LateBiclustering: Efficient Heuristic Algorithm for Time-Lagged Bicluster Identification

Joana P. Gonçalves and Sara C. Madeira

Abstract—Identifying patterns in temporal data is key to uncover meaningful relationships in diverse domains, from stock trading to social interactions. Also of great interest are clinical and biological applications, namely monitoring patient response to treatment or characterizing activity at the molecular level. In biology, researchers seek to gain insight into gene functions and dynamics of biological processes, as well as potential perturbations of these leading to disease, through the study of patterns emerging from gene expression time series. Clustering can group genes exhibiting similar expression profiles, but focuses on global patterns denoting rather broad, unspecific responses. Biclustering reveals local patterns, which more naturally capture the intricate collaboration between biological players, particularly under a temporal setting. Despite the general biclustering formulation being NP-hard, considering specific properties of time series has led to efficient solutions for the discovery of temporally aligned patterns. Notably, the identification of biclusters with time-lagged patterns, suggestive of transcriptional cascades, remains a challenge due to the combinatorial explosion of delayed occurrences. Herein, we propose LateBiclustering, a sensible heuristic algorithm enabling a polynomial rather than exponential time solution for the problem. We show that it identifies meaningful time-lagged biclusters relevant to the response of *Saccharomyces cerevisiae* to heat stress.

Index Terms—Time series, time lag, biclustering, string matching, pattern recognition, local pattern, pattern matching

1 Introduction

Gene expression is a dynamic process, reflecting changes orchestrated by the underlying regulatory mechanisms involved in cellular control. Temporal gene expression profiling enables to monitor the responses of a large number of regulatory players over time and is recognized as a key strategy to gain insight into the intricate circuitry of gene regulation. Ultimately, the analysis of time series gene expression data is critical to advance our understanding of complex biological mechanisms involved in processes such as growth and development, disease susceptibility and progression, and response to treatment [1], [2].

1.1 Motivation

The assumption that genes involved in a particular biological process tend to exhibit coherent behavior has long driven the discovery of common patterns in gene expression profiles. Effectively capturing relationships between gene responses requires consideration of specific features inherent to transcriptional activity. For instance, expression patterns have a local nature, since biological processes occur within delimited time frames [3], [4]. Moreover, each gene can potentially collaborate with different sets of partners in multiple biological processes. Additionally, genes may be activated or inhibited with particular delays, which causes resembling patterns to be shifted in time. Locality, overlap and time lags between transcriptional patterns are therefore prominent features of gene expression time series and should assume major relevance in the analysis of these data.

1.2 Background

Biclustering can effectively unravel local patterns, but its general formulation is NP-hard upon reduction to the maximum edge biclique problem [5]. Many general purpose biclustering algorithms have been proposed [4]. However, most are unsuitable for the analysis of time series data given that they disregard important temporal properties, such as time point dependency and time contiguity inherent to biological processes. The assumption that processes last for a delimited period of time motivates the discovery of local patterns spanning consecutive time points (time frames). Notably, this restriction coupled with a formulation based on discretized expression data have enabled a linear time solution for the temporal biclustering problem [6]. This is the most efficient temporal biclustering algorithm to date, also effective in unraveling biologically meaningful gene sets [6], [7]. However, it is limited to temporally aligned patterns.

Time-lagged relationships between gene expression profiles are an important aspect of gene regulation, as different target genes are often activated with a certain time delay rather than simultaneously. Temporal programs of expression in which genes are activated one by one in a predefined order are well-known and can be generated by widespread network topologies, including regulatory cascades [8]. The identification of time-lagged patterns to uncover such relationships has been addressed before [9], [10], but available

- J.P. Gonçalves is with Centrum Wiskunde & Informatica (CWI), Science Park 123, 1098 XG Amsterdam, Netherlands, and with INESC-ID, Rua Alves Redol 9, 1000-029 Lisboa, Portugal. E-mail: joana.goncalves@cwi.nl, jpg@kdbio.inesc-id.pt.
- S.C. Madeira is with INESC-ID and Instituto Superior Técnico, University of Lisbon, Rua Alves Redol 9, 1000-029 Lisboa, Portugal. E-mail: sara.madeira@tecnico.ulisboa.pt.

Manuscript received 7 Nov. 2013; revised 6 Feb. 2014; accepted 6 Mar. 2014. Date of publication 16 Mar. 2014; date of current version 2 Oct. 2014. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TCBB.2014.2312007

	$T1$	$T2$	$T3$	$T4$	$T5$
$G1$	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

(a) Original matrix.

	$T1$	$T2$	$T3$	$T4$	$T5$
$G1$	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

(b) Discretized matrix.

Fig. 1. Illustrative (a) time series expression matrix, together with its (b) discretized version using alphabet $\{D, N, U\}$ and interval $]-0.25, 0.25]$ for N.

	$T1$	$T2$	$T3$	$T4$	$T5$
$G1$	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

(a) Transformed matrix.

$G1$	N1	U2	D3	U4	N5	\$1
$G2$	D1	U2	D3	U4	D5	\$2
$G3$	N1	N2	N3	U4	N5	\$3
$G4$	U1	U2	D3	U4	U5	\$4
$G5$	U1	D2	U3	D4	U5	\$5

(b) Strings for suffix tree.

Fig. 2. Transformed matrix and strings used in suffix tree construction: (a) discretized matrix from Fig. 1b after alphabet transformation; (b) strings obtained from matrix (a) and used to build the suffix tree.

approaches are hampered by the potential explosion of pattern combinations which makes exhaustive enumeration infeasible in most cases of interest.

1.3 Our Approach

In this work, we discuss the exponential complexity of a solution for identifying and exhaustively reporting occurrences of time-lagged local patterns in temporal data [10]. We propose LateBiclustering, a heuristic algorithm with polynomial time complexity. Finally, we show that LateBiclustering identifies meaningful time-lagged biclusters and patterns relevant to the response of *Saccharomyces cerevisiae* to heat stress.

2 Methods

In this section, we propose LateBiclustering, an efficient algorithm for biclustering of time-lagged patterns. We first provide an overview of concepts related to biclustering in time series data and describe CCC-Biclustering [6], a linear time algorithm that identifies temporal biclusters with aligned pattern occurrences. We then present LateBiclustering in a constructive way, supported by definitions and theoretical results together with proofs. In this context, we describe bicluster identification and reporting heuristics that lead to a polynomial time complexity algorithm.

2.1 Time Series Data Matrix

Let M' be a matrix defined by a set of genes (rows), G, and a set of time points (columns), T, where M'_{ij} represents the expression of gene i at time point j. Real values in M' are discretized to a set of symbols, Σ, representing activation levels in a new matrix M. We use $\Sigma = \{D, N, U\}$, denoting *down-regulation*, *no-change* and *up-regulation* (Fig. 1). A reasonable approach for time series can be converting matrix M' into M such that $M_{ij} \in \Sigma$ reflects the expression trend of gene i between time points j and $j+1$ [6], [9].

Discretization causes loss of information and may induce further noise or errors. Notably, it is also an effective dimensionality reduction technique that can be exploited to derive tractable formulations of an inherently difficult problem [6]. Moreover, there is a growing interest in capturing trends by focusing on pattern shape rather than values [11], and biologists have long been using thresholds on fold changes to detect differentially expressed genes. Considering that the algorithms in this work are not tied to a particular discretization method, the use of sensible application-specific strategies can lead to improved results.

2.2 Mining Temporally Aligned Local Patterns

We briefly describe CCC-Biclustering [6], a linear time algorithm for mining biclusters of temporally aligned local patterns in a time series matrix.

2.2.1 *(Maximal) CCC-Bicluster*

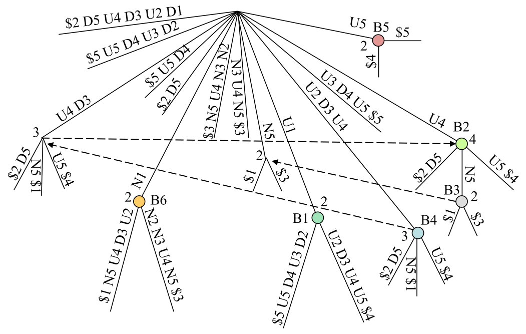
A CCC-Bicluster M_{IJ} is a subset of genes $I \subseteq G$ and a subset of contiguous time points $J \subseteq T$ such that $M_{ij} = M_{lj}$, $\forall i, l \in I$ and $\forall j \in J$. This means that every gene in I shares the same expression pattern spanning the time points in J. A CCC-Bicluster is maximal (Fig. 3) if adding rows to I violates the coherence of the expression pattern (row-maximality) and adding a symbol to the beginning or end of the expression pattern induces changes in I (left-/right-maximality). CCC-Biclusters containing a single row are biologically uninteresting and are thus disregarded.

