontology terms by having a Bonferroni corrected *p*-value below 0.01. An *e*-CCC-Bicluster is considered as *significant* if at least one of the GO terms analyzed is significantly enriched by having a (Bonferroni corrected) *p*-value in the interval [0.01, 0.05[. Note that, although we only consider as functionally enriched the terms with Bonferroni corrected *p*-values below 0.01 (for high statistical significance), or below 0.05 (for statistical significance), the *p*-values presented in the text are without correction, as it is common practice in the literature.

It is worth noting that all the 1-CCC-Biclusters analyzed have in general a large number of GO terms enriched (after Bonferroni correction), and all of them have at least one term whose *p*-value is highly significant (see Figure 14, for details). This means *all* the 1-CCC-Biclusters identified are biologically relevant as reported by functional enrichment analysis performed using the Gene Ontology.

Figure 15 and Figure 16 show a detailed analysis of the Gene Ontology annotations together with information about transcriptional regulations available in the YEASTRACT database, for the 1-CCC-Biclusters with transcriptional up-regulation patterns and 1-CCC-Biclusters with transcriptional down-regulation patterns, respectively. When the 1-CCC-Bicluster has more than 10 terms enriched or its genes are co-regulated by more than 10 transcription factors (TFs), only the 10 terms with lower *p*-value or the 10 transcription factors regulating the higher percentage of the genes in the 1-CCC-Bicluster are listed. The GO terms marked with * only passed the statistical test at the 5% level.

Comparison with CCC-Biclustering: perfect versus approximate expression patterns

To assess the biological relevance of *e*-CCC-Biclusters in real data, and test our thesis regarding the potential superiority of this approach relatively to finding CCC-Biclusters with perfect expression patterns, we compared the results of *e*-CCC-Biclustering to those of CCC-Biclustering in the **DiscretizedHeatShock** dataset, as recently published by Madeira et al. [9].

In order to perform this comparison we reproduced the results in [9] using a prototype implementation of CCC-Biclustering coded in Java and made available by the authors in http://www.inesc-id.pt/kdbio/software/ccc-biclustering. We have also reproduced the biological analysis of CCC-Biclustering results since the data in the two databases (GoToolbox and YEASTRACT) used by the authors for this purpose was updated since the results in [9] were published.

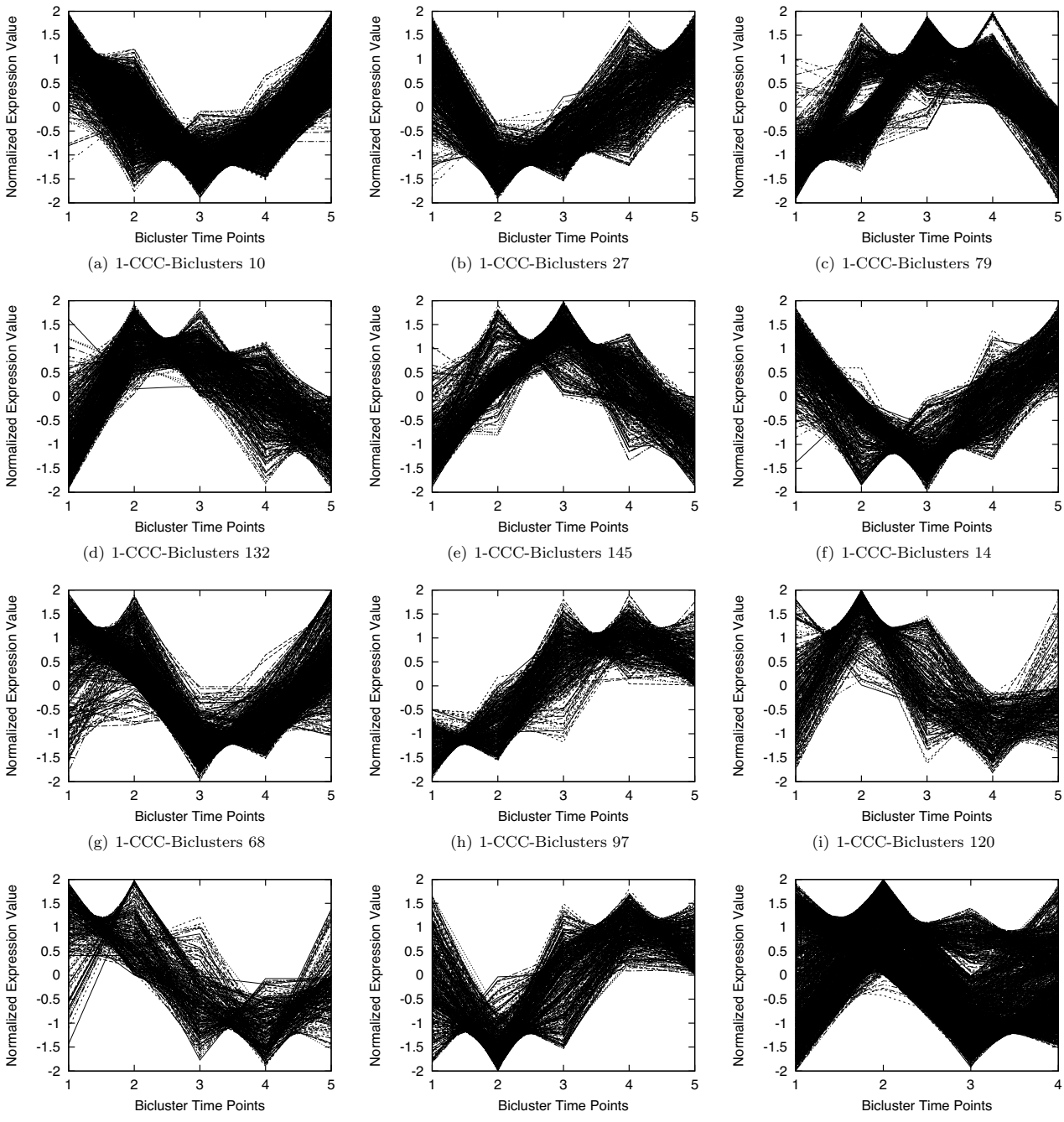
Our intuition, when performing this comparison, is that allowing a small number of errors, per gene, in the perfect expression patterns identifying the CCC-Biclusters (*0*-CCC-Biclusters) discovered by CCC-Biclustering should improve the biological significance of the biclusters by considering genes with approximate expression patterns and thus minimizing the effect of possible discretization errors.

