A Linear Time Biclustering Algorithm for Time Series Gene Expression Data

Sara C. Madeira and Arlindo L. Oliveira

INESC-ID / University of Beira Interior (smadeira@di.ubi.pt)
INESC-ID / IST (aml@inesc-id.pt)

ABSTRACT

Several non-supervised machine learning methods have been used in the analysis of gene expression data obtained from microarray experiments. Recently, biclustering, a non-supervised approach that performs simultaneous clustering on the row and column dimensions of the data matrix, has been shown to be remarkably effective in a variety of applications.

The goal of biclustering is to find subgroups of genes and subgroups of conditions, where the genes exhibit highly correlated behaviors. In the most common settings, biclustering is an NP-complete problem, and heuristic approaches are used to obtain sub-optimal solutions using reasonable computational resources.

In this work, we examine a particular setting of the problem, where we are concerned with finding biclusters in time series expression data. In this setting, we are interested in finding biclusters where the columns are consecutive in time. For this particular version of the problem, we propose an algorithm that finds and reports all relevant biclusters in time linear on the size of the data matrix.

This impressive reduction in complexity is obtained by manipulating a discretized version of the data matrix and by using advanced string manipulation techniques based on suffix trees. We report results in both synthetic and real world datasets that show the effectiveness of the approach.

1 INTRODUCTION

Recent developments in DNA chips now enable the simultaneous measure of the expression level of a large number of genes (sometimes all the genes of an organism) for a given experimental condition [1]. The samples may correspond to different time points, different environmental conditions, different organs or different individuals. Simply visualizing this kind of data, which is widely called *gene expression data* or simply *expression data*, is challenging. Using it to extract biologically relevant knowledge is even harder [6].

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

A number of different objectives are pursued when this type of analysis is undertaken. Among these, relevant examples are the classification of genes, the classification of conditions and the identification of regulatory processes. Clustering techniques have been extensively applied towards these objectives. However, applying clustering algorithms to gene expression data runs into a significant difficulty. Many activation patterns are common to a group of genes only under specific experimental conditions. In fact, our general understanding of cellular processes leads us to expect subsets of genes to be co-regulated and co-expressed only under certain experimental conditions, but to behave almost independently under other conditions. Discovering such local expression patterns may be the key to uncovering many genetic mechanisms that are not apparent otherwise. Researchers have therefore moved past this simple idea of row or column clustering and have turned to biclustering in their quest for the discovery of local patterns in microarray data.

The term biclustering was first used by Cheng and Church [3] in gene expression data analysis. It refers to a distinct class of clustering algorithms that perform simultaneous row-column clustering. The goal of biclustering techniques is to identify subgroups of genes and subgroups of conditions, by performing simultaneous clustering of the rows and columns of the gene expression matrix. Unlike clustering algorithms, biclustering algorithms identify groups of genes that show similar activity patterns under a specific subset of the experimental conditions.

Many approaches to biclustering gene expression matrices have been proposed to date [7]. In its general form, the problem is known to be NP-complete. In fact, even when the matrix contains only two distinct values, the problem of finding a maximal constant bicluster, i.e., a maximal sub-matrix with constant values, is NP-complete, since there exists a

straightforward reduction from the problem of finding a maximal biclique in a bipartite graph [10]. Given this result, almost all the approaches to biclustering presented to date are heuristic and obtain only approximate results. In a few cases, exhaustive search methods have been used, but limits are imposed on the size of the biclusters that can be found, in order to obtain reasonable runtimes.

There exists, however, one particular restriction to the problem that has not been considered before, and that leads to a tractable problem and, indeed, to a surprisingly efficient linear time algorithm for the problem of finding all maximal biclusters. This restriction is applicable when the gene expression data corresponds to snapshots in time of the expression level of the genes. Under this experimental setup, the researcher is, in many cases, particularly interested in biclusters with contiguous columns, that correspond to samples taken in consecutive instants in time. We show, in this paper, that in this case it is possible to have a linear time algorithm that finds all maximal consecutive column biclusters, under specific assumptions.

