BiMine+: An efficient algorithm for discovering relevant biclusters of DNA microarray data

Wassim Ayadi[1,2], Mourad Elloumi[1], and Jin Kao Hao[2]

[1]*LaTICE, Higher School of Sciences and Technologies of Tunis, University of Tunis, 1008 Tunis, Tunisia;*
[2]*LERIA, University of Angers, 2 Boulevard Lavoisier, 49045 Angers, France*

Abstract

Biclustering is a very useful tool for analyzing microarray data. It aims to identify maximal groups of genes which are coherent with maximal groups of conditions. In this paper, we propose a biclustering algorithm, called *BiMine+*, which is able to detect significant biclusters from gene expression data. The proposed algorithm is based on two original features. First, *BiMine+* is based on the use of a new tree structure, called *Modified Bicluster Enumeration Tree* (MBET), on which biclusters are represented by the profile shapes of genes. Second, *BiMine+* uses a pruning rule to avoid both trivial biclusters and combinatorial explosion of the search tree. The performance of *BiMine+* is assessed on both synthetic and real DNA microarray datasets. Experimental results show that *BiMine+* competes favorably with several state-of-the-art biclustering algorithms and is able to extract functionally enriched and biologically relevant biclusters.

Keywords: Biclustering, gene expression data, evaluation function, enumeration algorithm, data mining.

1. Introduction

DNA microarray technology is a revolutionary method enabling the measurement of expression levels of thousands of genes in a single experiment under diverse experimental conditions. Associated to this technology, microarray data analysis aims at extracting useful information that can be applied in medical and biological studies (47; 23; 31). In this context, biclustering of time series data is a particularly interesting approach since it allows the simultaneous identification of a maximum of genes that show highly correlated expression patterns through a maximum of time-dependent experimental conditions (samples) (7; 38; 27).

DNA microarray data is usually represented by a data matrix $M(I, J)$, where the i^{th} row, $i \in I=\{1, 2, \ldots, n\}$, represents the i^{th} gene, the j^{th} column, $j \in J=\{1, 2, \ldots, m\}$, represents the j^{th} condition (or time points) and the cell $M[i, j]$ represents the expression level of the i^{th} gene under the j^{th} condition. A bicluster is a subset of genes associated with a subset of conditions, i.e., a couple (I', J') such that $I' \subseteq I$ and $J' \subseteq J$.

Given a data matrix $M(I, J)$, the biclustering problem consists in extracting from $M(I, J)$ a group of coherent and significant biclusters of large size. In its general form, the biclustering problem is NP-hard (16; 38).

Existing biclustering algorithms can be grouped into two large classes (3): Those that adopt a systematic search approach and those that adopt a stochastic search one, also called metaheuristic approach. Algorithms that adopt a systematic search approach include greedy algorithms (9; 15; 16; 35; 50), divide-and-conquer algorithms (28; 44) and enumeration algorithms (4; 34; 48). Algorithms based on metaheuristic approach include neighbourhood-based algorithms (13), GRASP (20; 21) and evolutionary algorithms (12; 22; 25; 40).

In this paper, we introduce *BiMine+*[1], an enumerative heuristic algorithm designed for biclustering time series gene expression data. *BiMine+* is based on the use of a new tree structure, called *Modified Bicluster Enumeration Tree* (MBET). MBET can represent all types of biclusters, i.e., constant, additive, multiplicative and coherent evolution biclusters (38), of maximum size with similar trajectory patterns of expression levels and helps identify large biclusters with low overlap. The pruning rule employed by *BiMine+* allows it to avoid both trivial biclusters and combinatorial explosion of the search tree. Unlike many biclustering algorithms that may lead to highly overlapped biclusters or fail to detect certain types of biclusters, *BiMine+* is expected to extract all types of high quality biclusters of large size with low overlap.