Note that, in the specific case of allowing 1 error in the pattern of a CCC-Bicluster one of the following three situations can happen: (1) the 1-CCC-Bicluster is equal to the CCC-Bicluster; (2) one or more genes, excluded from the CCC-Bicluster due to a single error are added to the 1-CCC-Bicluster; (3) the pattern of the *0*-CCC-Bicluster is extended (by adding one contiguous column at its beginning/end) leading to a 1-CCC-Bicluster with at least as many genes as the CCC-Bicluster and one additional contiguous column.

In this context, we believe the improvement in the biological significance of the results obtained by *e*-CCC-Biclustering should be two-fold:

1. The functional enrichment of the *e*-CCC-Biclusters should improve not only regarding the *p*-values of the GO terms enriched but also in terms of the number of GO terms enriched.

2. The number of genes regulated by relevant transcription factors in 1-CCC-Biclusters (TFs) should be



(a) 1-CCC-Biclusters 10 (b) 1-CCC-Biclusters 27 (c) 1-CCC-Biclusters 79

(d) 1-CCC-Biclusters 132 (e) 1-CCC-Biclusters 145 (f) 1-CCC-Biclusters 14

(g) 1-CCC-Biclusters 68 (h) 1-CCC-Biclusters 97 (i) 1-CCC-Biclusters 120

(j) 1-CCC-Biclusters 63 (k) 1-CCC-Biclusters 39 (l) 1-CCC-Biclusters 122

Figure 13

Expression patterns of the 1-CCC-Biclusters surviving the overlapping filter. This figure shows two types of expression patterns: transcriptional up-regulation (1-CCC-Biclusters 79, 132, 145, 97, 120 and 39) and transcriptional down-regulation patterns (1-CCC-Biclusters 10, 27, 14, 68, 63 and 122).

#	ID	*Sorting p-value*	*Variation Pattern*	*#Time Points (first-last)*	*#Genes*	*# (corrected) GO p-values < 0.01*	*# (corrected) GO p-values 0.01 ≤< 0.05*
1	10	0.00E-00	DDNU	5 (1-5)	1079	58	16
2	27	0.00E-00	DNUU	5 (1-5)	597	22	13
7	79	0.00E-00	NNND	5 (1-5)	849	40	16
9	132	0.00E-00	UNDD	5 (1-5)	539	10	7
13	145	2.81E-41	UUDD	5 (1-5)	511	8	5
16	14	1.36E-37	DDUU	5 (1-5)	521	19	12
19	68	1.65E-33	NDNU	5 (1-5)	800	40	6
28	97	4.15E-15	NUUN	5 (1-5)	385	5	**0**
34	120	5.12E-10	UDDN	5 (1-5)	307	6	5
35	63	1.99E-09	NDDN	5 (1-5)	292	13	1
36	39	2.88E-09	DUUN	5 (1-5)	430	1	5
43	122	9.16E-07	UDN	4 (1-4)	1462	33	15

Figure 14

Summary of the 1-CCC-Biclusters surviving the overlapping filter. This table shows summary information about the 12 1-CCC-Biclusters surviving the overlapping filter. We show the statistical *p*-value of their expression patterns, the patterns themselves, the number of contiguous time points and the number of genes in each 1-CCC-Bicluster. Together with this information we present a summary of the results obtained when analyzing these 12 1-CCC-Biclusters using the Gene Ontology annotations restricted to those in the "Biological Process" ontology and above Level 2. We show the number of GO terms with highly significant and significant *p*-values, respectively, after Bonferroni correction for multiple testing. All 1-CCC-Biclusters are functionally enriched, having at least one term (several in general) whose *p*-value is highly statistical significant, after Bonferroni correction.

higher than the number of genes regulated by the same TFs in the corresponding CCC-Biclusters.

The validation of the two points above will, in our opinion, demonstrate that *e*-CCC-Biclustering is not only able to recover genes with approximate expression patterns, that are potentially lost when only perfect expression patterns are considered, but also that the recovered genes are, in fact, biologically relevant to the problem under study.

CCC-Biclustering discovered 167 maximal non-trivial CCC-Biclusters, which were then sorted in ascending order according to a statistical *p*-value similar to that we proposed here for *e*-CCC-Biclusters. From these only 25 CCC-Biclusters were considered as highly significant at the 1% level after applying the Bonferroni correction for multiple testing. In order to avoid the analysis of highly overlapping CCC-Biclusters, we have also computed the similarities between the sorted CCC-Biclusters using the Jaccard similarity score and filtered the CCC-Biclusters with similarity greater than 25%. The filtering process removed 9 of the 25 CCC-Biclusters originally selected. See additional file 4: **ccc_biclusters** for a summary of these 25 CCC-Biclusters. See also additional file 5: **1_ccc_biclusters_vs_ccc_biclusters** for a detailed comparison between the 47 highly significant 1-CCC-Biclusters discovered by 1-CCC-Biclustering restricted to errors in the 1-neighborhood of the symbols in the alphabet $\Sigma = \{D, N, U\}$ and the 16 highly significant CCC-Biclusters found by CCC-Biclustering and analyzed by Madeira et al. [9]. It is clear from this table that most of the 47 1-CCC-Biclusters discovered by the 1-CCC-Biclustering algorithm are highly overlapping with one or more of the top 16 CCC-Biclusters identified by the CCC-Biclustering algorithm. Figure 17 shows a summary of the remaining 16 CCC-Biclusters analyzed according to the Gene Ontology (GO) annotations obtained using the GoToolBox [28], together with information about transcriptional regulations available in the YEASTRACT database [29], as performed above for 1-CCC-Biclustering results. See additional file 6: **ccc_biclusters_biological_validation** for a detailed analysis of the GO terms enriched and transcriptional regulations of these top 16 CCC-Biclusters.