2.2.2 *CCC-Biclustering*

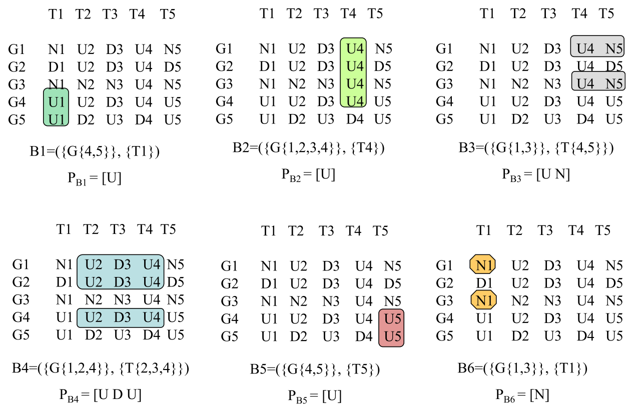
To find all maximal CCC-Biclusters, CCC-Biclustering first performs a simple alphabet transformation that appends the column number to each symbol in the discretized matrix (Fig. 2). This transformation ensures that patterns match only when both the symbol and time point match, and therefore when the patterns are temporally aligned. Regarding the rows of the transformed matrix as strings, denoting gene expression profiles, a generalized suffix tree $\mathcal{T}$ [12] is then built to match the common local patterns in the profiles and identify the maximal CCC-Biclusters (Fig. 3). Such identification relies on the following relationship between maximal CCC-Biclusters and nodes in $\mathcal{T}$: every right and row-maximal CCC-Bicluster with at least two rows corresponds to one internal node in $\mathcal{T}$ and every internal node in $\mathcal{T}$ corresponds to one right and row-maximal CCC-Bicluster with at least two rows. Right- and row-maximality of the CCC-Bicluster identified by an internal node v are guaranteed by generalized suffix tree construction. Left-maximality of a CCC-Bicluster identified by an internal node v is guaranteed when either v has no incoming suffix links [12] or it has incoming suffix links only from nodes for which the number of leaves in their subtree is equal to the number of leaves in the subtree rooted at v. CCC-Biclustering uses efficient string matching techniques to find these nodes and report all maximal CCC-Biclusters in time linear on the size of the matrix.

2.3 Mining Time-Lagged Local "Late" Patterns

In this work, we address the goal of finding occurrences of the same pattern which might not necessarily be temporally aligned. We shall focus on the general case of



(a) Maximal CCC-Biclusters in the suffix tree.



(b) Maximal CCC-Biclusters in the transformed matrix.

Fig. 3. Maximal CCC-Biclusters identified in the matrix in Fig. 2, shown in: (a) the suffix tree, and (b) the matrix.

	$T1$	$T2$	$T3$	$T4$	$T5$
$G1$	_	U	D	U	N
$G2$	D	U	_	U	_
$G3$	_	N	_	U	N
$G4$	U	_	D	U	U
$G5$	U	_	U	D	U

(a) Discretized matrix.

$G1$	U	D	U	N	\$1
$G2$	D	U	\$2		
	U	\$2			
$G3$	N	\$3			
	U	N	\$3		
$G4$	U	\$4			
	D	U	U	\$4	
$G5$	U	\$5			
	U	D	U	\$5	

(b) Set of strings.

Fig. 4. Discretized matrix and strings for LateBiclustering: (a) illustrative discretized matrix with missing values; (b) strings obtained from matrix (a) and used to build a generalized suffix tree for LateBiclustering.

unbounded time lags. For completeness, we present a sample matrix with missing values (Fig. 4).

2.3.1 (Maximal) LateBicluster and LateBiclustering

We first introduce key concepts, namely pattern, occurrence and bicluster in time series data, together with properties such as time delay and maximality.

Definition 1 (Pattern). *A string containing symbols from alphabet Σ used in the discretization of M.*

Definition 2 (Occurrence of a Pattern). *Instance of a given pattern found within a string representing a particular row of a time series matrix M. Besides the pattern itself, an occurrence is also characterized by its set of matrix "location coordinates": gene identifier (row) and earliest time point (left-most column).*

Definition 3 (Right-Maximal Pattern). *Pattern whose extension to the right through the addition of any valid symbol from alphabet Σ induces a decrease in the number of occurrences of such pattern in M.*

Definition 4 (LatePattern). *Right-maximal pattern with at least two occurrences in M.*

Note that the occurrences of a LatePattern are not necessarily in distinct gene profiles (rows of M).

Definition 5 (Time Lag). *Absolute difference between the earliest time points (left-most columns) of two occurrences of a given pattern.*

Definition 6 (Starting Occurrence). *The earliest (left-most) occurrence amongst all occurrences of a pattern.*

Definition 7 (LateBicluster). *A LateBicluster M_{IJ} is composed of a subset of genes (rows) $I \subseteq G$, a pattern P, and a collection J containing one subset of contiguous time points (columns) $J_i \subseteq T$ per gene $i \in I$. Each time frame J_i denotes an occurrence of P in the profile of gene (row) i in matrix M, which we denote by M_{iJ_i}. Thus, $P = M_{iJ_i}$ for all $i \in I$.*

For clarity, if M_{xJ_x} is the starting occurrence of P for a LateBicluster M_{IJ}, the occurrence M_{iJ_i} in M_{IJ} exhibits a time lag lag_i relative to M_{xJ_x}. Therefore a time point (column) $j_i \in J_i$ is given by $j_i = j_x + lag_i$.

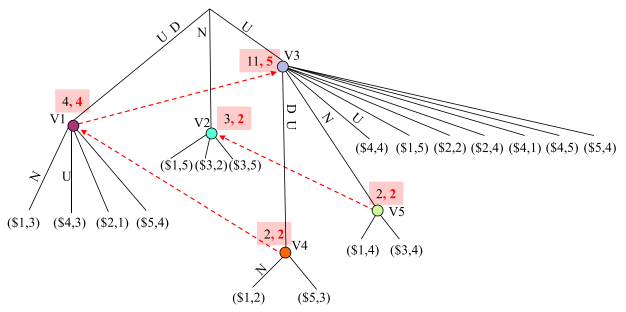
Definition 8 (Maximal LateBicluster). *A LateBicluster M_{IJ} is maximal if no rows can be added to I and no contiguous columns can be added to the left or right of J_l, for all $l \in I$, while maintaining the coherence property in Definition 7. A LateBicluster is maximal if it is row-maximal, left-maximal and right-maximal.*

When considering unbounded time lags, the range of possible lags is naturally upper bounded by the number of time points (columns). If we require a minimum number of time points Q_T per LateBicluster, then the time lag range is $[0, |T| - Q_T]$. In practice, there can be multiple instances of the pattern associated with a LateBicluster M_{IJ} in the expression profile of any gene in I. As a result, the number of instances over all LateBiclusters can be exponentially large. We exploit strategies to make reporting efficient, including allowing only one occurrence per gene (Definition 7).

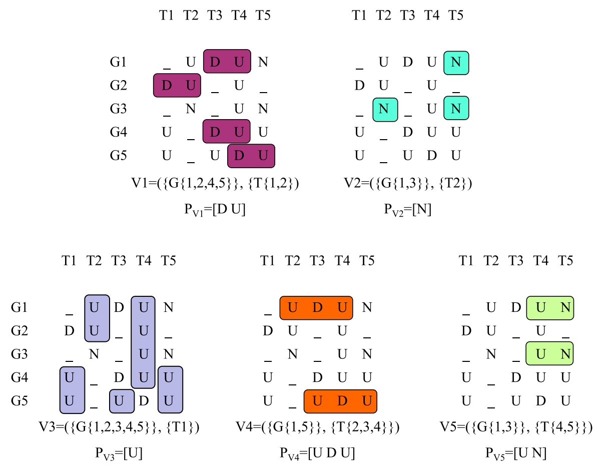
We propose an efficient algorithm enabling the identification of maximal LateBiclusters with unbounded time lags in a time series matrix. We discuss that reporting all maximal LateBiclusters is computationally infeasible in most cases of interest and propose heuristic approaches to select the LateBiclusters to report, achieving a polynomial time solution.

2.3.2 From Rows to Strings

When focusing on patterns potentially occurring with a time delay, similar patterns should match regardless of their starting time point (column). In this context, alphabet transformation becomes unnecessary and is therefore not performed. Consider that each row of the discretized matrix M is regarded as a string, corresponding to a single time series or temporal gene expression profile. Preprocessing techniques can be used to deal with missing values, namely by filtering the gene profiles where these occur or filling empty cells with predicted values. If missing values are still found at the string generation



(a) Internal nodes and LatePatterns in suffix tree $\mathcal{T}$.



(b) LatePattern occurrences per internal node in $\mathcal{T}$.

Fig. 5. Internal nodes in $\mathcal{T}$, LatePatterns and occurrences in M: (a) internal nodes in the suffix tree $\mathcal{T}$ built for the strings in Fig. 4; the pairs of values associated with each internal node denote the number of leaves and distinct genes (rows) found in its subtree; (b) occurrences of the LatePattern identified by each internal node in $\mathcal{T}$ highlighted in the matrix M.

stage, we split any string where they occur into multiple substrings, assuming the symbol denoting a missing value as the separator character. Each resulting substring receives the same gene (row) identifier of the original string. We note that string splitting can break longer patterns into several parts, but this can also occur when replacing missing values by predictions. Fig. 4 shows the strings obtained for an illustrative discretized matrix.

2.3.3 LatePatterns in Suffix Tree

A generalized suffix tree $\mathcal{T}$ can be built to find temporal patterns shared by multiple time series, or gene expression profiles, in matrix M (Fig. 5). LatePatterns are of particular interest, as each of them is guaranteed to be the longest pattern matching for as many occurrences, at least two, when extending the pattern to the right from a given starting time point. The fact that a LatePattern is right-maximal relative to its set of occurrences is an important property, given that we are not interested in reporting patterns that could be extended to the right while keeping their number of occurrences. Notably, there is a one-to-one relationship between LatePatterns in M and internal nodes in $\mathcal{T}$, which we will exploit further and that we present in Lemma 1. A proof of this lemma is provided in supplementary material, available on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCBB.2014.2312007).

Lemma 1. *Every LatePattern in M is identified by one internal node in $\mathcal{T}$, and every internal node in $\mathcal{T}$ identifies one LatePattern in M.*

2.3.4 LateBiclusters in Suffix Tree

We investigate the relationship between LatePatterns and LateBiclusters, and between LateBiclusters and nodes in $\mathcal{T}$. For clarity, we focus on LateBiclusters with at least two genes (rows). LateBiclusters with a single gene are trivial and uninteresting, and considering them would unnecessarily extend the proofs. In this context, we observe that a leaf node in $\mathcal{T}$ cannot identify a LateBicluster with at least two genes given that it denotes a single occurrence of a pattern, pertaining to only one gene (see leaf nodes in Fig. 5a).