The remainder of the paper is organized as follows: In section 2, we present the *Average Spearman's Rho* (ASR) evaluation function. In section

[1]The *BiMine+* software is available at: http://www.info.univ-angers.fr/pub/hao/BiMine+/BiMine+.html

3, we describe the general *BiMine+* algorithm. In section 4, experimental studies of *BiMine+* on both synthetic and real DNA microarray datasets are presented. Moreover, for real DNA microarray data, we illustrate a biological validation of the extracted biclusters via two web-tools, *FuncAssociate* (11) and *GOTermFinder*[2]. Conclusions are given in the last section.

2. The ASR evaluation function

Many evaluation functions exist for biclusters evaluation. One of the most popular evaluation functions is the Mean Squared Residue (MSR) (16). It has been used by several biclustering algorithms (2; 12; 15; 21; 40; 51; 52). MSR is deficient to assess correctly the quality of certain types of biclusters like multiplicative models (1; 15; 43; 50), though.

In (4), we have proposed a new evaluation function, called *Average Spearman's Rho* (ASR). Let (I', J') be a bicluster in a data matrix $M(I, J)$, the ASR evaluation function is then defined by:

$$ASR(I', J') = 2 * max \left\{ \frac{\sum\limits_{i \in I'} \sum\limits_{j \in I'; j \geq i+1} \rho_{ij}}{|I'|(|I'|-1)}, \frac{\sum\limits_{k \in J'} \sum\limits_{l \in J'; l \geq k+1} \rho_{kl}}{|J'|(|J'|-1)} \right\} \tag{1}$$

where ρ_{ij} $(i \neq j)$ is the spearman's rank correlation (33) associated with the row indices i and j in the bicluster (I', J'), ρ_{kl} $(k \neq l)$ is the spearman's rank correlation associated with the column indices k and l in the bicluster (I', J') and ASR(I', J') $\in$[-1..1].

A high (resp. low) ASR value, close to 1 (resp. close to -1), indicates that the genes/conditions of the bicluster are strongly (resp. weakly) correlated.

Let us notice that since the time complexity of ρ_{ij} is $O(m)$(46) then time complexity of ASR is $O(n^2m)$.

Finally, notice that the existing evaluation functions can roughly be classified into two families: numerical measures and qualitative measures. Numerical measures, like Pearson's correlation or Euclidean distance, are easy to compute but they are quite sensitive toward outliers and noise. Qualitative measures, like measures that consider only ups, downs and no change of conditions, are very sensitive to precise the values of changes. Hence, as ASR is based on Spearman's rank correlation it can be considered as a good compromise between numerical measures and qualitative ones.

[2]http://db.yeastgenome.org/cgi-bin/GO/goTermFinder

In the next section, we describe our new biclustering algorithm *BiMine+* which is an improvement of the *BiMine* algorithm (4).

3. The *BiMine+* Algorithm

BiMine+ is a heuristic (approximate) enumeration biclustering algorithm whose objective is to extract coherent and maximal size biclusters, i.e., a maximal groups of genes with a maximal groups of conditions where the genes exhibit highly correlated activities over a range of conditions. The algorithm uses a *Modified Bicluster Enumeration Tree* (MBET) to represent the identified biclusters, where each node of MBET contains the gene profile shape of a bicluster. The profile shape of a gene is defined as the behaviour of this gene, i.e., up, down or no change, over the conditions of the bicluster to which this gene belongs. This representation is important because it is recognized that in time-course microarray data analysis, genes are considered to be in the same cluster if their trajectory patterns of expression levels are similar (26; 36; 42; 45). To limit the size of MBET, *BiMine+* employs a pruning rule to eliminate any bicluster that has a number of the conditions lower than a given threshold. Finally, *BiMine+* uses the ASR evaluation function to provide a final assessment of each extracted bicluster.

Let M be a data matrix, *BiMine+* operates in three steps. The first step discretizes the data matrix M to obtain M'. The second step constructs from M' MBET that represents every possible maximal bicluster with a low-level overlap. Finally, we select among the extracted biclusters those that have an ASR value equal to or greater than a fixed threshold.