Note that, unlike what happened with the top 12 1-CCC-Biclusters discovered by 1-CCC-Biclustering (Figure 14), the top 16 CCC-Biclusters discovered by CCC-Biclustering (Figure 17) have in general a small number of GO terms enriched (after Bonferroni correction), and several of them are not functionally enriched (after Bonferroni cor-

ID	*#Genes*	*TFs (Top 10)*	*%*	*GO Terms Enriched (Top 10)*	*DF*	*p-value (level)*
79	849	Yap1p	28.8	carbohydrate metabolic process	9.60	6.86E-13 (4)
		Met4p	24.3	generation of precursor metabolites and energy	8.11	2.59E-12 (3)
		Sok2p	22.3	response to stress	15.23	5.22E-12 (3)
		Msn2p	17.3	cellular carbohydrate metabolic process	8.61	2.42E-11 (5)
		Atf1p	17.1	catabolic process	13.58	1.20E-10 (3)
		Hsf1p	16.7	cellular catabolic process	13.08	3.82E-10 (4)
		Rpn4p	14.8	energy derivation by oxidation of organic compounds	6.46	3.94E-10 (4)
		Msn4p	14.5	phosphorylation	6.13	3.45E-08 (6)
		Arr1p	13.5	actin filament-based process	4.97	4.51E-08 (6)
		Ste12p	10.6	actin cytoskeleton organization and biogenesis	4.80	5.71E-08 (7)
132	539	Sok2p	23.3	regulation of biological process	21.18	1.32E-07 (3)
		Yap1p	20.9	regulation of cellular process	20.59	1.84E-07 (4,3)
		Met4p	17.4	regulation of transcription from RNA polymerase II promoter	8.82	8.38E-07 (9,8)
		Aft1p	16.0	cell communication	9.12	1.59E-06 (3)
		Hsf1p	13.8	transcription from RNA polymerase II promoter	11.47	1.83E-06 (8,7)
		Arr1p	13.4	transcription	0.1618	2.19E-06 (5)
		Ste12p	11.9	reg. of nucleobase, nucleoside, nucleotide and nucleic acid met. proc.	13.53	2.20E-06 (6,5)
		Msn2p	11.6	regulation of metabolic process	15.29	4.61E-06 (4,3)
		Rpn4p	10.8	cell cycle process	13.24	4.70E-06 (4,3)
		Tec1p	10.4	regulation of transcription	12.06	5.35E-06 (7,6)
145	511	Yap1p	24.2	regulation of metabolic process	16.67	2.77E-07 (4,3)
		Sok2p	24.0	cell communication	9.75	3.86E-07 (3)
		Met4p	21.9	regulation of biological process	20.75	8.92E-07 (3)
		Aft1p	20.5	response to stress	15.09	1.09E-06 (3)
		Hsf1p	17.7	carbohydrate metabolic process	9.12	1.93E-06 (4)
		Msn2p	15.0	response to chemical stimulus	12.58	1.97E-06 (3)
		Rpn4p	14.2	post-translational protein modification	12.58	2.87E-06 (7)
		Arr1p	14.0	regulation of cellular metabolic process	14.78	6.83E-06 (5,4)
		Msn4p	12.6	regulation of cellular process	19.18	1.06E-05 (4,3)*
		Ste12p	11.8	cell cycle arrest in response to pheromone	1.57	1.34E-05 (6-9)*
97	385	Yap1p	30.9	organic acid metabolic process	13.01	4.16E-08 (4)
		Met4p	22.1	carboxylic acid metabolic process	13.01	4.16E-08 (5)
		Sok2p	20.0	nitrogen compound metabolic process	11.15	8.81E-08 (3)
		Rpn4p	17.1	cellular biosynthetic process	13.75	2.03E-07 (4)
		Arr1p	15.3	amine metabolic process	9.67	1.56E-06 (4)
		Aft1p	14.5			
		Hsf1p	14.0			
		Ste12p	11.9			
		Gcn4p	10.9			
		Leu3p	10.6			
120	307	Yap1p	25.7	nucleobase, nucleoside, nucleotide and nucleic acid metabolic process	38.78	6.29E-07 (4)
		Sok2p	16.3	regulation of cellular process	22.96	1.40E-06 (4,3)
		Met4p	16.0	cell cycle process	16.33	1.54E-06 (4,3)
		Aft1p	15.3	regulation of biological process	22.96	3.15E-06 (3)
		Arr1p	12.4	cell cycle phase	13.27	1.28E-05 (5,4)
		Ste12p	12.1	transcription	17.86	1.81E-05 (5)*
		Phd1p	10.1	RNA metabolic process	27.55	2.05E-05 (5)*
		Hsf1p	9.8	RNA biosynthetic process	16.33	5.09E-05 (6)*
		Ino4p	9.8	histone deacetylation	3.06	5.61E-05 (11,9)*
		Swi4p	9.4	M phase	10.20	6.35E-05 (6,5)*
39	292	Yap1p	24.9	biosynthetic process	29.15	3.90E-06 (3)
		Met4p	20.5	mitochondrial transport	4.08	1.04E-05 (5-7)*
		Sok2p	15.6	cellular metabolic process	63.64	1.07E-05 (3)*
		Rpn4p	11.9	lipid biosynthetic process	5.96	1.12E-05 (4-6)*
		Ste12p	11.0	lipid metabolic process	8.46	3.16E-05 (4-6)*
		Arr1p	10.7	cellular lipid metabolic process	8.15	3.28E-05 (4,5)*
		Swi4p	10.5			
		Abf1p	10.3			
		Aft1p	10.3			
		Mbp1p	9.8			

Figure 15

GO terms and transcriptional regulations of the 1-CCC-Biclusters describing transcriptional up-regulation patterns. This table shows a detailed analysis of the GO terms and transcriptional regulations of the 1-CCC-Biclusters describing transcriptional up-regulation patterns discovered by 1-CCC-Biclustering. When the set of genes in the 1-CCC-Bicluster has more than 10 transcription factors or more than 10 GO terms enriched, only the top 10 of each are shown. We only show the GO terms passing the Bonferroni correction for multiple testing at either the 1% level (highly significant) or the 5% level (significant). The *p*-values marked with * only passed the test at the 5% level. The *p*-values presented in the table are without correction as it is common practice in the literature.