The remainder of this paper is organized as follows. Section 2 defines the problem and describes the relevant related work and the basic concepts needed to understand the approach. Section 3 derives the central result of this work and presents the algorithm. Section 4 illustrates the efficiency of the algorithm with synthetic data and presents some preliminary results obtained with real data. Finally, Section 5 presents the conclusions and some open research problems.

2 DEFINITIONS AND RELATED WORK

2.1 Finding biclusters in genomic expression data

Let A' be an n row by m column matrix, where A'_{ij} represents the expression level of gene i under condition j.

Table 1. Toy example of a gene expression matrix

	Cond. 1	Cond. 2	Cond. 3	Cond. 4	Cond. 5
Gene 1	0.07	0.73	-0.54	0.45	0.25
Gene 2	-0.34	0.46	-0.38	0.76	-0.44
Gene 3	0.22	0.17	-0.11	0.44	-0.11
Gene 4	0.70	0.71	-0.41	0.33	0.35

In this work, we are interested in the case where the gene expression levels can be discretized to a set of symbols of interest, Σ, that represent distinctive activation levels. In the simpler case, Σ may contain only three symbols, $\{U, D, N\}$ meaning *UpRegulated*, *DownRegulated* or *NoChange*. In other applications, the values in matrix A' may be discretized to a larger set of values. The discretized version of matrix A' is matrix A. Therefore, $A_{ij} \in \Sigma$ represents the discretized value of the expression level of gene i under condition j.

Table 2 represents a possible discretization of the gene expression values in Table 1. In this discretization, an expression level was considered as *NoChange* if it falls in the range $[-0.3 : 0.3]$.

Table 2. Discretized toy example of gene expression matrix

	1	2	3	4	5
1	N	U	D	U	N
2	D	U	D	U	D
3	N	N	N	U	N
4	U	U	D	U	U

Such a matrix A, with n rows and m columns, is defined by its set of rows, R, and its set of columns, C. Let $I \subseteq R$ and $J \subseteq C$ be subsets of the rows and columns, respectively. Then, $A_{IJ} = (I, J)$ denotes the sub-matrix of A that contains only the elements a_{ij} belonging to the sub-matrix with set of rows I and set of columns J. We will use A_{iC} to denote row i of matrix A and A_{Rj} to denote column j of matrix A.

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

The specific problem addressed by biclustering algorithms can now be defined. Given a data matrix, A', or its discretized version, A, we want to identify a set of biclusters $B_k = (I_k, J_k)$ such that each bicluster B_k satisfies some specific characteristics of homogeneity. The exact characteristics of homogeneity vary from approach to approach, and will be studied in Section 2.2.

2.2 Bicluster models and metrics

A large number of models and metrics have been proposed for the selection of interesting biclusters [7]. When dealing with the non-discretized matrix A, the most flexible measures of interest identify highly correlated activity of genes under a subset of conditions. To understand these measures, let a_{iJ} represent the mean of the ith row in the bicluster, a_{Ij} the mean of the jth column in the bicluster and a_{IJ} the mean of all elements in the bicluster:

$$a_{iJ} = \frac{1}{|J|} \sum_{j \in J} a_{ij} \quad (1)$$

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

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

Constant biclusters are then identified as biclusters where the value of

$$VAR(I,J) = \sum_{i \in I, j \in J} (a_{ij} - a_{IJ})^2 \quad (4)$$

is as low as possible [5].

More interesting than constant biclusters are biclusters where the activity of the genes varies in a highly correlated way in different conditions, i.e., biclusters that minimize the following expression:

$$VAR_c(I,J) = \sum_{i \in I, j \in J} (a_{ij} - a_{Ij})^2 \quad (5)$$

The more general model seeks to find biclusters where the activity of gene i under condition j can be obtained from the sum of a factor due to the gene and a factor due to the column:

$$a_{ij} = \mu + \alpha_i + \beta_j \quad (6)$$

It can be shown [7, 3] that such a bicluster minimizes the value of the sum of the residues,

$$H(I,J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} r(a_{ij})^2 \quad (7)$$

where the residues $r(a_{ij})$ are given by

$$r(a_{ij}) = a_{ij} - a_{iJ} - a_{Ij} + a_{IJ} \quad (8)$$

Many heuristic approaches have been proposed [7] have been proposed for the selection of biclusters that minimize this type of merit functions.