We then observe that a maximal LateBicluster with at least two genes (rows) always has a LatePattern. The existence of a LatePattern does not imply, however, that any LateBicluster associated with it either satisfies the minimum number of (two) genes or the conditions for maximality. We present this relationship in Lemma 2 (proof in supplementary material available online). Knowing that only LatePatterns can be associated with maximal LateBiclusters with at least two genes (rows) in M is quite important, as it enables to narrow down the set of patterns of interest from all possible patterns to those that are LatePatterns (Definition 4).

Lemma 2. *A LatePattern is necessary but not sufficient for the maximality and genes (rows) quorum of a LateBicluster with at least two genes (rows).*

Furthermore, we know from Lemma 1, that every LatePattern is identified by a single internal node in $\mathcal{T}$. It follows that a LateBicluster with a given LatePattern in M that matches the pattern of a specific internal node in $\mathcal{T}$ must be identified by such node. We also note that, although there is only one internal node per LatePattern in $\mathcal{T}$, there can be several LateBiclusters with such LatePattern due to the potential existence of multiple occurrences of such pattern in the individual gene profiles (rows) in M. We formalize these relationships in Lemma 3 (proof in supplementary material, available online).

Lemma 3. *Every LateBicluster with a LatePattern in M is identified by one internal node in $\mathcal{T}$, and every internal node in $\mathcal{T}$ identifies at least one LateBicluster with a LatePattern in M.*

We have shown that each internal node in $\mathcal{T}$ identifies at least one LateBicluster with a LatePattern in M. However, we also know that the right-maximality of a LatePattern is not sufficient to determine the maximality of a LateBicluster. Moreover, the guarantee that the number of occurrences of a LatePattern is at least two does not ensure that any LateBicluster with such pattern will have at least two genes (rows). We now focus on determining maximality.

2.3.5 Row-Maximal LateBiclusters in Suffix Tree

According to Definition 8, a LateBicluster is maximal iff it is row-maximal, right-maximal and left-maximal. In other words, a LateBicluster is maximal if its occurrences are not fully contained in any other LateBicluster. Recall that every internal node v in $\mathcal{T}$ identifies a unique pattern and every occurrence of that pattern in M can be identified by the leaves in the subtree rooted at v. In addition, there can be multiple occurrences of the pattern per gene. Since a LateBicluster contains only one occurrence per gene (row), in order to identify a LateBicluster it is necessary to select a valid combination of occurrences from those in the subtree of v. Row-maximality is guaranteed as long as the LateBicluster generated for v contains one occurrence per distinct gene identifier found at the leaves in the subtree of v. This is true given that all occurrences of the LateBicluster pattern can be found labelling the leaves under v, and therefore such LateBicluster contains the maximum number of gene profiles (rows) where that particular pattern occurs (see Lemma 4).

Lemma 4. *A LateBicluster identified by an internal node v in $\mathcal{T}$ is row-maximal iff it contains one occurrence for every distinct gene labelling the leaves in the subtree rooted at v.*

Proof of Lemma 4. We split the statement into two:

1) If a LateBicluster B for internal node v has one occurrence per distinct gene found at the leaves in the subtree of v, then B is row-maximal.
2) If a LateBicluster B for internal node v is row-maximal, then B has one occurrence per distinct gene labelling the leaves in the subtree of v.

Proof of statement 1), by contraposition. Let B be a non-row-maximal LateBicluster in M. By Definition 7 there must exist at least one occurrence of the pattern of B in the profile of a gene in M that is not included in B, otherwise B would be row-maximal. By Lemma 3, B is identified by a single internal node v in $\mathcal{T}$, which denotes the pattern of B. By the properties of suffix trees, all occurrences of the pattern denoted by a node v, thus the pattern of B, can be found at the leaves under v [12]. As a result, the occurrence pertaining to the gene not included in B can also be found at the leaves in the subtree of v. It follows that B cannot contain an occurrence per gene in the subtree of v.

Proof of statement 2), by contraposition. Let v be an internal node in $\mathcal{T}$ identifying a LateBicluster B, such that B does not contain an occurrence for every gene labelling the leaves in the subtree of v. This means that an occurrence of the pattern of B can be found at the leaves in the subtree of v for at least one additional gene not in B. As a result, such gene could be added to the subset of genes of B while maintaining the pattern and the properties in the definition of a LateBicluster (Definition 7). If B can be extended by adding genes, it cannot be row-maximal. □

We note that, if the number of distinct genes is the same as the number of leaves in the subtree of v, then every leaf denotes an occurrence from a distinct gene and only one row-maximal LateBicluster exists. Otherwise, when the number of leaves is larger, several row-maximal occurrence combinations leading to row-maximal LateBiclusters can be computed.

2.3.6 Left/Right-Maximal LateBiclusters in Suffix Tree

Ensuring left- and right-maximality requires additional work, since the occurrences under an internal node v need to be matched and checked against those of other nodes in $\mathcal{T}$. In particular, we note that every valid extension of the LatePattern identified by v to the left is represented by an internal node in $\mathcal{T}$ from which v has an incoming suffix link. Likewise, any valid extension of the LatePattern denoted by v to the right is represented by an internal node that is a child of v in $\mathcal{T}$. As a result, internal nodes from which v has incoming suffix links, or of which v is a parent, need to be investigated in order to guarantee that no LateBicluster is generated from v with a set of occurrences that can be found in another LateBicluster whose pattern is an extension of the pattern of v.

Let $\mathcal{L}(v)$ and $\mathcal{G}(v)$ respectively denote the number of leaves and the number of distinct genes in the subtree rooted at node v in $\mathcal{T}$. In Lemma 5, we present the two cases in which the left- and right-maximality, and thus maximality, of a row-maximal LateBicluster in M identified by an internal node v in $\mathcal{T}$ is directly determined by constraints on these properties of v.

Lemma 5. *A row-maximal LateBicluster identified by an internal node v in $\mathcal{T}$ is maximal if v satisfies one of the following conditions:*

1) *It does not have incoming suffix links or internal children nodes.*
2) *It has incoming suffix links or internal children nodes only from nodes u such that $\mathcal{G}(u) < \mathcal{G}(v)$ is true for every u.*

Proof of Lemma 5. We prove the following: "If a row-maximal LateBicluster B is identified by an internal node v satisfying 1) or 2), then B is maximal." For the purpose of contraposition, let B be a row-maximal LateBicluster identified by an internal node v. Assume also that B is not left/right-maximal.

Proof for condition 1). Since B is not left- or right-maximal, then its pattern can be extended to the left or right while maintaining the set of occurrences in M. Since v is the node identifying B and B can be extended, there must be at least one node u from which v has incoming suffix links or of which v is a parent, and for which the row-maximal set of occurrences of B can be found in the leaves under u. As a result, condition 1) is false under the contrapositive assumption, thus condition 1) is sufficient for the maximality of B.

Proof for condition 2). Let B_{ext} be a LateBicluster identified from an internal node u that extends the pattern of B to the left or right with the same row-maximal set of occurrences. From Lemma 4, B and B_{ext} contain one occurrence per distinct gene labelling the leaves in the subtree of their corresponding internal nodes, respectively v and u. Note that the row-maximal sets of occurrences for u and v contain $\mathcal{G}(u)$ and $\mathcal{G}(v)$ distinct genes, respectively. Since the sets of occurrences of B_{ext} and B are the same, then $\mathcal{G}(u) = \mathcal{G}(v)$ must be true. As a result, condition 2) is false under the contrapositive assumption, thus condition 2) is sufficient for the maximality of B. □

If an internal node v in $\mathcal{T}$ does not have incoming suffix links or internal children nodes, then there are no valid extensions of the LatePattern denoted by v and therefore all

TABLE 1
Bit Array $colors$ Representation

G	1					2					3					4					5				
T	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
C	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
B	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

'G' denotes genes, 'T' time points, 'C' color bit values, and 'B' bit indexes. Bit array $colors(V2)$ for internal node V2 in the suffix tree from Fig. 5(a).

LateBiclusters identified by v are maximal. If any internal node u from which v has an incoming suffix link or of which v is a parent has less genes in its subtree than v, $\mathcal{G}(u) < \mathcal{G}(v)$, then any row-maximal LateBicluster generated from v will always contain one additional gene (row) and it will therefore be maximal on genes (rows) relative to any of the LateBiclusters with extended patterns.

The most difficult case arises, however, when v does not satisfy any of the conditions in Lemma 5. Specifically, if at least one node from which v has an incoming suffix link or of which v is a parent has the same number of distinct genes in its subtree as v. A straightforward solution would be to generate all possible combinations of occurrences and then filter them against the combinations of every incoming suffix link and internal child node. Unfortunately this is infeasible in most cases, as there can be $O(|T|^{|G|})$ combinations, leading to the same number of distinct LateBiclusters, per each of the $O(|G||T|)$ internal nodes in $\mathcal{T}$. Efficient reporting approaches can be exploited to address this problem, but they usually require one to disregard the definitions of LateBicluster and/or maximality. We postpone such discussion to a later section and proceed by describing a solution to ensure LateBicluster maximality for the case in which one still wants to report maximal LateBiclusters.

2.3.7 *Maximal LateBiclusters in Suffix Tree*

We note that all row-maximal LateBiclusters arising from a given internal node v have the same pattern and set of genes (rows). Considering that the goal in biclustering is to group entities in both dimensions rather than exhaustively enumerating the conditions leading to a grouping, it seems reasonable to report at most one row-maximal LateBicluster per internal node v. This also leads to more efficient solutions, given that it allows some freedom in filtering occurrences leading to non-maximal LateBiclusters if v does not satisfy the conditions stated in Lemma 5.