ID	*#Genes*	*TFs (Top 10)*	*%*	*GO Terms Enriched (Top 10)*	*DF*	*p-value (level)*
10	1079	Yap1p	30.7	ribonucleoprotein complex biogenesis and assembly	26.91	2.56E-77 (4)
		Sfp1p	26.7	ribosome biogenesis and assembly	23.90	4.42E-72 (5)
		Met4p	23.1	nucleobase, nucleoside, nucleotide and nucleic acid metabolic process	44.68	3.82E-45 (4)
		Rap1p	16.2	RNA processing	21.40	3.00E-42 (6)
		Rpn4p	15.4	organelle organization and biogenesis	41.05	8.48E-42 (4)
		Arr1p	12.8	cellular component organization and biogenesis	56.70	2.22E-41 (3)
		Sok2p	10.6	RNA metabolic process	34.04	2.40E-40 (5)
		Ifh1p	10.6	rRNA metabolic process	13.52	7.60E-35 (6)
		Hhl1p	10.0	rRNA processing	13.02	4.33E-33 (6,7)
		Gcn4p	9.6	primary metabolic process	67.83	1.67E-27 (3)
27	597	Yap1p	18.3	glycoprotein biosynthetic process	5.34	6.28E-11 (7,5)
		Met4p	14.8	glycoprotein metabolic process	5.34	8.56E-11 (6)
		Sok2p	13.3	biopolymer glycosylation	4.91	5.38E-10 (6)
		Swi4p	12.6	protein amino acid glycosylation	4.91	5.38E-10 (6-8)
		Ste12p	12.1	cellular component organization and biogenesis	47.65	1.34E-09 (3)
		Rap1p	9.7	protein modification process	15.60	6.15E-09 (6)
		Rpn4p	9.7	protein amino acid N-linked glycosylation	3.63	1.36E-08 (7-9)
		Mbp1p	9.4	cellular metabolic process	64.53	1.54E-08 (3)
		Phd1p	8.7	biopolymer modification	17.95	5.18E-08 (5)
		Arr1p	8.7	cell cycle	13.25	3.32E-07 (3)
14	521	Yap1p	22.1	cellular component organization and biogenesis	50.24	1.51E-11 (3)
		Met4p	13.8	nucleobase metabolic process	3.14	1.62E-08 (5)
		Sfp1p	13.1	ribonucleoprotein complex biogenesis and assembly	14.98	2.73E-08 (4)
		Rpn4p	11.9	nucleobase, nucleoside, nucleotide and nucleic acid metabolic process	34.54	6.68E-08 (4)
		Sok2p	11.7	cellular biosynthetic process	12.08	9.61E-08 (4)
		Ste12p	11.3	glycoprotein biosynthetic process	4.59	2.18E-07 (7,5)
		Rap1p	11.3	ribosome biogenesis and assembly	12.80	2.34E-07 (5)
		Swi4p	10.4	glycoprotein metabolic process	4.59	2.70E-07 (6)
		Arr1p	10.2	biopolymer glycosylation	4.35	3.73E-07 (6)
		Leu3p	10.2	protein amino acid glycosylation	4.35	3.73E-07 (9,7,6)
68	800	Yap1p	34.3	ribonucleoprotein complex biogenesis and assembly	32.16	1.04E-68 (4)
		Sfp1p	32.6	ribosome biogenesis and assembly	28.82	1.68E-64 (5)
		Met4p	27.1	organelle organization and biogenesis	45.29	1.26E-36 (4)
		Rap1p	22.1	rRNA metabolic process	16.86	3.72E-34 (6)
		Rpn4p	19.4	RNA metabolic process	37.06	9.56E-33 (5)
		Fhl1p	16.1	RNA processing	23.33	5.74E-32 (6)
		Arr1p	15.8	cellular component organization and biogenesis	59.22	1.41E-31 (3)
		Ifh1p	15.2	rRNA processing	15.88	5.11E-31 (6,7)
		Sok2p	11.2	nucleobase, nucleoside, nucleotide and nucleic acid metabolic process	45.29	1.99E-29 (4)
		Ino4p	8.9	ribosomal large subunit biogenesis and assembly	6.08	3.56E-18 (6,5)
63	292	Yap1p	33.0	nucleobase, nucleoside, nucleotide and nucleic acid metabolic process	44.09	2.51E-10 (4)
		Rap1p	23.7	DNA metabolic process	20.43	3.85E-08 (5)
		Met4p	22.7	RNA metabolic process	31.72	6.39E-08 (5)
		Sfp1p	21.3	primary metabolic process	68.82	7.24E-08 (3)
		Ifh1p	18.6	cellular metabolic process	70.97	8.86E-08 (3)
		Rpn4p	17.5	cellular component organization and biogenesis	52.69	2.06E-07 (3)
		Fhl1p	16.2	establishment and/or maintenance of chromatin architecture	12.37	5.60E-07 (7)
		Arr1p	15.1	DNA packaging	12.37	5.60E-07 (6)
		Sok2p	13.4	chromosome organization and biogenesis (sensu Eukaryota)	19.89	8.44E-07 (6)
		Ino4p	11.3	chromatin modification	11.29	1.01E-06 (8)
122	1462	Yap1p	27.4	ribonucleoprotein complex biogenesis and assembly	19.68	6.32E-39 (4)
		Met4p	21.9	ribosome biogenesis and assembly	17.42	9.41E-37 (5)
		Sfp1p	19.1	nucleobase, nucleoside, nucleotide and nucleic acid metabolic process	39.03	1.07E-28 (4)
		Rap1p	16.4	RNA metabolic process	29.75	1.11E-27 (5)
		Sok2p	15.1	cellular component organization and biogenesis	50.68	1.88E-25 (3)
		Rpn4p	15.1	organelle organization and biogenesis	35.29	3.77E-25 (4)
		Arr1p	14.3	RNA processing	16.29	1.33E-21 (6)
		Fhl1p	10.5	rRNA metabolic process	10.18	3.13E-19 (6)
		Ste12p	10.2	rRNA processing	9.73	7.31E-18 (6,7)
		Ino4p	9.6	cellular metabolic process	63.57	3.07E-13 (3)

Figure 16

GO terms and transcriptional regulations of the 1-CCC-Biclusters describing transcriptional down-regulation patterns. This table shows a detailed analysis of the GO terms and transcriptional regulations of the 1-CCC-Biclusters describing transcriptional down-regulation patterns discovered by 1-CCC-Biclustering. When the set of genes in the 1-CCC-Bicluster has more than 10 transcription factors or more than 10 GO terms enriched, only the top 10 of each are shown. We only show the GO terms passing the Bonferroni correction for multiple testing at either the 1% level (highly significant) or the 5% level (significant). The *p*-values marked with * only passed the test at the 5% level. The *p*-values presented in the table are without correction as it is common practice in the literature.