However, the inherent difficulty of this problem has led many authors to a formulation based on a discretized version of the gene expression matrix [2, 9, 12]. In the discretized versions, the objective is to find biclusters that exhibit either constant rows (or constant columns) [2] or to find biclusters that contain genes that jointly respond to the conditions (columns) in the biclusters.

The model used in our work falls in the category of biclusters with conserved columns. In particular, we are interested in finding column coherent biclusters, i.e., biclusters that satisfy the following definition:

DEFINITION 1. *A column coherent bicluster $A_{IJ} = (I, J)$ is a subset $I = \{i_1, \ldots, i_k\}$ and a subset of columns $J = \{j_1, \ldots, j_s\}$ from the matrix A such that $A_{ij} = A_{lj}$ for all $i, l \in I$ and $j \in J$.*

With this definition, finding a set a maximal biclusters that satisfy this coherence property remains an NP-complete problem. However, we are specially interested in the analysis of time series expression data, and that leads to an important restriction.

2.3 Biclusters in time series expression data

When analyzing time series expression data, with the objective of isolating coherent activity between genes in a subset of conditions, it is reasonable to restrict the attention to biclusters with contiguous columns. We support this view by assuming that the activation of a set of genes under specific conditions corresponds to the activation of a particular biological process. As time goes on, biological processes start and finish, leading to increased (or decreased) activity of sets of genes that can be identified because they form biclusters with contiguous columns, as illustrated in Figure 1.

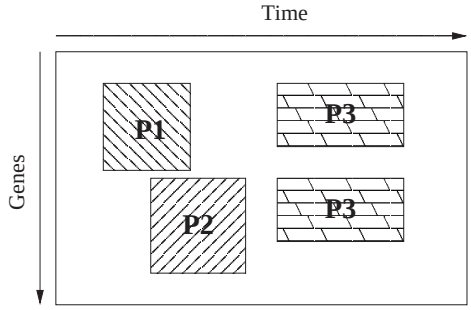


Fig. 1. Biclusters in time series expression data .

In this figure, the existence of three processes (P1, P2 and P3) leads to increased activity of different sets of genes, represented by three biclusters. Note that, although the columns of each of the biclusters are contiguous, the rows are in arbitrary positions, and are represented as contiguous for processes P1 and P2 only for convenience. The identification of biological processes that lead to the creation of the biclusters, together with their relationship, is crucial for the identification of gene regulatory networks and for the classification of genes.

This leads us to the definition of the type of biclusters that are of interest in this work.

DEFINITION 2. *A contiguous column coherent bicluster (ccc-bicluster) $A_{IJ} = (I, J)$ is a subset $I = \{i_1, \ldots, i_k\}$ and a* **contiguous** *subset of columns $J = \{r, r+1, \ldots, s-1, s\}$ from the matrix A such that $A_{ij} = A_{lj}$ for all $i, l \in I$ and $j \in J$.*

For the remainder of this work, we will refer to a contiguous column coherent bicluster simply as a ccc-bicluster. By definition, each row in matrix A is a ccc-bicluster. These are trivial biclusters and will not be of interest, in general. The bicluster with only one row or only on column will also be considered as trivial.

Each ccc-bicluster defines a string s that is common to every row in the ccc-bicluster, between columns r and s of matrix A. Figure 2 illustrates two ccc-biclusters that appear in the matrix in Table 2. These two ccc-biclusters are maximal, in the sense that they are not properly contained in any other ccc-biclusters. This notion will be defined more clearly later on.