If v is in this situation, some row-maximal occurrence combinations may lead to non-maximal LateBiclusters. Take as an example the row-maximal LateBicluster defined by the occurrences of pattern N at time point $T5$ in the profiles of genes $G1$ and $G3$, found under node $V2$ in Fig. 5. This row-maximal LateBicluster is not left-maximal, given that node $V5$ defines another row-maximal LateBicluster with a larger pattern UN occurring in the profiles of the same genes $G1$ and $G3$, and spanning time points $T4$ and $T5$. However, there is one maximal LateBicluster identified by node $V2$, when considering time points $T2$ and $T5$ in genes $G1$ and $G3$, respectively.

We present an efficient heuristic approach, termed maximality update, that enables to filter all occurrences leading to non-maximal combinations for certain internal nodes. Consider a "problematic" internal node v in $\mathcal{T}$, together with nodes $u \in \mathcal{U}(v)$, where $\mathcal{U}(v)$ is the set of nodes from which v has incoming suffix links or of which v is a parent and such that $\mathcal{G}(u) = \mathcal{G}(v)$. The main idea of the maximality update for v is as follows: (1) assume v as invalid if every occurrence in the subtree of v can be found in the subtree of a node u; (2) for genes that have at least one occurrence in the subtree of v that is not found in the subtree of any node u, preserve only the occurrences not in the subtree of any node u (filter the remaining); for the other genes, keep the original occurrences.

For this purpose, we store a bit array with $|G||T|$ bits per node v, $colors(v)$, such that each bit denotes an occurrence that can potentially be used to compute maximal LateBiclusters. Consider that index $k \in \{0, \ldots, |G||T| - 1\}$, with k increasing from right (least significant) to left (most significant) in the bit representation. In $colors(v)$, each group of contiguous $|T|$ bits represents all potential starting positions $j \in T$ of occurrences of a pattern for a particular gene $i \in G$. Genes and time points underlying the occurrences are found in increasing order of their identifiers from left to right in the bit representation. Table 1 shows the representation of a bit array $colors$, together with gene and time point identifiers, and bit indexes.

We first compute $colors(v)$ for each node in $\mathcal{T}$ as follows. The bit in position k is set, $colors(v)[k] = 1$, whenever there is a gene and time point pair (i, j) labelling an occurrence in the subtree rooted at v such that $|T|(|G| - i + 1) - j = k$ (otherwise $colors(v)[k] = 0$). Table 1 shows $colors(V2)$ for node $V2$ in Fig. 5. We then update $colors(v)$ by clearing bits that can lead to non-maximal LateBiclusters (Fig. 6). We compute the union of all occurrences in the subtrees of the nodes $u \in \mathcal{U}(v)$ through bitwise *OR*ing of their $colors(u)$ arrays, denoted by $colors_{prob}(v)$ (1-8). When processing a bit array $colors(u)$ from a node u from which v has an incoming suffix link, we use a copy of $colors(u)$ that has been right-shifted by one position before the *OR*, so as to align the starting points with the bit array of v (3-4). Next, we seek for differences between the occurrences under nodes in $\mathcal{U}(v)$ and those under v by computing $colors_{diff}(v)$, a bitwise XOR between $colors_{prob}(v)$ and $colors(v)$ (9). If $colors_{diff}(v) = 0$, meaning that every bit in $colors_{diff}(v)$ is clear, then every occurrence in the subtree of v is present in the subtree of at least one node in $\mathcal{U}(v)$ and we clear the updated colors for v (10-11). Otherwise, we transform $colors_{diff}(v)$ into a bit mask $colors_{mask}(v)$ that will be used in the update of $colors(v)$ (12-24). In this transformation, we process $colors_{diff}(v)$ as follows. For each gene i, if all bits corresponding to gene i in $colors_{diff}(v)$ are clear (value 0), we set all such bits and otherwise we skip them. The interval of bits for a given gene i includes all indexes k such that $k = |T|(|G| - i + 1) - p$ for all $p \in \{1, \ldots, |T|\}$. Finally, we compute the updated bit array $colors_u(v)$ by performing a bitwise AND between $colors(v)$ and $colors_{mask}(v)$ (25).

We observe that the described maximality update can eliminate combinations leading to the identification of maximal LateBiclusters whose occurrences are not fully enclosed in any other maximal LateBicluster. Specifically, this can happen for v if every occurrence in its subtree is also in the subtree of some internal node $u \in \mathcal{U}(v)$ with $\mathcal{G}(u) = \mathcal{G}(v)$

```
Require: $colors(v) \neq null \wedge \mathcal{U}(v) \neq \emptyset$
1: $colors_{prob}(v) \leftarrow 0$
2: for all $u \in \mathcal{U}(v)$ do
3:   if $v = slink(u)$ then
4:     $colors_{prob}(v) \leftarrow colors_{prob}(v) \vee (colors(u) \gg 1)$
5:   else
6:     $colors_{prob}(v) \leftarrow colors_{prob}(v) \vee colors(u)$
7:   end if
8: end for
9: $colors_{diff}(v) \leftarrow colors(v) \oplus colors_{prob}(v)$
10: if $colors_{diff}(v) = 0$ then
11:   $colors_u(v) \leftarrow 0$
12: else
13:   $colors_{mask}(v) \leftarrow colors_{diff}(v)$
14:   $upper_{bound} \leftarrow |G||T|$
15:   $pos \leftarrow$ prevSetBitPos $(colors_{diff}(v), upper_{bound} - 1)$
16:   while $pos \geq 0$ do
17:     $pos \leftarrow \lfloor k/|T| \rfloor \times |T|$
18:     setBits $(colors_{mask}(v), pos + |T|, upper_{bound})$
19:     $upper_{bound} \leftarrow pos$
20:     $pos \leftarrow$ prevSetBitPos $(colors_{diff}(v), pos - 1)$
21:   end while
22:   if $upper_{bound} > 0$ then
23:     setBits $(colors_{mask}(v), 0, upper_{bound})$
24:   end if
25:   $colors_u(v) \leftarrow colors(v) \wedge colors_{mask}(v)$
26: end if
```

Fig. 6. Maximality update for a “problematic” node v, with non-empty $\mathcal{U}(v)$. Every internal node $u \in \mathcal{U}(v)$ has $\mathcal{G}(u) = \mathcal{G}(v)$ and is either a child of v, or its suffix link points to v. Function `prevSetBitPos (a,p)` returns the index of the first set bit in array a to the right of index p (see Table 1). Procedure `setBits (a,p_i,p_f)` sets all bits of a in the range of indexes $[p_i, p_f[$. Logical connectives $\vee$, $\wedge$ and $\oplus$ are used to represent the bitwise operators OR, AND and XOR, respectively.

and from which v has an incoming suffix link or of which v is a parent. Then the maximality update filters all occurrences for v and v can no longer identify maximal LateBiclusters.

Counteracting such behavior would require a solution for choosing a row-maximal set of occurrences from the subtree of v that cannot be found in the subtree of one of the nodes u. Doing this efficiently is a challenge and we postpone a more thorough analysis to future work. We develop our heuristic approach for the case in which at least one occurrence in the subtree of v is not found in the subtree of any of the nodes in $\mathcal{U}(v)$. For such a case, the update procedure guarantees the identification of at least one maximal LateBicluster for v, as formalized by Lemma 6.

Lemma 6. *The maximality update of $colors(v)$ for an internal node v in $\mathcal{T}$ with at least one internal node u for which $\mathcal{G}(u) = \mathcal{G}(v)$ among those nodes from which v has incoming suffix links or of which v is a parent, preserves at least one row-maximal combination of occurrences leading to the identification of a maximal LateBicluster iff at least one occurrence for at least one gene is present at the leaves under v but not at the leaves under any of the nodes u.*

Proof. We split the Lemma into two statements:

1) If $colors_u(v)$ obtained by maximality update of $colors(v)$ contains at least one row-maximal occurrence combination, then at least one occurrence for at least one gene is found at the leaves under v but not at the leaves of any node u.

2) If at least one occurrence for at least one gene is at the leaves under v but not at the leaves under any node u, then the maximality update of $colors(v)$ preserves at least one row-maximal occurrence combination for v.

Let v be a “problematic” internal node. This means that there is at least one internal node u with $\mathcal{G}(u) = \mathcal{G}(v)$ among all the internal nodes of which v is a parent or from which v has an incoming suffix link.

Proof of 1), by contraposition. Assume that every occurrence at the leaves under v is also found at the leaves under that one node u. Since $colors_{prob}(v)$ contains the union of occurrences at the leaves under nodes u, then every bit in $colors_{prob}(v)$ that corresponds to an occurrence at the leaves under v is set. This means that $colors_{prob}(v)$ and $colors(v)$ are equal and therefore the difference between the two, denoted by $colors_{diff}(v)$, is zero (all bits in $colors_{diff}(v)$ are clear). In this case, no row-maximal occurrence combination is preserved, thus no maximal LateBicluster can be identified from v. Therefore, 1) must be true.

Proof of 2), by contraposition. Assume that $colors_u(v)$ does not contain any row-maximal occurrence combination. In other words, it does not contain at least one set bit per gene found at the leaves under v. This can only happen if all bits in $colors_{diff}(v)$ are clear and therefore $colors_{prob}(v)$ and $colors(v)$ are the same. If this was not the case and $colors_{diff}(v)$ had at least one set bit, then the updated array $colors_u(v)$ would contain at least one set bit per gene at the leaves under v and thus an occurrence combination denoting a maximal LateBicluster (see below), contrary to our initial assumption. Therefore, 2) is true.