#	ID	Sorting p-value	Variation Pattern	#Time Points (first-last)	#Genes	# (corrected) GO p-values < 0.01	# (corrected) GO p-values $0.01 \leq < 0.05$
1	**124**	2.56E-84	DNU	4(2-5)	904	40	8
3	**14**	1.64E-58	UND	4(2-5)	1091	62	12
4	**27**	3.69E-44	UUND	5(1-5)	290	7	6
5	**39**	8.65E-42	UNND	5(1-5)	258	**0**	**0**
8	**151**	3.99E-31	DNNU	5(1-5)	232	12	2
9	48	1.35E-26	UDUD	5(1-5)	182	**0**	1
10	142	2.84E-24	DUDU	5(1-5)	248	8	19
11	43	6.56E-24	UNDD	5(1-5)	109	**0**	**0**
12	**147**	6.03E-21	DNUU	5(1-5)	144	**0**	3
15	83	1.90E-16	NUNN	5(1-5)	224	2	4
17	42	3.30E-11	UNDN	5(1-5)	131	2	1
18	148	6.00E-11	DNUN	5(1-5)	192	4	**0**
21	159	1.37E-07	DDUU	5(1-5)	56	**0**	**0**
22	79	4.41E-07	NUUN	5(1-5)	97	2	3
24	92	3.88E-05	NNUN	5(1-5)	52	2	**0**
25	99	4.79E-05	NNDN	5(1-5)	39	1	**0**

Figure 17

CCC-Biclusters surviving the overlapping filter. This table shows a summary of the CCC-Biclusters discovered by the CCC-Biclustering algorithm surviving the overlapping filter. It also shows the statistical *p*-value of their expression patterns, the patterns themselves, the number of contiguous time points and the number of genes in each CCC-Bicluster. Together with this information we present a summary of the results obtained when analyzing the 16 CCC-Biclusters using the Gene Ontology annotations restricted to those in the "Biological Process" ontology and terms above Level 2. We show the number of GO terms with highly significant and significant *p*-values, respectively, after Bonferroni correction for multiple testing. Several CCC-Biclusters are not functionally enriched after Bonferroni correction.

rection). This means some of the CCC-Biclusters identified by the CCC-Biclustering algorithm may not be biologically relevant according with the GO analysis.

Figure 18 shows the relationship between the top 12 1-CCC-Biclusters discovered by 1-CCC-Biclustering in Figure 14 (CCC-Biclusters with approximate patterns allowing one error per gene relatively to the expression pattern identifying the 1-CCC-Bicluster) and the top 16 CCC-Biclusters discovered by CCC-Biclustering in Figure 17 (CCC-Biclusters with perfect expression patterns). It is clear from this figure that, apart from two 1-CCC-Biclusters (IDs 68 and 122), all other 1-CCC-Biclusters correspond to the extension of one or several of the 16 CCC-Biclusters by adding genes with approximate expression patterns. The CCC-Bicluster with ID 124 was extended not only with genes with approximate patterns but also with a contiguous column at the left of its expression pattern.

It is worth noting that all the resulting 1-CCC-Biclusters have a larger number of GO terms functionally enriched. Moreover, even when the CCC-Biclusters are not functionally enriched, the 1-CCC-Biclusters obtained by considering approximate expression patterns instead of perfect patterns are *always* functionally enriched.

In order to show that the number of genes regulated by relevant TFs has increased in the 1-CCC-Biclusters when compared with the same number in the corresponding CCC-Biclusters, we used a set of relevant CCC-Biclusters chosen by Madeira et al. among the top 16 CCC-Biclusters in Figure 17. From these top 16 CCC-Biclusters the authors selected 6 CCC-Biclusters, which were then analyzed in more detail using the Gene Ontology annotations together with information about transcriptional regulation available in the YEASTRACT database. These *selected CCC-Biclusters* describe either transcriptional up-regulation (CCC-Biclusters with IDs 39, 27 and 14) or down-regulation patterns (CCC-Biclusters with IDs 147, 151 and 124). For these 6 CCC-Biclusters the authors identified relevant transcription factors (TFs) according to their expression pattern and relevant GO terms. For example, the heat-shock factor Hsf1p, together with the transcription factors Msn2p and Msn4p, known regulators of the general stress response in yeast, and the transcription factor Rpn4p, known stimulator of the proteosome genes, involved in the degradation of denatured or unnecessary proteins in stressed yeast cell [9], were identified by the authors as relevant TFs in CCC-Biclusters 39, 27 and 14. Note that apart from CCC-Bicluster 14, whose corresponding 1-CCC-Biclusters were removed during the

#	ID	Variation Pattern	#Genes	# (corrected) GO p-values < 0.01	# (corrected) GO p-values 0.01 ≤< 0.05	Corresponding CCC-Bicluster
1	**10**	**DDNU**	**1079**	**58**	**16**	
			904	40	8	**#1 (ID 124)**
			232	12	2	**#8 (ID 151)**
			56	0	0	**#21 (ID 159)**
2	**27**	**DNUU**	**597**	**22**	**13**	
			144	0	3	**#12 (ID 147)**
			232	12	2	**#8 (ID 151)**
			192	4	4	**#18 (ID 148)**
			56	0	0	**#21 (ID 159)**
7	**79**	**NNND**	**849**	**40**	**16**	
			258	0	0	**#5 (ID 39)**
9	**132**	**UNDD**	**539**	**10**	**7**	
			109	0	0	**#11 (ID 43)**
			258	0	0	**#5 (ID 39)**
			131	2	1	**#17 (ID 42)**
13	**145**	**UUDD**	**511**	**8**	**5**	
			290	7	6	**#4 (ID 27)**
			109	0	0	**#11 (ID 43)**
16	**14**	**DDUU**	**521**	**19**	**12**	
			56	0	0	**#21 (ID 159)**
			144	0	3	**#12 (ID 147)**
19	**68**	**NDNU**	**800**	**41**	**6**	
28	**97**	**NUUN**	**385**	**5**	**0**	
			97	2	3	**#22 (ID 79)**
			224	2	4	**#15 (ID 83)**
			52	2	0	**#24 (ID 92)**
34	**120**	**UDDN**	**307**	**6**	**5**	
			131	2	1	**#17 (ID 42)**
35	**63**	**NDDN**	**292**	**13**	**1**	
			52	2	0	**#25 (ID 99)**
36	**39**	**DUUN**	**430**	**1**	**5**	
			192	0	4	**#18 (ID 148)**
			97	2	3	**#22 (ID 79)**
43	**122**	**UDN**	**1462**	**33**	**15**	

Figure 18

Best CCC-Biclusters versus best 1-CCC-Biclusters. This table shows the relationship between the top 12 1-CCC-Biclusters and the top 16 CCC-Biclusters. It is clear that, apart from two 1-CCC-Biclusters (IDs 68 and 122), all other 1-CCC-Biclusters correspond to the extension of one or several of the 16 CCC-Biclusters by adding genes with approximate expression patterns or extending the expression pattern of the CCC-Bicluster with a contiguous columns. All the resulting 1-CCC-Biclusters have a larger number of GO terms functionally enriched and are thus more relevant according to the functional enrichment analysis performed using the Gene Ontology.

application of the overlapping filter (see additional file 5: **1_ccc_biclusters_vs_ccc biclusters**), all these selected CCC-Biclusters have at least one corresponding 1-CCC-Bicluster in the top 12 (as it is also shown in Figure 18).