3.1. Discretization of the data matrix

During this step, we discretize the initial data matrix $M(I, J)$ where I=$\{1, 2, \ldots, n\}$ and J=$\{1, 2, \ldots, m\}$, into a matrix M' defined as follows:

$$M'[i,l] = \begin{cases} 1 & \text{if } M[i,l] < M[i,l+1] \\ -1 & \text{if } M[i,l] > M[i,l+1] \\ 0 & \text{if } M[i,l] = M[i,l+1] \end{cases} \tag{2}$$

with $i \in [1..n]$ and $l \in [1..m-1]$.

In microarray data analysis, genes are considered to be in the same cluster if their curves of genes expression levels are similar across a set of conditions (36; 42; 45). Hence, thanks to the Equation 2, the data matrix M' represents information about the profile shape, i.e., up (1), down (-1) and no change (0), of all rows (genes) over columns (conditions).

Discretization may have some drawbacks in some cases since it transforms real-valued data into discrete-valued data. Despite of this, discretization has been largely used for analysis of time series gene expression data (24; 29; 30; 32; 41). In our case, the purpose of using the discretized matrix M' is to identify coherent biclusters that share similar profile patterns regardless of the exact numeric values in the data matrix.

Finally, the ASR evaluation function is based on the real values of the initial data matrix and guarantees further the assessment of each identified bicluster.

3.2. Construction of the MBET tree and extraction of biclusters

After the discretization step, we construct the *Modified Bicluster Enumeration Tree* . The MBET tree is structured as follows (see Figures 1–5 for an illustration example):

1. The root is the empty bicluster.
2. The nodes at level one are the possible biclusters made up by one gene and its corresponding profile shape.
3. The i^{th} child of a node is made up by two parts: The first one is the union of the genes of the father and those of the i^{th} uncle, starting from the right side of the father. The second one is the intersection of the conditions of the father and those of the i^{th} uncle.

Since the number of the possible biclusters (nodes of MBET) increases exponentially, we employ a parametric rule to prune progressively some nodes. Indeed, a node is pruned if it has a number of conditions lower than a fixed threshold.

During the construction step, we extract the largest bicluster (leaf) from each subtree rooted by a node of the first level. In fact, the leaves of each subtree have a high-level overlap because they share, most of the time, the same genes. Hence, for each subtree we extract only the largest bicluster with a low-level overlap.

Among the extracted biclusters we drop those that are included in other biclusters or that have an ASR value lower than a fixed threshold or contain no more than two genes. The set of the remaining biclusters represents a solution to the biclustering problem.

To describe formally the *BiMine+* algorithm, let us define some notations:
M: data matrix,
M': discretized data matrix,
T_n: subtree made up by a node n and its children,
n, n': nodes in MBET,
$gene_n$: set of genes in the node n,
$cond_n$: set of conditions in the node n,
$Bc = (Ic, Jc)$: current bicluster,
δ: threshold of conditions number,
β: quality threshold of a bicluster,
$\mathcal{B}$: set of biclusters.

Algorithm 1 initializes MBET to the subtree T_0 made up by the empty node and its children.

Algorithm 1 *InitMBET*

```
1: **Input**: $M'$
2: **Output**: $MBET$ // subtree made up by the empty node and its children
3: $T_0$ = empty node
4: **for** each $gene \in M'$ **do**
5:     Extend $T_0$ by $Bc$=($Ic$,$Jc$) as a child of the root // where $Ic = gene$
       and $Jc$ = conditions of $gene$
6: **endfor**
7: $MBET = T_0$
8: **Return** $MBET$
```

Proposition 1: Time complexity of *InitMBET* is $O(nm)$, where n is the number of the rows and m the number of the columns of the data matrix M'.

Proof: Indeed, this step is achieved via a scanning of the whole data matrix M' that is of size nm. •

Algorithm 2 continues the construction of MBET and extracts biclusters. The first call to *BuildMBET* is made with T_0 as a parameter.

Proposition 2: Time complexity of *BuildMBET* is $O(2^n m log(m))$.