	1	2	3	4	5
1	N	U	D	U	N
2	D	U	D	U	D
3	N	N	N	U	N
4	U	U	D	U	U

Bicluster 1: ({1,2,4),(2,3,4)}

Bicluster 2: ({1,3),(4,5)}

Fig. 2. Two ccc-biclusters for the matrix in Table 2 .

2.4 Suffix trees

A suffix tree is a data structure built over all the suffixes of a string that exposes its internal structure of a string. It has been extensively used to solve a large number of string manipulation problems.

DEFINITION 3. A *suffix tree* of a $|s|$-character string s is a rooted directed tree with exactly $|s|$ leaves, numbered 1 to $|s|$. Each internal node, other than the root, has at least two children and each edge is labeled with a nonempty substring of s. No two edges out of a node have edge-labels beginning with the same character. The key feature of the suffix tree is that for any leaf i, the label of the path from the root to the leaf i exactly spells out the suffix of s that starts at position i.

In order to enable the construction of a suffix tree obeying this definition when one suffix of s matches a prefix of another suffix of s, a character terminator, that does not appear nowhere else in the string, is added to its end. For example, the suffix tree for the string s=TACTAG is presented in Figure 3.

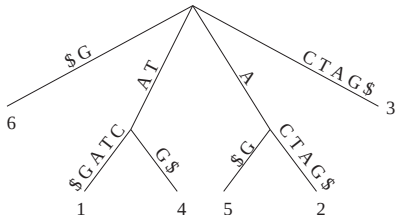


Fig. 3. Suffix tree for s=TACTAG.

The suffix tree construction for a set of strings, called a generalized suffix tree, can be easily obtained by consecutively building the suffix tree for each string of the set. The leaf number of the single string suffix tree can easily be converted to two numbers, one identifying the string and the other identifying the starting position in that string. For example, the generalized suffix tree for the strings s_1=TACTAG and s_2=CACT is presented in Figure 4. When dealing with generalized suffix trees, each string is terminated with a different terminator that is not in the original alphabet. In our case, the terminator for string s_i is denoted by $\$i$. There is a suffix link from node v to node u if the path-label of node u represents a suffix of the path-label of node v and the lenght of the path-label of u is exactly equal to the length of the path-label of v minus 1. For a constant size alphabet, suffix trees can be

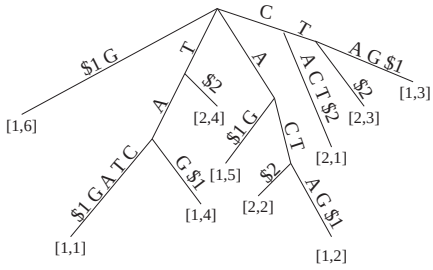


Fig. 4. Generalized suffix tree for S_1=TACTAG and S_2=CACT.

built in time that is linear on the size of the string, using a number of different algorithms [11, 8, 14]. Generalized suffix trees can be built in time linear on the sum of the sizes of the strings, that is, on time linear on the size of the input data. This impressively low complexity result is at the root of many efficient algorithms for string manipulation.

When the number of symbols in Σ, $|\Sigma|$, is large, then the analysis of the time complexity of the tree construction algorithms is slightly more elaborated [4], since it becomes dependent on the data structure used at each node to maintain a list of its children.

3 AN ALGORITHM FOR BICLUSTERING TIME SERIES EXPRESSION DATA

3.1 Biclusters and suffix trees

We can now introduce the major results of this work, that lead to the linear time biclustering algorithm. We first introduce the concept of contiguous column maximal bicluster.

DEFINITION 4. *A ccc-bicluster A_{IJ} is maximal if no other ccc-bicluster exists that properly contains A_{IJ}, i.e., if for all other ccc-biclusters A_{LM}, $I \subseteq L$ and $J \subseteq M \Rightarrow I = L \wedge J = M$.*

We will also call a ccc-bicluster *right-maximal* if it cannot be extended to the right by adding one more column at the end, and *left-maximal* if it cannot be extended to the left by adding one more column at the beginning.