In this context, we decided to compare the number of genes regulated by each relevant TF identified in each of the selected CCC-Biclusters and the number of genes regulated by the same TFs in the corresponding 1-CCC-Biclusters. Remember that, if relevant genes (not included in these CCC-Biclusters due to a single error) were recovered and included in the corresponding 1-CCC-Biclusters, the number of genes regulated by relevant TFs should increase in the 1-CCC-Bicluster.

Figure 19 shows, for each of the 5 selected CCC-Biclusters considered (CCC-Biclusters with IDs 39, 27, 147, 151 and 124), the set of relevant transcription factors together with the number of regulated genes and compares these numbers with those obtained for the same TFs in the corresponding 1-CCC-Bicluster(s). It is clear from this figure that the number of genes regulated by relevant TFs always increases in the corresponding 1-CCC-Biclusters. These results support our idea that *e*-CCC-Biclustering is able to recover genes with relevant expression patterns, that were missed due to small errors, and are in fact, biologically relevant to the problem under study. For example, in CCC-Bicluster 39, the relevant transcription factors Sok2p, Arr1p, Hsf1p, Rpn4p and Msn2p, regulated, respectively, 70, 37, 36, 32 and 32 of the 258 genes in the CCC-Bicluster. In the corresponding 1-CCC-Bicluster (ID 79) with 849 genes, these key transcription factors regulate, respectively, 189, 115, 142, 123 and 147 genes.

These results demonstrate that allowing the discovery of CCC-Biclusters with approximate patterns (*e*-CCC-Biclusters), rather than restricting the analysis to CCC-Biclusters with perfect expression patterns, can in fact improve the biological significance of the obtained results. These results also show, the superiority of the proposed *e*-CCC-Biclustering, algorithm, when compared with the CCC-Biclustering approach, in the identification of biologically relevant temporal patterns of expression.

Conclusion and future work

In this work we proposed *e*-CCC-Biclustering, a new biclustering algorithm specifically developed for time series gene expression data analysis, that finds and reports all maximal contiguous column coherent biclusters with approximate expression patterns in time polynomial in the size of the expression matrix. These approximate patterns allow a given number of errors, per gene, relatively to an expression profile representing the expression pattern in the *e*-CCC-Bicluster. We described the algorithmic details of *e*-CCC-Biclustering, analyzed its computational complexity, and proposed extensions to improve the ability of the algorithm to discover other relevant expression patterns by being able to deal with missing values and allowing anticorrelated and scaled expression patterns. We also discussed different ways to compute the errors allowed in the approximate expression patterns. Finally, we described a scoring criterion based on a statistical test, used to sort *e*-CCC-Biclusters by increasing value of the probability that they have appeared by a random coincidence of events. Coupled with a similarity measure, used to filter highly overlapping *e*-CCC-Biclusters, this scoring criterion effectively identifies not only statistically but also biologically relevant *e*-CCC-Biclusters, which can then be useful to identify regulatory modules.

The results show the effectiveness of the approach and its relevance in the discovery of regulatory modules describing the transcriptomic expression patterns occurring in *Saccharomyces cerevisiae* in response to heat stress. Moreover, the comparison performed with a state of the art biclustering algorithm specifically developed for time series gene expression data analysis demonstrated the superiority of *e*-CCC-Biclustering in discovering statistically and biologically relevant temporal patterns of expression.

As short term future work, we plan to extend the algorithm to detected time-lagged regulations between genes and temporal patterns of expression in multiple time series gene expression matrices. The proposed algorithm can be easily extended to discover *e*-CCC-Biclusters with time-lags, enabling the discovery of important time-lagged regulations between genes, such as activation and inhibition, as well as temporal programs of expression, in which genes are activated one by one in a predefined order. Moreover, extending the algorithm to identify local temporal patterns of expression using multiple datasets should enable the discovery of conserved expression patterns and potentially help in the identification of common regulatory modules within and across-species. Our medium and long term research will be related with the use of the information about coherent expression patterns and co-regulation in the identification of regulatory modules, potentially helpful in the challenging area of inferring regulatory networks. This will require the development of efficient inference methods able to integrate heterogeneous data such as gene expression data, sequence data, and textual information scattered in scientific literature.

Competing interests

The authors declare that they have no competing interests.

Selected CCC-Biclusters					Corresponding 1-CCC-Biclusters				
ID	*#Genes*	*Relevant TFs*	*%*	#Regulated Genes	*ID*	*#Genes*	*TFs*	*%*	*#Regulated Genes*
39	258	Sok2p	27.3	70	**79**	849	Sok2p	22.3	189
		Arr1p	14.5	37			Arr1p	13.5	115
		Hsf1p	14.1	36			Hsf1p	16.7	142
		Rpn4p	12.5	32			Rpn4p	14.5	123
		Msn2p	12.5	32			Msn2p	17.3	147
		Msn4p	9.8	25			Msn4p	14.5	123
27	290	Sok2p	26.0	75	**145**	511	Sok2p	24.0	123
		Hsf1p	22.1	64			Hsf1p	17.7	90
		Msn2p	19.4	56			Msn2p	15.0	77
		Rpn4p	17.3	50			Rpn4p	14.2	73
		Msn4p	16.6	48			Msn4p	12.6	64
147	144	Ste12p	16.0	23	**27**	597	Ste12p	12.1	72
		Rap1p	13.2	19			Rap1p	9.7	58
		Swi4p	12.5	18			Swi4p	12.6	75
		Rpn4p	11.1	16			Rpn4p	9.7	58
		Ino4p	9.7	14			Ino4p	8.1	48
					14	1091	Ste12p	11.3	123
							Rap1p	11.3	123
							Swi4p	10.4	113
							Rpn4p	11.9	130
							Ino4p	10.0	109
151	232	Swi4p	13.8	32	**10**	1079	Swi4p	9.4	101
		Mbp1p	10.3	24			Mbp1p	8.8	95
		Arr1p	9.5	22			Arr1p	12.8	138
		Rpn4p	8.2	19			Rpn4p	15.4	166
		Ino4p	7.3	17			Ino4p	9.4	101
					27	597	Swi4p	12.6	75
							Mbp1p	9.4	56
							Arr1p	8.7	52
							Rpn4p	9.7	58
							Ino4p	8.1	48
124	904	Sfp1p	29.6	268	**10**	1079	Sfp1p	26.7	288
		Rap1p	18.7	169			Rap1p	16.2	175
		Rpn4p	16.9	153			Rpn4p	15.4	166
		Arr1p	14.5	131			Arr1p	12.8	138
		Fhl1p	11.6	105			Fhl1p	10.0	108