Algorithm 2 $BuildMBET(MBET)$//Current_$MBET$

```
1: **Output**: $\mathcal{B}$
2: $\mathcal{B} = \emptyset$
3: for each node $n$ in $MBET$ do
4:     for each unprocessed brother $n'$ of $n$ do
5:       $Ic$= $gene_n \cup gene_{n'}$; $Jc$= $cond_n \cap cond_{n'}$;
6:       if $|Jc| \geq \delta$ then
7:           $Bc$= $(Ic, Jc)$
8:           Insert $Bc$ as a child of $n$
9:           if $Bc$ have a maximum size leaf in the current subtree, rooted in level 1 then
10:              $\mathcal{B} = \mathcal{B} \cup \{Bc\}$
11:          endif
12:      endif
13:    endfor
14:    $T_n$= subtree made up by $n$ and its children
15:    $BuildMBET(T_n)$
16: endfor
17: Return $\mathcal{B}$
```

Proof: Indeed, to construct a new node from two existing ones in MBET, we merge the genes and we make the intersection of the conditions. Since the conditions of a node are sorted, the construction of the intersection of two subsets of conditions of size m boils down to the search of m elements in a sorted array of size m. This can be done via a dichotomic search with a time complexity $O(mlog(m))$. Then, a linear scanning is achieved on the extracted conditions to keep those that have the same profile shape in both nodes (biclusters). This can be done in time $O(m)$. Hence, the construction of a node is made in time $O(mlog(m))$. Since we have $O(2^n)$ nodes in the worst case then the construction of MBET is made in a time $O(2^n mlog(m))$. Hence, time complexity of $BuildMBET$ is $O(2^n mlog(m))$. •

Algorithm 3 selects among the extracted biclusters those for which the value of the ASR evaluation function is greater than or equal to a fixed threshold β.

Algorithm 3 *SelectBiclusters*

```
1: **Input**: $\mathcal{B}$
2: **Output**: $\mathcal{B}$
3: **for** each bicluster $(I', J')$ in $\mathcal{B}$ **do**
4:     **if** $ASR(I', J') < \beta$ **then** $\mathcal{B} = \mathcal{B} \backslash \{(I', J')\}$
5:     **endif**
6: **endfor**
7: **Return** $\mathcal{B}$
```

Proposition 3: Time complexity of *SelectBiclusters* is $O(n^3 m)$.

Proof: In fact, for a given bicluster, the ASR evaluation function is computed in time $O(n^2 m)$. In the worst case, we have $n-1$ extracted biclusters. Hence, time complexity of *SelectBiclusters* is $O(n^3 m)$. •

Then the whole *BiMine+* algorithm can be described in Algorithm 4.
Proposition 4: Time complexity of *BiMine+* is $O(2^n mlog(m))$.

Proof : Time complexity of the discretization step is $O(nm)$. Indeed, this step is achieved via a scanning of the whole data matrix M of size nm.

Algorithm 4 *BiMine+*

```
1: **Input**: $M$ ; $\delta$ ; $\beta$
2: **Output**: $\mathcal{B}$
3: Discretize $M$ by using Equation 2 to obtain $M'$
4: $MBET = InitMBET(M')$
5: $\mathcal{B} = BuildMBET(MBET)$
6: $\mathcal{B} = SelectBiclusters(\mathcal{B})$
7: **Return** $\mathcal{B}$
```

According to proposition 1, time complexity of $InitMBET$ is $O(nm)$. According to proposition 2, time complexity of $BuildMBET$ is $O(2^n mlog(m))$. Finally, according to proposition 3, time complexity of $SelectBiclusters$ is $O(n^3 m)$.

Hence, time complexity of $BiMine+$ is $O(2^n mlog(m))$. •

3.3. Illustrative Example

Table 1: Data matrix M.

	c'_1	c'_2	c'_3	c'_4	c'_5	c'_6
g_1	10	20	5	15	40	18
g_2	20	40	10	30	30	20
g_3	23	12	8	15	29	50
g_4	4	8	2	6	5	5
g_5	23	12	8	15	29	50
g_6	73	73	88	11	9	62

Let M be a data matrix (Table 1). Let us set $\delta = 4$ and $\beta = 0.85$. During the first step, we make the discretization of M using Equation 2 to obtain the data matrix M' (Table 2). During the second step, we construct MBET that represents every possible maximal bicluster that can be obtained from M'.