To support the proof of 2) we show, by construction, that if at least one occurrence at the leaves under v is not at the leaves of some node u, and therefore $colors_{diff}(v)$ has at least one set bit, then at least one bit is set for every gene at the leaves under v in $colors_u(v)$ after the update. The transformation of $colors_{diff}(v)$ into a bit mask $colors_{mask}(v)$ is such that: (i) genes with at least one set bit in $colors_{diff}(v)$ are left unchanged in $colors_{mask}(v)$; (ii) genes with all bits clear in $colors_{diff}(v)$ have all their bits set in $colors_{mask}(v)$. In such a case, the update $colors_u(v)$ obtained by computing the bitwise AND between $colors_{mask}(v)$ and $colors(v)$ will result in the following: (i) for genes with at least one set bit in $colors_{diff}(v)$, there will be at least one set bit in $colors_u(v)$, given that any set bit in $colors_{diff}(v)$ corresponds to one occurrence that is at the leaves under v but not at the leaves under any of the challenging nodes u, meaning that the bit is set in both $colors_{diff}(v)$ and $colors(v)$ and therefore the bit in the result of the AND operation is also one; (ii) for genes with all bits clear in $colors_{diff}(v)$, the corresponding bits in $colors_{mask}(v)$ are all set, meaning that the values of the bits in the result of the AND operation will be the same as those in the original $colors(v)$. This means that, if there is at least one bit set in $colors_{diff}(v)$, there is at least one set bit for every gene at the leaves under v in $colors_u(v)$ after the update. □

2.3.8 Maximal LateBicluster Node Identification

Below we present Theorem 1, the main result leading to an efficient solution to identify nodes in $\mathcal{T}$, together with a set of valid occurrences, which denote at least one maximal LateBicluster in M.

Theorem 1. *An internal node v in $\mathcal{T}$ identifies at least one maximal LateBicluster with at least two genes if it satisfies $\mathcal{G}(v) \geq 2$, together with one of the following conditions:*

1) *node v does not have incoming suffix links or internal children nodes;*
2) *for every node u from which v has an incoming suffix link or of which v is a parent, $\mathcal{G}(u) < \mathcal{G}(v)$ is true;*
3) *$\mathcal{G}(u) = \mathcal{G}(v)$ holds for at least one node u from which v has an incoming suffix link or of which v is a parent; and $colors_u(v)$, obtained upon maximality update of $colors(v)$, has at least one set bit for at least two genes.*

Proof. Let v be an internal node in $\mathcal{T}$ for which $\mathcal{G}(v) \geq 2$ is true. First we show that v identifies at least one LateBicluster. By Lemmas 1 and 3, node v identifies at least one LateBicluster (Definition 7) with a LatePattern (right-maximal pattern with at least two occurrences—Definition 4) in the time series matrix M. By Lemma 2, a LatePattern is a necessary condition for the maximality of a LateBicluster. We then show that v identifies at least one maximal LateBicluster with at least two genes. According to Lemma 4, a LateBicluster obtained for node v is row-maximal iff it contains one occurrence per gene labelling the leaves in the subtree rooted at v. If v satisfies conditions 1) or 2), all the occurrences under v are considered and therefore it is always possible to select a combination that identifies a row-maximal LateBicluster. Additionally, by Lemma 6 any row-maximal LateBicluster generated for a node v satisfying conditions 1) or 2) is also left- and right-maximal, and therefore maximal. Since $\mathcal{G}(v) \geq 2$ is true, every maximal LateBicluster generated for node v has at least two genes. If v does not satisfy condition 1) or 2), then whether v identifies maximal LateBiclusters or not is dependent on the output of the maximality update procedure. By Lemma 6 the maximality update is guaranteed to either filter all occurrences or preserve at least one valid combination of occurrences leading to the identification of a maximal LateBicluster. The latter case is guaranteed by condition 3), which states that some occurrences are still present after the update. Condition 3) further specifies that there is at least one set bit for at least two genes in the updated bit array $colors_u(v)$ and therefore there are occurrences for at least two distinct genes. It follows that at least one maximal LateBicluster with at least two genes can be identified from a node v satisfying condition 3). □

2.4 Reporting Strategies

When focusing on aligned patterns (CCC-Bicluster), there can be at most one occurrence of the pattern per gene. As a result, the number of maximal CCC-Biclusters is $O(|G||T|)$. Since the information to report per CCC-Bicluster is $O(|G|)$, the time necessary for reporting all maximal CCC-Biclusters is then $O(|G|^2|T|)$.

In the time-lagged setting, however, it is possible to find multiple occurrences of a pattern within the profile of a given gene at different starting time points. In such a case, the number of maximal LateBiclusters corresponds to the number of all possible combinations of one occurrence of the pattern per gene, for every distinct gene labelling the leaves in the subtree of each internal node of interest. In the worst case, the number of maximal LateBiclusters that can be obtained for a given temporal pattern is therefore $O(|T|^{|G|})$. The fact that the number of maximal LateBiclusters can grow exponentially with the number of genes often makes exhaustive enumeration infeasible. Take the rather small (5×5) illustrative matrix in this section as an example (Figs. 4 and 5), for which we obtain 40 maximal LateBiclusters with at least two genes (rows) (Fig. 7). We discuss alternatives for reporting the information under each valid internal node v in $\mathcal{T}$, such that reporting becomes tractable.

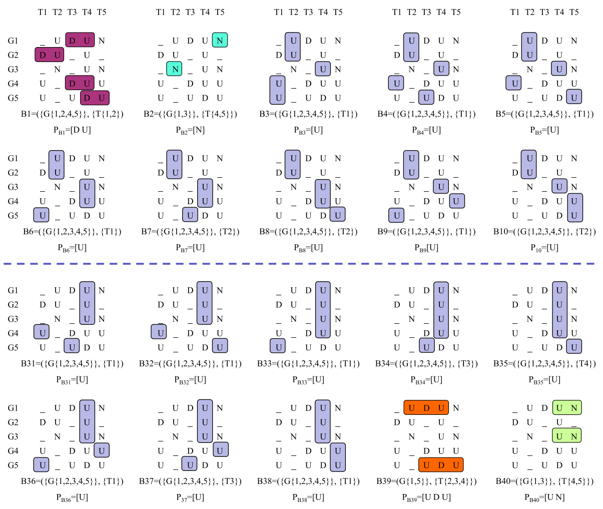
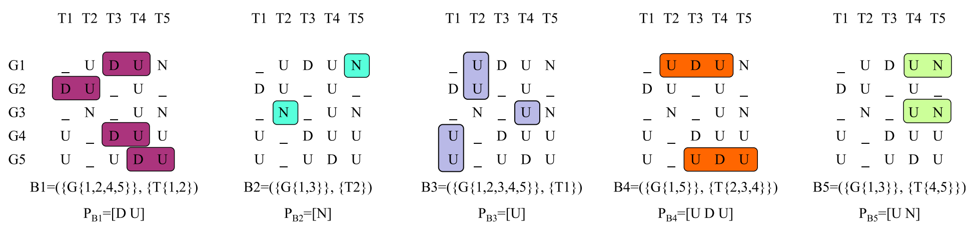

B1=({G{1,2,4,5}}, {T{1,2}}) P_B1=[D U]
B2=({G{1,3}}, {T{4,5}}) P_B2=[N]
B3=({G{1,2,3,4,5}}, {T1}) P_B3=[U]
B4=({G{1,2,3,4,5}}, {T1}) P_B4=[U]
B5=({G{1,2,3,4,5}}, {T1}) P_B5=[U]
B6=({G{1,2,3,4,5}}, {T1}) P_B6=[U]
B7=({G{1,2,3,4,5}}, {T2}) P_B7=[U]
B8=({G{1,2,3,4,5}}, {T2}) P_B8=[U]
B9=({G{1,2,3,4,5}}, {T1}) P_B9=[U]
B10=({G{1,2,3,4,5}}, {T2}) P_10=[U]
B31=({G{1,2,3,4,5}}, {T1}) P_B31=[U]
B32=({G{1,2,3,4,5}}, {T1}) P_B32=[U]
B33=({G{1,2,3,4,5}}, {T1}) P_B33=[U]
B34=({G{1,2,3,4,5}}, {T3}) P_B34=[U]
B35=({G{1,2,3,4,5}}, {T4}) P_B35=[U]
B36=({G{1,2,3,4,5}}, {T1}) P_B36=[U]
B37=({G{1,2,3,4,5}}, {T3}) P_37=[U]
B38=({G{1,2,3,4,5}}, {T1}) P_B38=[U]
B39=(G{1,5}}, {T{2,3,4}}) P_B39=[U D U]
B40=({G{1,3}}, {T{4,5}}) P_B40=[U N]

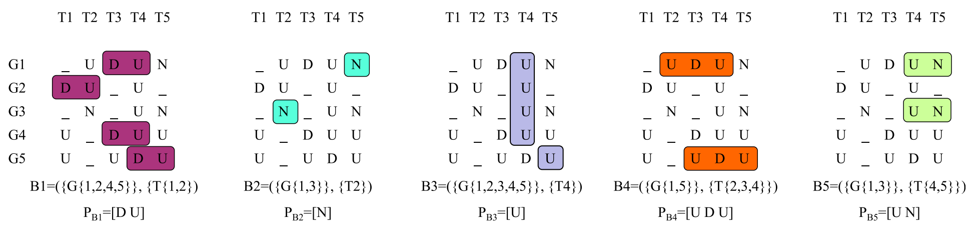
Fig. 7. Exhaustive reporting of 40 maximal LateBiclusters for nodes $V1$-$V5$ in the suffix tree from Fig. 5a. Each individual matrix shows the pattern occurrences of a LateBicluster. A single LateBicluster is obtained for $V1$, $V2$, $V4$, $V5$. The remaining 36 LateBiclusters are generated for node $V3$ (in purple, only 16 are shown).