Stated more plainly, a ccc-bicluster is maximal if no more rows nor contiguous columns (either at the right or at the left) can be added to it while maintaining the coherence property from Definition 2.

We will now consider a new alphabet $\Sigma' = \Sigma \times \{1 \dots m\}$, where each element of the new alphabet Σ' is obtained by concatenating one symbol in Σ and one number in the range $\{1 \dots m\}$ and a function $f : \Sigma \times \{1 \dots m\}$ defined by $f(a,k) = a|k$ where $a|k$ represents the character in Σ' obtained by concatenating the symbol a with the number k. For example, if $\Sigma = \{U, D, N\}$ and $m = 3$, then $\Sigma' = \{U1, U2, U3, D1, D2, D3, N1, N2, N3\}$. For this case, $f(U,2) = U2$ and $f(D,1) = D1$. Consider now the set of strings $S = \{s_1, \dots, s_n\}$ obtained by mapping each row A_{iJ} in matrix A to string s_i such that $s_i(j) = f(a_{ij}, j)$. Each of these strings has m characters and corresponds to the symbols in a row of matrix A after the above alphabet transformation. As such, the matrix that we have been using as example becomes, after the transformation, the matrix in Table 3.

Table 3. Discretized toy example of gene expression matrix, after alphabet transformation

	1	2	3	4	5
1	N1	U2	D3	U4	N5
2	D1	U2	D3	U4	D5
3	N1	N2	N3	U4	N5
4	U1	U2	D3	U4	U5

Consider now the generalized suffix tree T obtained from the set of strings S. Let v be an internal node of T and let $L(v)$ denote the number of leaves in the sub-tree rooted at node v. Additionally, let $P(v)$ be the path-length of node v, that is, the number of characters in the string that labels the path from the root to node v.

Is is easy to verify that every internal node of the generalized suffix tree T corresponds to one ccc-bicluster of the matrix A. This is so because an internal node v in T corresponds to a given substring that is common to every row that has a leaf rooted in v. Therefore, node v defines a ccc-bicluster that has $P(v)$ columns and a number of rows equal to $L(v)$.

We will state with only sketches of proofs two lemmas that lead to the main theorem.

LEMMA 1. *Every right-maximal ccc-bicluster corresponds to one node in T.*

Proof: let B be a ccc-bicluster that cannot be extended to the right, i.e., a right-maximal ccc-bicluster. Since B is a ccc-bicluster, every row in B shares the substring that defines B. Since B cannot be extended to the right, at least one of the rows in B must have a character that differs from the character in the other rows, in the first column to the right that is not in B. Therefore, there is a node in T that matches B and the path-label of that node is the string that defines B.

LEMMA 2. *Let node v_1 correspond to a ccc-bicluster B_1 and node v_2 correspond to a ccc-bicluster B_2. Then, if there is a suffix link from node v_1 to node v_2, bicluster B_2 contains one less column than bicluster v_1.*

Proof: follows directly from the definition of suffix links.

From these lemmas, we can now derive the theorem that is our main result.

THEOREM 1. *Let v be an internal node in the generalized suffix tree T. Then, v corresponds to a maximal ccc-bicluster iff $L(v) > L(u)$ for every node u such that there is a suffix link from u to v. Furthermore, every maximal ccc-bicluster corresponds to a node v satisfying this condition.*

Proof: let B be a maximal ccc-bicluster. Let s be the string that defines B. Now, s must lead to a node v (by lemma 1). If node v does not have an incoming suffix link, the conditions of the theorem are met. Since B is also left-maximal, every node u that defines a bicluster B' with one more column than B (lemma 2) must have $L(v) > L(u)$, since B' cannot contain all the rows in B (otherwise, B would not be left-maximal). Therefore, it is sufficient to check that every node u that has a suffix link directed at v has $L(u) < L(v)$ to ensure that node v corresponds to a maximal ccc-bicluster.