The first level of MBET is made up of nodes that represent the possible biclusters with one gene. Each node represents a row of data matrix M' (Figure 1). The second level of MBET is composed of nodes that are the union of genes and the intersection of the conditions in the first level. In Figure 2, we explain the construction of the children of node g_1. Each edge

Table 2: Data matrix M'.

	c_1	c_2	c_3	c_4	c_5
g_1	1	-1	1	1	-1
g_2	1	-1	1	0	-1
g_3	-1	-1	1	1	1
g_4	1	-1	1	-1	0
g_5	-1	-1	1	1	1
g_6	0	1	-1	-1	1

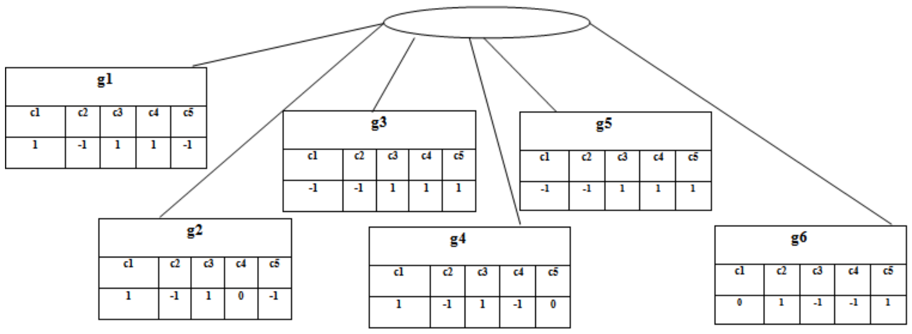


Figure 1: First level of MBET.

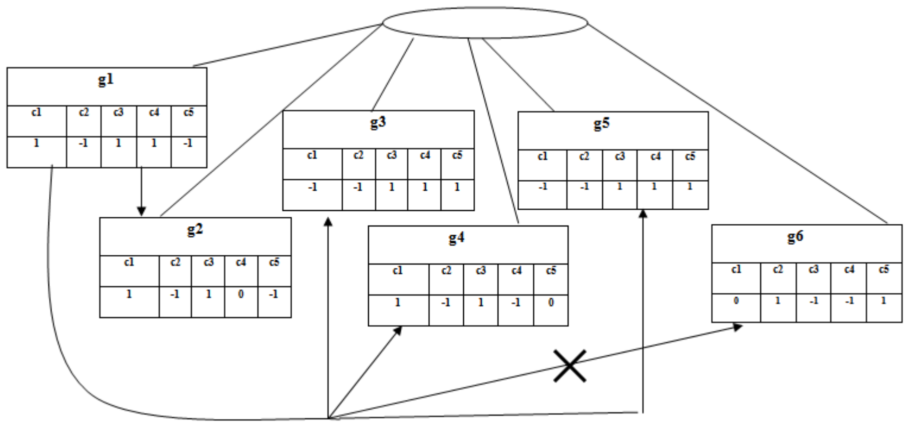


Figure 2: Children construction of the first node of the second level of MBET.