2.4.1 LatePatterns and Occurrences

If we disregard the definition of LateBicluster and its maximality, we can accomplish the identification of internal nodes yielding LatePatterns as described in Sections 2.3.1 to 2.3.3. Each internal node satisfies Lemma 1 and therefore identifies a LatePattern, which essentially guarantees the right-maximality of the pattern relative to its set of occurrences. Left-maximality of the LatePattern can additionally be enforced by checking the internal node against other nodes from which it has incoming suffix links. This is similar to what is done in CCC-Biclustering, but in this case taking into account that leaves identify occurrences rather than genes. Finally, for each internal node, we report the LatePattern together with information found at the leaves in the subtree of such node. We consider two reporting options. The first outputs only the genes labelling the leaves in the



(a) Left-most occurrence heuristic approach.



(b) Most frequent starting time point heuristic approach.

Fig. 8. Heuristic reporting of maximal LateBiclusters $B1$ to $B5$ for nodes $V1$ to $V5$ in the suffix tree from Fig. 5a. (a) earliest (left-most) pattern occurrence per gene; and (b) earliest starting time point per gene larger than the most frequent starting time point among all occurrences of the pattern (otherwise, the closest starting point smaller than the most frequent one): most frequent starting points, after maximality update, are $T3$ for $V1/B1$, $T2$ for $V2/B2$, $T4$ for $V3/B3$, $T2$ for $V4/B4$, and $T4$ for $V5/B5$.

subtree of the internal node, provided that one is not interested in where the pattern occurs in the profile of each gene. The second reports every gene together with the starting points of all occurrences of the pattern in the profile of each of such genes (similar to Fig. 5b).

2.4.2 Maximal LateBiclusters

An alternative reporting procedure consists in identifying a single maximal LateBicluster per internal node satisfying the conditions in Theorem 1. When identifying maximal LateBiclusters as described in Sections 2.3.1 to 2.3.7, maximality can sometimes be automatically determined if the node satisfies Lemma 5. Alternatively, a new set of occurrences containing only instances leading to the identification of maximal LateBiclusters is computed according to Lemma 6. Since this new set may still contain multiple occurrences per gene, we further need to select a single occurrence per distinct gene. We propose two different strategies for choosing a single row-maximal combination of occurrences for a given node.

Left-most occurrence per gene. Consider the left-most time point among the starting positions of all the occurrences of the pattern in the subtree rooted at a valid node v in $\mathcal{T}$ as the starting point of a cascade of delayed activations/inhibitions. In this context, it seems reasonable to store for each gene only the starting time point of the left-most occurrence of the pattern. Fig. 8a shows the output expected when applying this heuristic approach to the LateBiclustering output for the matrix in Fig. 4.

Most frequent starting time point. Consider the most frequent time point among the starting positions of all the occurrences of the pattern in the subtree rooted at a valid internal node v in $\mathcal{T}$, denoted as p, as the starting time point of a cascade of delayed activations/inhibitions. In this context, we propose the following heuristic approach: for each gene, we store the starting point of the first occurrence starting at or after p; whenever such occurrence does not exist, we store the starting point of the closest occurrence before p. Fig. 8b shows the output expected when applying this heuristic approach to the LateBiclustering output for the matrix in Fig. 4.

2.5 Complexity

In this section we present the complexity analysis of the identification of nodes denoting maximal LateBiclusters (Section 2.3) and of different strategies for reporting the results (Section 2.4).

2.5.1 Suffix Tree and LatePatterns

To identify nodes denoting LatePatterns, we first build a generalized suffix tree $\mathcal{T}$ for the set of strings, which can be done in $O(|G||T|)$ time using Ukkonen's algorithm [12], [13]. If left-maximality of LatePatterns relative to its occurrences is to be enforced, then an additional calculation of the number of leaves in the subtree of each node v, previously denoted by $L(v)$, is needed. This can be done for all nodes with a single traversal of $\mathcal{T}$, requiring constant time processing at each node, and therefore taking $O(|G||T|)$ time. The number of leaves $L(v)$ is also used when determining maximality of LateBiclusters (see Lemma 5).

2.5.2 Suffix Tree and Maximal LateBiclusters

The identification of nodes denoting LateBiclusters (maximal or not) described in Section 2.3.4 does not require any further processing of $\mathcal{T}$. However, if the goal is to report only those LateBiclusters that are maximal, then additional operations are necessary.

For exhaustive enumeration, the most immediate solution to ensure maximality would be to check all row-maximal occurrence combinations generated for a given internal node against those generated for nodes in the conditions of challenging maximality. Given that the number of maximal

LateBiclusters can be exponential on the number of genes, reporting is often infeasible. For this reason, we do not discuss maximality checking for exhaustive enumeration.

In the case of heuristic enumeration, maximality is ensured as described in Sections 2.3.5 to 2.3.7. Specifically, maximality can be automatically determined if the node satisfies Lemma 5, provided that we know the number $\mathcal{G}(v)$ of distinct genes labelling the leaves under each node v in $\mathcal{T}$. This can be obtained in $O(|G||T|)$ time using the solution of Chi and Hui for the color set size problem [12], [14]. It requires an $O(|G||T|)$ time preprocessing of $\mathcal{T}$ to enable constant time lowest common ancestor queries, which has been first proposed by Schieber and Vishkin [15] and Gusfield [12].

In the case in which maximality cannot be directly confirmed, the maximality update procedure from Section 2.3.7 is applied. We analyze its complexity below. Initializing the bit arrays $colors(v)$ for all nodes v in $\mathcal{T}$ takes $O(|G||T|\lceil\frac{|G||T|}{|w|}\rceil)$ time by ORing the bit arrays using a single traversal of $\mathcal{T}$, where $|w|$ is the number of bits per computer word. This is so given that there are $O(|G||T|)$ nodes in $\mathcal{T}$, a single bitwise OR operation is performed per node (at its unique parent) and that the cost of such operation is $O(\lceil\frac{|G||T|}{|w|}\rceil)$. It is reasonable to assume that a bitwise operation involving two bit arrays of the size of a computer word can be performed in constant time.

Updating the bit arrays $colors(v)$ also takes $O(|G||T|\lceil\frac{|G||T|}{|w|}\rceil)$. In the worst case, all internal nodes need to be investigated and the complexity of the update is as follows. First, computing the union $colors_{prob}(v)$ of the $colors$ arrays of all nodes that can challenge the maximality of LateBiclusters identified by a node v requires bitwise OR operations and can be done for all nodes using a single traversal of $\mathcal{T}$. The total number of operations is $O(|G||T|)$, including an OR per node in $\mathcal{T}$ (with the $colors$ array of the corresponding parent) and a right-shift and an OR per internal node with an outgoing suffix link in $\mathcal{T}$ (with the $colors$ array of the node to which the suffix link points). Assuming that the right shift can also be performed in time linear on the number of bit words in $colors$, computing the union can therefore be done in $O(|G||T|\lceil\frac{|G||T|}{|w|}\rceil)$ time. Second, computing the difference $colors_{diff}(v)$ for every internal node v in $\mathcal{T}$ is also $O(|G||T|\lceil\frac{|G||T|}{|w|}\rceil)$, given that it requires a single traversal of $\mathcal{T}$ and a bitwise XOR operation per node v. Third, the transformation of $colors_{diff}(v)$ into a bit mask requires iterating through set bits and at most one per gene. Since we actually do not need to know the position of the set bits, the transformation can be done by checking and manipulating entire and/or partial words in the range pertaining to each gene. In the worst case, there are no set bits and it is necessary to process the whole range of words (if there are more than one). After this scanning, if no set bit is found, all bits in the range need to be set. Retrieving and setting whole words can be performed in constant time using AND and OR operations and an auxiliary bit mask. We also assume that a bit mask can be created in constant time to enable the retrieval and setting of bits for partial words, considering that most modern CPUs can shift a word by an arbitrary number of positions in one cycle using barrel shifting. Each (partial) word is processed at most twice, one for retrieving and another for setting, the latter only if the condition above is satisfied. The total number of (partial) words to process is given by the maximum between the number of genes and the number of words in a bit array $colors$ representation, thus $O(\max\{|G|, \lceil\frac{|G||T|}{|w|}\rceil\})$. Therefore, the maximality update procedure can be performed for all nodes in $O(|G||T|\max\{|G|, \lceil\frac{|G||T|}{|w|}\rceil\})$ time.

2.5.3 Reporting LatePatterns and Occurrences

Recall that we considered two reporting options for LatePatterns and their occurrences in Section 2.4. We also mentioned the choice between reporting all LatePatterns or only those that are also left-maximal relative to their sets of occurrences. However, this option does not have an impact in the asymptotical complexity of the solution, given that ensuring left-maximality only requires the computation of the number of leaves $L(v)$ for all nodes and an additional traversal of $\mathcal{T}$ in order to mark invalid nodes. Both operations can be done in $O(|G||T|)$.

The first reporting option is to output the gene identifier for each of the $O(|G|)$ genes in the subtree of each valid internal node in $\mathcal{T}$, without specifying the starting time points of all the $O(|T|)$ occurrences of the pattern. This type of reporting takes $O(|G|^2|T|)$. The second option is to report the gene identifier for each of the $O(|G|)$ genes in the subtree together with the starting time points of all occurrences of the pattern in each gene expression profile. Reporting this information takes $O(|G|^2|T|^2)$ time.