Figure 5 illustrates the generalized suffix tree obtained from the strings that correspond to the rows of the matrix in Table 3. This figure does not contain the leaves that represent string terminators that are direct daughters of the root. Each non-terminal node, other than the root, is labeled with the value of $L(v)$, the number of leaves in its subtree. Also shown in this tree are the suffix links between nodes. For clarity, leaves that have as parent the root are not shown. Also not shown are the suffix links that end at the root.

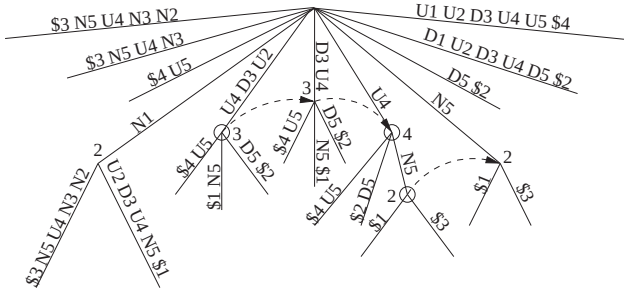


Fig. 5. Generalized suffix tree for matrix A3

Figure 5 shows that there are 6 non-terminal nodes, other than the root. Each one of these nodes corresponds to one ccc-bicluster. However, some of them are trivial (nodes with edge labels $N1$, $U4$ and $N5$, since they have edge labels with only one character). Others are non maximal (nodes with edge labels $D3U4$ and $N5$), since they have an incoming suffix

link from a node with the same number of leaves. This leaves nodes with edge labels $U2D3U4$ and $U4N5$ as maximal, non-trivial ccc-biclusters. These nodes correspond to the maximal ccc-biclusters $(\{1,2,4\},\{2,3,4\})$ and $(\{1,3\},\{4,5\})$ (see Figure 2). The rows in each ccc-bicluster are obtained from the terminators in the leaves in the subtree of each node v, while the columns in each ccc-bicluster are obtained from the value of $P(v)$ and the information on the edge-label that connects v to its parent.

3.2 Finding and reporting biclusters

Theorem 1 directly implies that there is a linear time algorithm that lists all maximal contiguous column coherent biclusters in matrix A. This algorithm is shown in Figure 6. With appropriate data structures at the nodes, each of the four first lines and the final two iterations over all nodes in the tree are executed in time linear on the size of the input matrix. A more detailed analysis shows that the increase in the alphabet size does not have an impact on this linear time complexity. In fact, only two types of nodes have more than $|\Sigma|$ children: the root node and nodes that have as children only leaf nodes. In both cases, it is easy to devise a data structure that enables constant time manipulation of these nodes.

Map each row in matrix A, A_{iJ} to a string s_i using f
Build a generalized suffix tree T for the set S
Compute $P(v)$ for each node v in T
Compute $L(v)$ for each node v in T and mark v as Valid
For each node v in T
  If there is a suffix link (v,u) and $L(v) >= L(u)$
    Mark u as Invalid
For each node v in T
  If v is Valid
    Report the ccc-bicluster that corresponds to v

Fig. 6. Linear time algorithm for the extraction of all maximal ccc-biclusters in a discretized and transformed gene expression matrix.

4 EXPERIMENTAL RESULTS

In order to validate the approach, we performed experiments with both synthetic data and real data, using a prototype implementation of the algorithm, coded in Java. All experiments were performed in a a 3GHz Pentium-4 machine, running Linux with 1GB of memory.

4.1 Experiments with synthetic data

To evaluate the efficiency of the algorithm, and validate experimentally the predicted linear time complexity, we generated matrices with random values, on which 10 biclusters were hidden, with dimensions ranging from $15-25$ rows and $8-12$ columns. The size of the matrices varied from 250×50 (rows $\times$ columns) up to 1000×250. We used a three character alphabet, $\Sigma = \{U, D, N\}$.