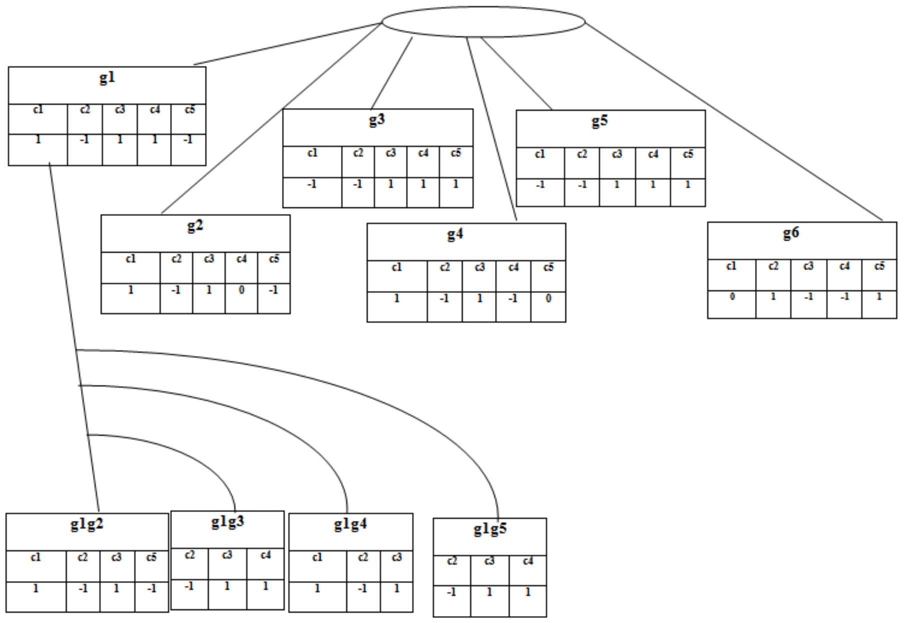


Figure 3: Second level of MBET subtree rooted by the node g_1.

without cross represents a valid combination between two nodes (with $\delta \geq 4$). First, we perform the union of genes of nodes labelled g_1 and g_2 (first uncle), and the intersection of the conditions (c_1:1, c_2:-1, c_3:1, c_4:1, c_5:-1) of g_1 with those of (c_1:1, c_2:-1, c_3:1, c_4:0, c_5:-1) of g_2. The number of real conditions (Table 1) after the intersection is greater than 4, i.e., $c'_1, c'_2, c'_3, c'_4, c'_5, c'_6$ where each condition l in M' corresponds to $(l, l+1)$ in M. Hence, we insert it as a first child of g_1. After that, we process g_1 with the node labelled g_3 (second uncle). We obtain the bicluster (g_1, g_3; c_2:-1, c_3:1, c_4:1) with the number of real conditions equal to 4, i.e., c'_2, c'_3, c'_4, c'_5, hence, we insert it as a child of g_1. We carry out the same process with node g_4. We obtain the bicluster (g_1, g_4; c_1:1, c_2:-1, c_3:1) with the number of conditions equal to 4, we insert it as a child of g_1. We process now g_1 with g_5, we obtain the bicluster (g_1, g_5; c_2:-1, c_3:1, c_4:1) with the number of real conditions equal to 4, hence we insert it. Finally, with g_6 we obtain the bicluster (g_1, g_6; $\emptyset$) with the number of real conditions equal to 0, hence we do not insert it (Figure 3). This completes the second level of the subtree of MBET rooted by the node g_1. The third level of MBET is made up of nodes that are the union of genes and the intersection of the conditions in the second level (Figure 4). At each level of