2.5.4 Reporting Maximal LateBiclusters

Exhaustive enumeration of maximal LateBiclusters takes $O(|G|^2|T|^{1+|G|})$ time in the worst case, considering that: (i) there can be $O(|T|^{|G|})$ maximal LateBiclusters to report per each of the $O(|G||T|)$ potentially valid internal nodes in $\mathcal{T}$; (ii) for each maximal LateBicluster with time lags, we have to report $O(|G|)$ genes together with the starting point of the pattern occurrences for each gene. This combinatorial explosion is practically challenging and often infeasible (Fig. 7). To address this issue, we introduced two heuristic approaches that report a single maximal LateBicluster per internal node in $\mathcal{T}$ satisfying the conditions in Theorem 1 (see Section 2.4). One selects the left-most starting time point per gene. The other chooses the starting time point closest to the most frequent starting time point per gene. Using either of them the reporting step takes $O(|G|^2|T|)$ time to generate a maximal LateBicluster with $O(|G|)$ genes, together with a single time point per gene, for each of the $O(|G||T|)$ potentially valid internal nodes in $\mathcal{T}$.

3 Results

In order to show the usefulness of LateBiclustering, we applied the algorithm to a real expression time series

TABLE 2
Maximal LateBiclusters with at Least 10 Genes and Four Time Points, Obtained Using the Left-Most Occurrence Heuristic

Rank	ID	Pattern	#Genes	#Time Points	Starting Time Points	#Highly Sig. Terms	Best p-value	Annotation
11	459	$DNNUU$	156	6	$\{5', 10', 15'\}$	31	1.16×10^{-13}	ribosome biogenesis
17	523	$DDNN$	283	5	$\{5', 10', 15', 20'\}$	11	1.67×10^{-8}	ribosome biogenesis
24	270	$NNND$	383	5	$\{5', 10', 15', 20'\}$	4	2.62×10^{-4}	ATP synthesis
26	277	$NNNDD$	102	6	$\{5', 10', 15'\}$	3	4.81×10^{-4}	oxidoreductase activity
29	441	$DNUND$	116	6	$\{5', 10', 15'\}$	3	1.60×10^{-4}	DNA-directed RNA polymerase II
33	10	$UUNND$	49	7	$\{5', 10'\}$	2	1.62×10^{-3}	glutathione metabolic process
34	63	$UNNND$	266	6	$\{5', 10', 15'\}$	2	4.56×10^{-3}	oxidation-reduction process
37	78	$UNNDDN$	104	7	$\{5', 10'\}$	2	3.10×10^{-3}	carbohydrate catabolic process
42	376	$DUNN$	246	5	$\{5', 10', 15', 20'\}$	2	4.00×10^{-3}	response to stimulus
50	102	$UNDNDN$	20	7	$\{5', 10'\}$	1	9.19×10^{-3}	protein kinase regulator activity

We show 10 LateBiclusters with interesting patterns and annotations, sorted by number of highly significant terms (p-value < 0.01: hypergeometric distribution, term-for-term calculation, Bonferroni correction).

dataset. We used data from Gasch et al. [16], concerning *Saccharomyces cerevisiae*'s response to heat shock. This data set comprises expression levels of 6,142 genes measured at eight distinct time points (5′, 10′, 15′, 20′, 25′, 30′, 40′, 60′ and 80′) for over an hour of exposure to 37° C. Similar to previous analyses of this kind [6], we first filtered all genes with missing values and normalized the expression levels per gene to zero mean and unit standard deviation. We also discretized the preprocessed matrix using a technique based on transitions between time points [6], [9].

3.1 Exhaustive versus Heuristic Reporting

We applied LateBiclustering using exhaustive enumeration (Section 2.4), and also the left-most time point heuristic reporting. Our tests were performed in a machine with an Intel®i7-3632QM CPU and 8 GB of RAM running Windows 8 64-bit, which can be considered a reasonably accessible user setting in a modern biology lab. To provide a real-world estimate, we included and ran the algorithms in BiGGEsTS [17], a free software tool for biclustering analysis of time series gene expression data. The exhaustive version rapidly exceeded the amount of memory in the system and aborted computation, while the heuristic finished within 1 minute using less than 1 GB of RAM.

3.2 LateBiclustering Heuristic Results

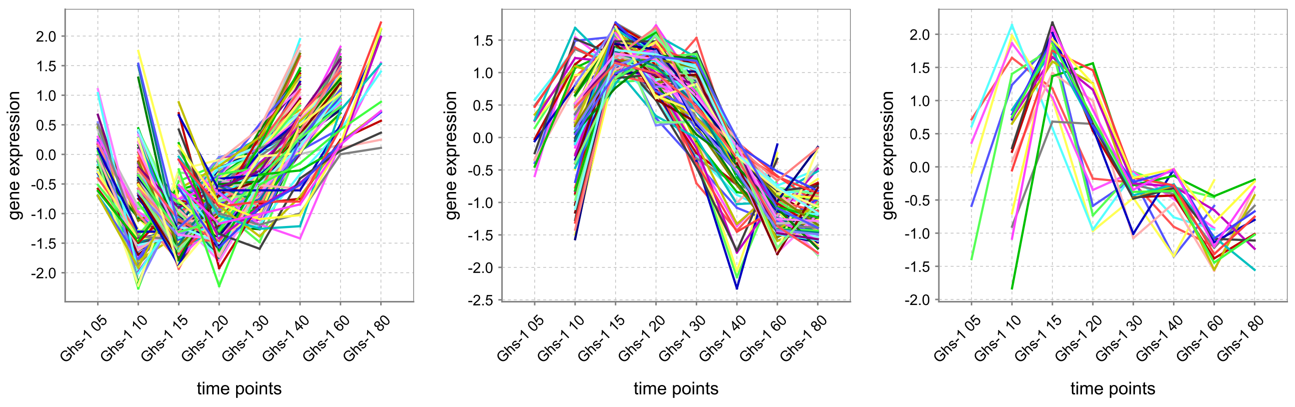
We provide an overview of the results obtained using LateBiclustering with unbounded time lags, combined with the left-most time point heuristic reporting. We restricted the search to LateBiclusters with at least 10 genes and four time points, which are reasonably sized and potentially more interesting from a biological perspective. The algorithm delivered 542 LateBiclusters.

3.2.1 Statistical Significance of Functional Annotations

We assessed the enrichment of functional annotations of the genes in each LateBicluster by computing the statistical overrepresentation of Gene Ontology (GO) terms [18]. For this purpose, we used the Ontologizer package [19], together with ontology and annotation files downloaded from the GO repository on May 20, 2013. We calculated a p-value based on the hypergeometric distribution and term-for-term calculation, and further applied a Bonferroni correction for multiple testing [19]. For approximately 12 percent of the LateBiclusters, there was at least one highly significant GO term (corrected p-value < 0.01). Table 2 presents 10 LateBiclusters with interesting patterns and enriched functions, sorted by number of highly significant GO terms. Alternative sorting criteria can be used, such as pattern length or p-value [17]. Each row corresponds to a given LateBicluster and the different columns contain the following information: 'Rank', the rank in the overall sorted list; 'Bicluster ID', a sequential bicluster identifier, '#Genes' the number of genes, '#Time Points' the number of time points, 'Starting Time Points' the starting time points of the different instances of the pattern, and '#Highly Sig. Terms' the number of highly significant GO terms. The last column, 'Best p-value', shows the best corrected p-value for any term annotated with the genes in the LateBicluster. As a result of the discretization denoting variations between time points, the number of time points is one unit larger than the pattern length.

3.2.2 Biological Relevance and Statistical Significance

For LateBicluster 459, "ribosome biogenesis" and "rRNA processing" were among the most enriched GO terms. The corresponding expression profile shows a sharp decrease during the first 5 minutes (Fig. 9a), consistent with the inhibition of ribosome and rRNA synthesis observed in yeast cells as part of the response to heat stress [16]. LateBicluster 78 was associated with "carbohydrate catabolic process" and "trehalose metabolic process". Heat shock reportedly activates energy consuming defense mechanisms and the yeast cell seeks alternative carbon sources, namely by production of ATP through glycolysis and trehalose synthesis [16]. The increased activity of these pathways is reflected by the pattern of this LateBicluster (Fig. 9b). Prior analysis of heat shock time series data using CCC-Biclustering revealed CCC-Biclusters with similar patterns and annotations (see CCC-Biclusters 124, and 27 or 14 in [6]). However, due to the inherent temporal alignment restriction, those CCC-Biclusters did not include genes exhibiting a similar but slightly delayed response, which could be involved in the same biological response. By grouping delayed responses together, LateBiclustering further delivers a more manageable number of biclusters: CCC-Biclustering reported 924 CCC-Biclusters when applied to the data in this work (under the same constraints). This is a desirable property if computational analysis is to be followed by manual inspection.



(a) Late Bicluster 459 (b) Late Bicluster 78 (c) Late Bicluster 102

Fig. 9. Expression profiles of genes in LateBiclusters. 'Ghs-1' denotes the sample (Gasch heat shock 1).

Statistical significance suggests potential bicluster functions. These are indicative of bicluster quality, but do not constitute a necessary condition for biological relevance. In addition, most enrichment tests are suitable for the analysis of large gene lists [20]. Even if parent-child relationships between GO terms are considered [19], many associated terms are assessed independently. This fragmentation of the GO term space only augments the limitations inherent to the analysis of small samples. As a result, small biclusters often yield few or no statistically significant functions.