Figure 19

Number of genes regulated by relevant TFs in selected CCC-Biclusters versus corresponding 1-CCC-Biclusters. This table compares the number of genes regulated by the relevant TFs of the 5 selected CCC-Biclusters (CCC-Biclusters with IDs 39, 27, 147, 151 and 124) with the number of genes regulated by the same TFs in the corresponding 1-CCC-Biclusters. Note that these TFs might not appear in the top 10 in Figure 15 and Figure 16. It is clear from this table that the number of regulated genes by relevant TFs always increases in the corresponding 1-CCC-Biclusters.

Authors' contributions

SCM and ALO designed the *e*-CCC-Biclustering algorithm together with the proposed extensions and defined the scoring criterion for *e*-CCC-Biclusters based on the statistical significance of their expression patterns and similarity with other overlapping biclusters. SCM coded the prototype implementation of the algorithm in Java and wrote the first draft of the manuscript. SCM and ALO worked together towards the final version of the manuscript. All authors read and approved the final manuscript.

Additional material

Additional file 1

e-*CCC-Biclustering: Related work on biclustering algorithms for time series gene expression data*. *Supplementary material describing related work on biclustering algorithms for time series gene expression data analysis. We describe in detail three state of the art biclustering approaches specifically designed to identify biclusters in gene expression time series and identify their strengths and weaknesses. We also explain and justify why we decided to compare the performance of* e-CCC-Biclustering *with that of CCC-Biclustering, but not with that of the* q-*clustering and CC-TSB algorithms.*
Click here for file
[http://www.biomedcentral.com/content/supplementary/1748-7188-4-8-S1.pdf]

Additional file 2

e-*CCC-Biclustering: Algorithmic and complexity details*. *Supplementary material describing algorithmic and complexity details of* e-CCC-Biclustering.
Click here for file
[http://www.biomedcentral.com/content/supplementary/1748-7188-4-8-S2.pdf]

Additional file 3

Highly significant 1-CCC-Biclusters. *Table showing a summary of the 47 1-CCC-Biclusters passing the Bonferroni correction for multiple testing at the 1% level when 1-CCC-Biclustering restricted to errors in the 1-neighborhood of the symbols in the alphabet* $\Sigma = \{D, N, U\}$ *was applied to the* ***DiscretizedHeatShock*** *dataset.*
Click here for file
[http://www.biomedcentral.com/content/supplementary/1748-7188-4-8-S3.pdf]

Additional file 4

Highly significant CCC-Biclusters. *Table showing a summary of the 25 CCC-Biclusters passing the Bonferroni correction for multiple testing at the 1% level when CCC-Biclustering was applied to the* ***DiscretizedHeatShock*** *dataset.*
Click here for file
[http://www.biomedcentral.com/content/supplementary/1748-7188-4-8-S4.pdf]

Additional file 5

Highly significant 1-CCC-Biclusters **versus** ***highly significant CCC-Biclusters***. *Table showing a comparison between the 47 highly significant 1-CCC-Biclusters discovered by 1-CCC-Biclustering restricted to errors in the 1-neighborhood of the symbols in the alphabet* $\Sigma = \{D, N, U\}$ *and the 16 highly significant CCC-Biclusters found by CCC-Biclustering (after the applying the overlapping filter) and analyzed by Madeira et al. [9]. Both sets of biclusters were identified when the algorithm was applied to the* ***DiscretizedHeatShock*** *dataset.*
Click here for file
[http://www.biomedcentral.com/content/supplementary/1748-7188-4-8-S5.pdf]

Additional file 6

GO terms enriched and transcriptional regulations of the top 16 CCC-Biclusters. *Table showing a detailed analysis of the GO terms enriched and transcriptional regulations of the top 16 CCC-Biclusters discovered with CCC-Biclustering. When the set of genes in the CCC-Bicluster have more than 10 transcription factors or more than 10 GO terms enriched, only the top 10 of each are shown. We only show the GO terms passing the Bonferroni correction for multiple testing at either the 1% level (highly significant) or the 5% level (significant). The* p-*values marked with * only passed the test at the 5% level. The* p-*values presented in the table are without correction as it is common practice in the literature.*
Click here for file
[http://www.biomedcentral.com/content/supplementary/1748-7188-4-8-S6.pdf]

Acknowledgements

Parts of this work have appeared previously in [32]. However, this manuscript describes algorithmic and complexity details not included in the conference version of the paper. We also present a detailed comparison with other algorithms with similar goals highlighting their strengths and weaknesses. The extensions to allow missing values, anticorrelation, scaling, alphabet range weighted errors and pattern length adaptive errors are original. The proposed approach to score e-CCC-Biclusters using a statistical significance criterion and a similarity measure, only superficially mentioned in the conference paper, was improved and used in the experimental results. All the experimental results are new. The algorithm was applied to a new dataset to identify transcriptional regulatory modules. Moreover, the superiority of CCC-Biclusters with approximate expression patterns (e-CCC-biclusters) relatively to CCC-Biclusters (perfect expression patterns) was demonstrated using two biological criteria: stronger evidence of functional enrichment (regarding the *p*-values of the GO terms enriched and the number of GO terms enriched) and increased number of genes regulated by relevant transcription factors.

This work was partially supported by projects ARN – Algorithms for the Identification of Genetic Regulatory Networks, PTDC/EIA/67722/2006, and Dyablo – Models for the Dynamic Behavior of Biological Networks, PTDC/EIA/71587/2006, funded by FCT, Fundafição para a Ciência e Tecnologia.

References

1. Bar-Joseph Z: **Analyzing time series gene expression data.** *Bioinformatics* 2004, **20(16):**2493-2503.
2. Androulakis IP, Yang E, Almon RR: **Analysis of Time-Series Gene Expression Data: Methods, Challenges and Opportunities.** *Annual Review of Biomedical Engineering* 2007, **9:**205-228.
3. McLachlan GJ, Do K, Ambroise C: *Analysing microarray gene expression data* Wiley Series in Probability and Statistics; 2004.