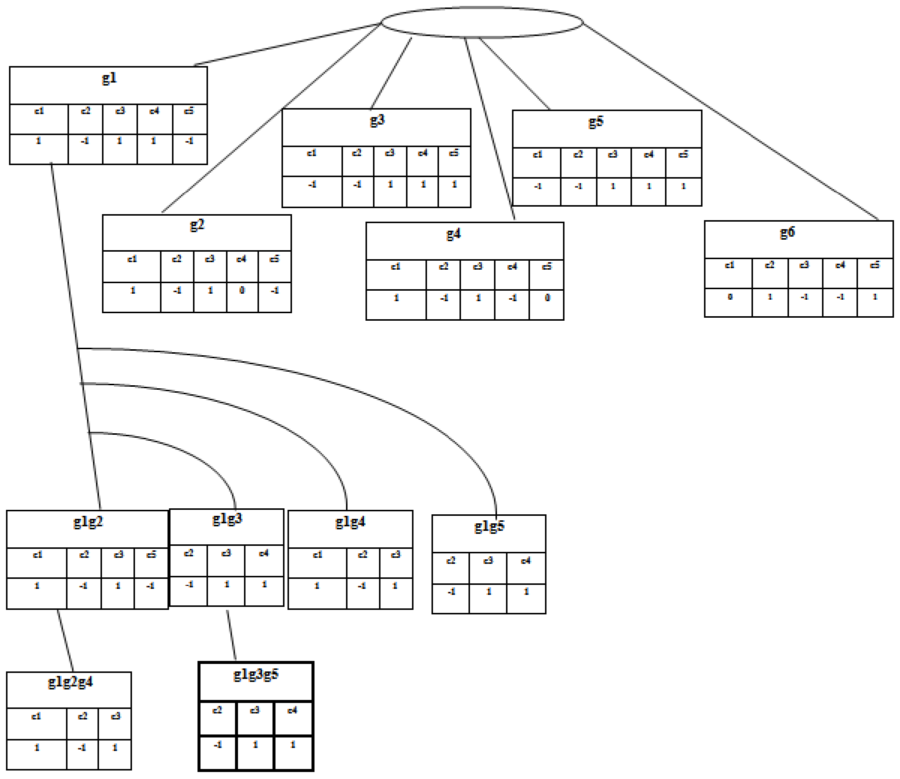


Figure 4: Last level of MBET subtree rooted by the node g_1.

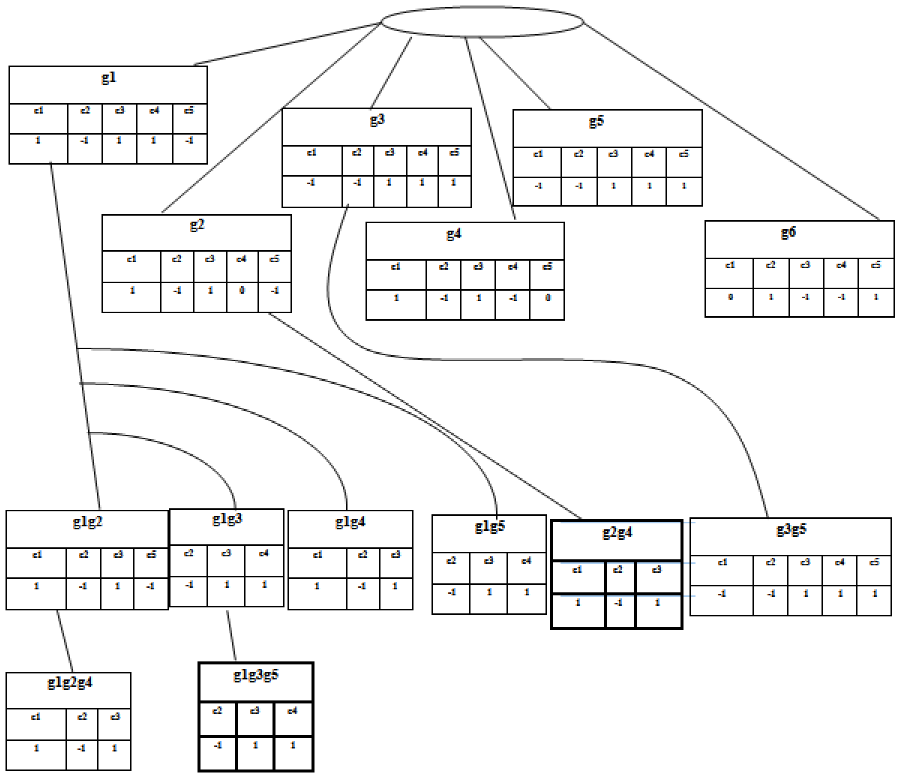


Figure 5: Final MBET: Best biclusters are presented with bold line.

MBET, we keep only nodes whose number of conditions is greater than or equal to 4.