In all cases, we recovered the *planted* ccc-biclusters, together with a large number of artifacts that result from random coincidences in the data matrix. Figure 7 shows a plot of the variation of the CPU time with the size of the input data matrix. A clear linear relationship over several orders of magnitude is apparent from this plot. It is also clear from this plot that the algorithm runs even in the larger matrices used in the synthetic data in less than 15 seconds.

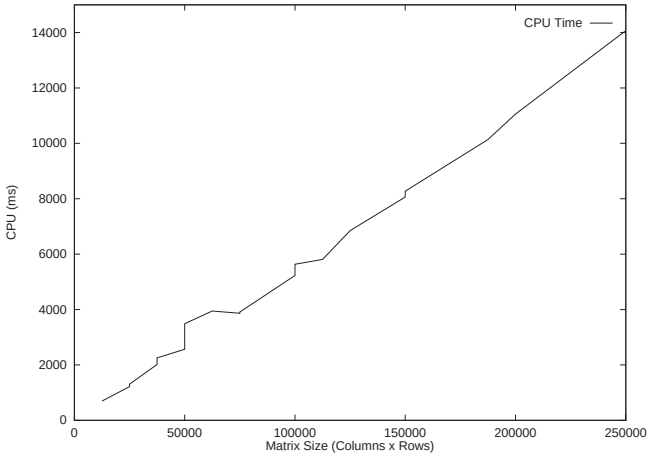


Fig. 7. CPU time versus size of the synthetic input data .

4.2 Experiments with real data

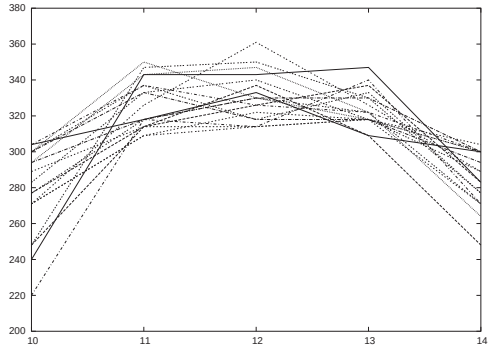
To validate the approach, we used the time-series data from yeast described in [13]. This represents a dataset with 2884 genes and 17 conditions, corresponding to 17 successive instants of time.

The original data was processed to a range between 0 and 600 (as in [3]) and then discretized to an alphabet $\Sigma = \{U, D, N\}$. Gene expression levels were considered to be in the N range if they did not deviate more than 0.8 standard deviations from the average value.

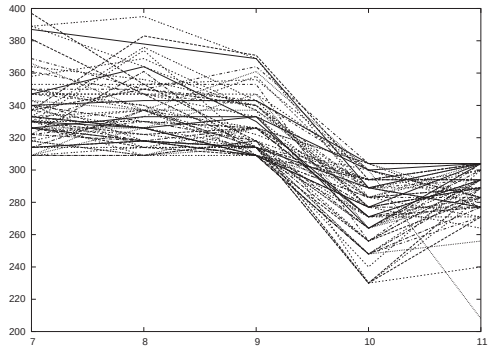
The resulting matrix was processed by our algorithm and all maximal 5967 ccc-biclusters were extracted. The whole process took 28.3 seconds.

Due to time limitations, we present only very preliminary results on this data. Removal of uninteresting ccc-biclusters (e.g., constant N ccc-biclusters, trivial ccc-biclusters with only one column) still left us with more than 5000 ccc-biclusters. Many other criteria can be applied to filter irrelevant ccc-biclusters, but were not used in these experiments.

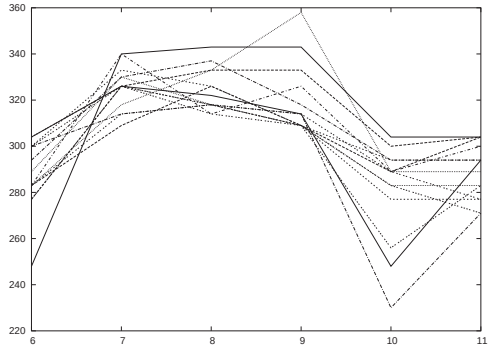
The results obtained with some ccc-biclusters of moderate size that exhibit correlated gene activity are shown in Figure 8.