We analyze LateBicluster 102 in greater detail. The 20 genes in this LateBicluster were enriched only for "protein kinase regulator activity". Their pattern denotes a drastic increase in expression level within the first minutes of exposure to heat (Fig. 9c). Broadly, this behavior is consistent with the role of kinases in the activation of signal transduction cascades that mediate several specific stress-related responses in the subsequent time points. The LateBicluster exhibits two sets of genes with distinct activation points. The earliest reaches an expression peak at 10′ and is composed of six genes, of which four yield functional annotations on the Saccharomyces Genome Database (SGD) [21]. Two of them are triggers of cell wall integrity signalling and plasma membrane reorganization (*CKB1*, *APS2*). In addition, there is a regulator of oxidation reduction homeostasis and stress-related vacuole functions, as well as growth factor signalling (*TRX1*). Finally, the group includes a mediator of energy generation through glycolysis (*GPM1*). The second set, with 14 genes, has its peak at 15′ and contains additional key players in cellular defense mechanisms against heat stress. For instance, *LRE1* and *YPK1* are relevant to the control of pathways such as the Pkc1-MAPK involved in confering resistance of the cell wall structure [21], [22]. Notably, *Ckb1p* (first set) has been suggested as a regulator of *LRE1* (second set), which can be a potential explanation for the delay between the activation of the corresponding genes. Four genes in the set, *ASH1*, *CLN3*, *DHH1* and *PCL5*, encode inhibitors of the cell cycle, which is likely related to the growth arrest experimented by the yeast cell following exposure to heat shock [16]. Strikingly, *ASH1* is regulated by *Ace2p*, whose phosphorylation promoted by *Ckb1p* (first set) delays the M/G1 transition [21]. Two other genes, *FMP37* and *TDH2*, have roles in glycolysis. Specifically, *Tdh2p* catalyzes a reaction within the conversion to pyruvate, and *Fmp37p* mediates mitochondrial pyruvate uptake [21]. Interestingly, the existence of a sudden expression increase observed for *FMP37* and *TDH2* between 10′ and 15′, following a similar behavior of *GPM1* (first set) between 5′ and 10′, is consistent with the participation of the latter at an earlier stage of glycolysis (glucose breakdown). The second set further includes genes linked to protein degradation (*NTA1*, *SEL1*) and response to DNA damage (*MEC1*). In reality, the mechanisms underlying LateBicluster 102 are tightly interconnected and assume complementary roles in the yeast response to heat stress [22]. For instance, *CKB1* (first set) is also involved in the response to DNA damage, and in growth and proliferation.

4 Conclusion

In this work, we discussed the complexity of identifying and reporting groups of genes exhibiting a similar temporal pattern, with potential delay, in a time series gene expression matrix. Available algorithms yield exponential time complexity due to the combinatorial explosion of pattern occurrences, which makes exhaustive enumeration of maximal time-lagged biclusters infeasible in most cases of interest [9], [10]. We proposed LateBiclustering, a heuristic time-lagged biclustering algorithm able to deliver biologically meaningful biclusters while enabling a significant reduction of the result space. Our strategy guarantees a time complexity that is polynomial on the size of the input.

Using real data concerning the response of *Saccharomyces cerevisiae* to heat stress, LateBiclustering was able to: (i) compute successfully in a regular desktop machine, while the exhaustive version quickly exceeded the amount of memory available in the system; (ii) capture interesting cascades of time-lagged patterns whose associated genes, functions and delays were consistent with the adaptations experimented by the yeast cell under those adverse conditions.

The results highlight that LateBiclustering is practical and can provide valuable insight into the functioning of biological systems. To make LateBiclustering accessible, we will integrate it into the BiGGEsTS software [17]. Additionally, we plan to address the discovery of patterns with bounded time lags, useful for discarding instances with unreasonable delays in long time series. Further challenges deserve investigation. For instance, although pattern matching uncovers interesting relationships, it is known that genes may exhibit similar expression profiles even if not functionally related. Likewise, players involved in the same biological process can yield distinct behavior.

ACKNOWLEDGMENTS

JPG was funded by European Union's FP7 during the tenure of an ERCIM "Alain Bensoussan" postdoctoral fellowship (ref. 246016). Additionally, this work was funded by Fundação para a Ciência e a Tecnologia (FCT) through projects PEst-OE/EEI/LA0021/2013 and PTDC/EIA-EIA/111239/2009 (Neuroclinomics).

REFERENCES

[1] I. P. Androulakis, E. Yang, and R. R. Almon, "Analysis of time-series gene expression data: Methods, challenges, and opportunities," *Annu. Rev. Biomed. Eng.*, vol. 9, pp. 205–28, Feb. 2007.

[2] Z. Bar-Joseph, A. Gitter, and I. Simon, "Studying and modelling dynamic biological processes using time-series gene expression data," *Nature Rev. Genetic.*, vol. 13, no. 8, pp. 552–564, Jul. 2012.

[3] A. Kundaje, X. Xin, C. Lan, S. Lianoglou, M. Zhou, L. Zhang, and C. Leslie,"A predictive model of the oxygen and heme regulatory network in yeast," *PLoS Comput. Biol.*, vol. 4, no. 11, p. e1000224, Nov. 2008.

[4] S. C. Madeira and A. L. Oliveira, "Biclustering algorithms for biological data analysis: A survey," *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, vol. 1, no. 1, pp. 24–45, Jan. 2004.

[5] R. Peeters, "The maximum edge biclique problem is NP-complete ," *Discr. Appl. Math.*, vol. 131, no. 3, pp. 651–654, Sep. 2003.

[6] S. C. Madeira, M. C. Teixeira, I. Sá-Correia, and A. L. Oliveira, "Identification of regulatory modules in time series gene expression data using a linear time biclustering algorithm," *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, vol. 7, no. 1, pp. 153–165, 2010.

[7] S. C. Madeira and A. L. Oliveira, "A polynomial time biclustering algorithm for finding approximate expression patterns in gene expression time series," *Algorithms Mol. Biol.*, vol. 4, p. 8, 2009.

[8] U. Alon, "Network motifs: Theory and experimental approaches," *Nature Rev. Genetics*, vol. 8, no. 6, pp. 450–461, 2007.

[9] L. Ji and K.-L. Tan, "Identifying time-lagged gene clusters using gene expression data," *Bioinformatics*, vol. 21, no. 4, pp. 509–516, Feb. 2005.

[10] S. C. Madeira and A. L. Oliveira, "Efficient biclustering algorithms for time series gene expression data analysis," in *Proc. 10th Int. Work-Conf. Artif. Neural Netw.*, Salamanca, Spain, 2009, pp. 1013–1019.

[11] T. Mitsa, *Temporal Data Mining*. London, U.K.: Chapman & Hall/CRC, 2010.

[12] D. Gusfield, *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge, U.K.: Cambridge Univ. Press, 1997.

[13] E. Ukkonen, "On-line construction of suffix trees," *Algorithmica*, vol. 14, no. 3, pp. 249–260, Sep. 1995.

[14] L. Chi and K. Hui, "Color set size problem with applications to string matching," in *Proc. Third Annu. Symp. Combin. Pattern Matching*, 1992, pp. 230–243.

[15] B. Schieber and U. Vishkin, "On finding lowest common ancestors: Simplification and parallelization," *SIAM J. Comput.*, vol. 17, no. 6, pp. 1253–1262, 1988.

[16] A. P. Gasch, P. T. Spellman, C. M. Kao, O. Carmel-Harel, M. B. Eisen, G. Storz, D. Botstein, and P. O. Brown, "Genomic expression programs in the response of yeast cells to environmental changes," *Mol. Biol. Cell*, vol. 11, no. 12, pp. 4241–57, Dec. 2000.

[17] J. P. Gonçalves, S. C. Madeira, and A. L. Oliveira, "BiGGEsTS: integrated environment for biclustering analysis of time series gene expression data," *BMC Res. Notes*, vol. 2, p. 124, 2009.

[18] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock, "Gene ontology: tool for the unification of biology. The Gene Ontology Consortium." *Nat. Genet.*, vol. 25, no. 1, pp. 25–9, May 2000.

[19] S. Bauer, S. Grossmann, M. Vingron, and P. N. Robinson, "Ontologizer 2.0–a multifunctional tool for GO term enrichment analysis and data exploration," *Bioinformatics*, vol. 24, no. 14, pp. 1650–1651, Jul. 2008.

[20] D. W. Huang, B. T. Sherman, and R. A. Lempicki, "Bioinformatics enrichment tools: Paths toward the comprehensive functional analysis of large gene lists," *Nucleic Acids Res.*, vol. 37, no. 1, pp. 1–13, Jan. 2009.

[21] J. M. Cherry, C. Adler, C. Ball, S. A. Chervitz, S. S. Dwight, E. T. Hester, Y. Jia, G. Juvik, T. Roe, M. Schroeder, S. Weng, and D. Botstein, "SGD: Saccharomyces Genome Database.," *Nucleic Acids Res.*, vol. 26, no. 1, pp. 73–79, Jan. 1998.

[22] D. E. Levin, "Regulation of cell wall biogenesis in Saccharomyces cerevisiae: The cell wall integrity signaling pathway," *Genetics*, vol. 189, no. 4, pp. 1145–1175, Dec. 2011.

Joana P. Gonçalves received a five-year "Licenciatura" (roughly equivalent to BSc+MSc) degree in computer science from the University of Beira Interior, Covilhã, Portugal, in 2007, and the PhD degree in computer science from Instituto Superior Técnico, Technical University of Lisbon, Portugal, in 2013. She was a doctoral researcher at INESC-ID in Lisbon between 2007 and 2013, a visiting scholar at the University of Leuven, Belgium, in 2009/2010, and a research assistant at Imperial College London, United Kingdom, in 2012. She is currently an ERCIM/Marie Curie postdoctoral fellow at the Centrum Wiskunde & Informatica (CWI) in Amsterdam, The Netherlands. Her research interests include algorithms and data structures, pattern matching and recognition, data mining, machine learning, and bioinformatics.

Sara C. Madeira received a five-year BSc degree in computer science from the University of Beira Interior, Covilhã, in 2000, and the MSc and PhD degrees in computer science and engineering (CSE) from Instituto Superior Técnico (IST), Technical University of Lisbon, Portugal, in 2002 and 2008. She is currently an assistant professor at the CSE Department of IST, and a senior researcher at the Knowledge Discovery and Bioinformatics (KDBIO) group of INESC-ID in Lisbon. Her research interests include algorithms and data structures, data mining, machine learning, bioinformatics, and medical informatics.