4. Cheng Y, Church GM: **Biclustering of Expression Data.** In *Proc of the 8th International Conference on Intelligent Systems for Molecular Biology* 2000:93-103.
5. Mechelen IV, Bock HH, Boeck PD: **Two-mode clustering methods: a structured overview.** *Stat Methods Med Res.* 2004, **13(5):**363-394.
6. Madeira SC, Oliveira AL: **Biclustering algorithms for biological data analysis: a survey.** *IEEE/ACM Trans Comput Biol Bioinform* 2004, **1(1):**24-45.
7. Tanay A, Sharan R, Shamir R: **Discovering statistically significant biclusters in gene expression data.** *Bioinformatics.* 2002, **18(Suppl 1):**S136-S144.
8. Ben-Dor A, Chor B, Karp R, Yakhini Z: **Discovering Local Structure in Gene Expression Data: The Order-Preserving Submatrix Problem.** *J Comput Biol* 2002, **10(3-4):**373-384.
9. Madeira SC, Teixeira MC, Sá-Correia I, Oliveira AL: **Identification of Regulatory Modules in Time Series Gene Expression Data using a Linear Time Biclustering Algorithm.** In *IEEE/ACM Transactions on Computational Biology and Bioinformatics, 21 Mar. 2008* IEEE Computer Society Digital Library. IEEE Computer Society.
10. Peeters R: **The maximum edge biclique problem is NP-complete.** *Discrete Applied Mathematics* 2003, **131(3):**651-654.
11. Yang E, Foteinou PT, King K, Yarmush ML, Androulakis I: **A novel non-overlapping bi-clustering algorithm for network generation using living cell array data.** *Bioinformatics* 2007, **23(17):**2306-2313.
12. Murali TM, Kasif S: **Extracting conserved gene expression motifs from gene expression data.** *Pac Symp Biocomput* 2003:77-88.
13. Koyuturk M, Szpankowski W, Grama A: **Biclustering Gene-Feature Matrices for Statistically Significant Dense Patterns.** In *Proc of the 8th International Conference on Research in Computational Molecular Biology* 2004:480-484.
14. Liu J, Wang W, Yang J: **Biclustering in gene expression data by tendency.** In *Proc of the 3rd International IEEE Computer Society Computational Systems Bioinformatics Conference* 2004:182-193.
15. Liu J, Wang W, Yang J: **A framework for ontology-driven subspace clustering.** In *Proc of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 2004:623-628.
16. Liu J, Wang W, Yang J: **Gene ontology friendly biclustering of expression profiles.** In *Proc of the 3rd IEEE Computational Systems Bioinformatics Conference* 2004:436-447.
17. Liu J, Wang W, Yang J: **Mining Sequential Patterns from Large Data Sets.** *Kluwer* 2005, **18:**.
18. Lonardi S, Szpankowski W, Yang Q: **Finding Biclusters by Random Projections.** In *Proc of the 15th Annual Symposium on Combinatorial Pattern Matching* 2004:102-116.
19. Sheng Q, Moreau Y, Moor BD: **Biclustering microarray data by Gibbs sampling.** *Bioinformatics* 2003, **19 Suppl 2:**ii196-ii205.
20. Ji L, Tan K: **Identifying time-lagged gene clusters using gene expression data.** *Bioinformatics* 2005, **21(4):**509-516.
21. Wu C, Fu Y, Murali TM, Kasif S: **Gene expression module discovery using Gibbs sampling.** *Genome Informatics* 2004, **15:**239-248.
22. Madeira SC, Oliveira AL: **A Linear Time Biclustering Algorithm for Time Series Gene Expression Data.** In *Proc of the 5th Workshop on Algorithms in Bioinformatics* Springer Verlag, LNCS/LNBI 3692; 2005:39-52.
23. Prelic A, Bleuler S, Zimmermann P, Wille A, Buhlmann P, Gruissem W, Hennig L, Thiele L, Zitzler E: **A systematic comparison and evaluation of biclustering methods for gene expression data.** *Bioinformatics* 2006, **22(10):**1282-1283.
24. Zhang Y, Zha H, Chu CH: **A Time-Series Biclustering Algorithm for Revealing Co-Regulated Genes.** In *Proc of the 5th IEEE International Conference on Information Technology: Coding and Computing* 2005:32-37.
25. Gusfield D: *Algorithms on strings, trees, and sequences* Computer Science and Computational Biology Series, Cambridge University Press; 1997.
26. Sagot MF: **Spelling approximate repeated or common motifs using a suffix tree.** In *Proc of Latin'98* Springer Verlag, LNCS 1380; 1998:111-127.
27. Madeira SC: **Efficient Biclustering Algorithms for Time Series Gene Expression Data Analysis.** In *PhD thesis* Instituto Superior Técnico, Technical University of Lisbon; 2008.
28. Martin D, Brun C, Remy E, Mouren P, Thieffry D, Jacq B: **GOToolBox: functional investigation of gene datasets based on Gene Ontology.** *Genome Biology* 2004, **5(12**R101 [http://burgundy.cmmt.ubc.ca/GOToolBox/].
29. Teixeira MC, Monteiro P, Jain P, Tenreiro S, Fernandes AR, Mira NP, Alenquer M, Freitas AT, Oliveira AL, Sa-Correia I: **The YEASTRACT database: a tool for the analysis of transcription regulatory associations in Saccharomyces cerevisiae.** *Nucleic Acids Research* 2006, **34:**D446-D451 [http://www.yeastract.com/].
30. Gasch AP, Spellman PT, Kao CM, Carmel-Harel O, Eisen MB, Storz G, Botstein D, Brown PO: **Genomic Expression Programs in the Response of Yeast Cells to Environmental Changes.** *Molecular Biology of the Cell* 2000, **11:**4241-4257.
31. Ji L, Tan K: **Mining gene expression data for positive and negative co-regulated gene clusters.** *Bioinformatics* 2004, **20(16):**2711-2718.
32. Madeira SC, Oliveira AL: **An Efficient Biclustering Algorithm for finding Genes with Similar Patterns in Time-Series Expression Data.** In *Proc of the 5th Asia Pacific Bioinformatics Conference, Series in Advances in Bioinformatics and Computational Biology Volume 5*. Imperial College Press; 2007:67-80.