(a) Bicluster 471



(b) Bicluster 1014



(c) Bicluster 2490

Fig. 8. Expression level of genes in selected ccc-biclusters .

These figures show that the methodology proposed in this work is able to identify highly correlated expression patterns of genes, under a given subset of conditions. It is worthwhile to note that the highly correlated activity under this subset of columns does not necessarily translate into highly correlated activity under all conditions. For example, the genes in ccc-bicluster 1014 do not exhibit good correlation except between time instants 7 and 11, as a comparison of Figure 9 and Figure 8(b) clearly shows. This ability to identify highly correlated behaviors under specific subsets of conditions is inherent to biclustering approaches (and not, in particular, to our method)

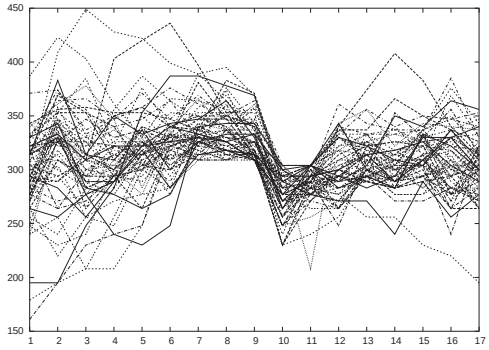


Fig. 9. Expression level of genes in bicluster 1014 .

and is very important in the identification of relevant genomic regulation mechanisms.

5 CONCLUSIONS

In this work, we presented a linear time algorithm for the identification of all maximal contiguous column biclusters in a discretized gene expression matrix, obtained from time series genomic expression data.

By discretizing the gene expression values, and manipulating the strings that correspond to each row using string manipulation techniques, we have been able to demonstrate that there is a correspondence between the maximal ccc-biclusters and the internal nodes of the generalized suffix tree that represents the rows of the matrix. This leads to a very efficient algorithm for the extraction of ccc-biclusters, that runs in a few seconds even for matrices with thousands of genes and hundreds of conditions.

We have demonstrated the correctness of the algorithm and sketched the complexity analysis. We have also presented experimental results with synthetic data and very preliminary results with real data from yeast.

This work opened many promising directions for future research, both in the short and in the long term. In the short term, we are working on the selection and evaluation of criteria for filtering the ccc-biclusters obtained by the algorithm. In fact, although the algorithm generates only ccc-biclusters that are maximal (and, in that sense, as interesting as possible), other criteria of interest need to be developed in order to reduce the number of potentially interesting ccc-biclusters. Additionally, we are interested in the development of generalizations of these algorithm to deal with imperfect ccc-biclusters. Many techniques developed by the string processing community can be applied to this problem, and are likely to derive very efficient solutions to that more general problem.

In the long term, it is interesting to apply this and related techniques to the identification of candidate regulatory mechanism, related with the processes that are potentially causing the appearance of each bicluster.

REFERENCES

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

[2]A. Califano, G. Stolovitzky, and Y. Tu. Analysis of gene expression microarays for phenotype classification. In *Proceedings of the International Conference on Computacional Molecular Biology*, pages 75–85, 2000.

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

[4]D. Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press, 1997.

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

[6]L. Lazzeroni and A. Owen. Plaid models for gene expression data. Technical report, Stanford University, 2000.

[7]S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, Jan-Mar 2004.

[8]E. McCreight. A space economical suffix tree construction algorithm. *Journal of the ACM*, 23:262–272, 1976.

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

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

[11]P.Weiner. Linear pattern matching algorithms. In *Proceedings of the 14th IEEE Symposium on Switching and Automata Theory*, pages 1–11, 1973.

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

[13]S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church. Systematic determination of genetic network architecture. *Nature Genetics*, 22:281–285, 1999.

[14]E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14:249–260, 1995.