At each constructed subtree, we extract the maximum size bicluster (leaf with a maximum size). In our case, We see that for the subtree rooted by g_1, we have two maximum size leaves with the same volume, i.e., $\{(g_1, g_2, g_4), (c'_1, c'_2, c'_3, c'_4)\}$ and $\{(g_1, g_3, g_5), (c'_2, c'_3, c'_4, c'_5)\}$, hence, we take the last one (Figure 4).

We repeat the same process for the nodes g_2, g_3, g_4 and g_5 in level 1. We do not process g_6 because it has no right brothers.

Finally, each extracted bicluster is assessed using ASR, then we obtain $\mathcal{B} = \{\{(g_1, g_3, g_5),(c'_2, c'_3, c'_4, c'_5)\};\{(g_2, g_4),(c'_1, c'_2, c'_3, c'_4)\}\}$ with ASR respectively equal to 0.86 and 1. This constitutes the best group of biclusters (Figure 5).

3.4. Discussion

BiMine+ is an improvement of *BiMine* (4). It differs from *BiMine* on several features. First, the MBET tree used by *BiMine+* is different from the structure (BET) used by *BiMine*. In fact, BET stores the real values of the genes expression dataset, while MBET stores the genes expression profile shape from the discretized gene expression dataset. This enables to find coherent behaviours of genes regardless of the exact numeric values in the data matrix (37). In fact, the discretization step of the algorithms for time course expression analysis that takes explicitly into account the temporal dependencies between the time-course gene expression profiles should perform better than those that neglect them (19; 37). MBET allows *BiMine+* to explore this property.

Second, the pruning rule used by *BiMine+* to cut MBET branches is different from the one used by *BiMine*. In fact, *BiMine+* cuts all the nodes having a number of conditions lower than a fixed threshold (requiring in the worst case $O(m)$ time) while *BiMine* uses the ASR function as a pruning rule (requiring in the worst case $O(n^2m)$ time). Using ASR for each node of MBET would simply be too time consuming because in the worst case we have 2^n nodes in MBET to calculate. In practice, *BiMine+* is less time consuming than *BiMine* while outperforming *BiMine*.

Finally, *BiMine* considers almost all the leaves of each subtree of BET as extracted biclusters. Since several biclusters may share same genes, *BiMine* may lead to overlapped biclusters. For this reason, *BiMine+* extracts only one maximum size bicluster for every subtree to reduce biclusters overlap and to maximize the volume of each obtained bicluster.

4. Experimental Studies

In this section, we assess the $BiMine+$ algorithm on both synthetic and real DNA microarray data. For the synthetic data, we compare $BiMine+$ results with the results of the original *BiMine* (4), BILS (5) and *BicFinder* (6), and some prominent biclustering algorithms used by the community, namely, CC (16), OPSM (9), ISA (10) and *Bimax* (44). For these reference algorithms, we have used Biclustering Analysis Toolbox (BicAT), a recent software platform for clustering-based data analysis that integrates all these biclustering algorithms (8). For the real datasets, in addition to the algorithms mentioned before, we compare our algorithm with the results of Samba (48), MOEA (40) and EA FRAMEWORK (12).

For our experiments, BILS needs an initial bicluster as its starting point. This initial bicluster can be provided by any means. For instance, one can generate a bicluster randomly, but this may lead to an initial solution of bad quality. A more interesting strategy is to employ a fast greedy algorithm to obtain rapidly a bicluster of reasonable quality. We use this strategy in this work and adopt two well-known algorithms: CC and OPSM.

The *BiMine+* algorithm was implemented in Java and was run on a PC Intel Core 2 Duo T6400 with 2.0GHz CPU and 3.5Gb RAM.

4.1. Synthetic Data

4.1.1. Data, comparison criteria and experimental settings

Datasets: Following (13; 14; 50), we have generated two types of synthetic datasets of size (I,J)= (200, 20). These datasets contains constant, additive, multiplicative and coherent evolution biclusters (38). The first (resp. second) type of dataset contains biclusters without (resp. with) overlap of biclusters. To obtain statistically stable results, for each type of datasets, we have generated 10 problem instances by randomly inserting the biclusters at different places without altering the original column order of the data matrix. Each embedded bicluster has contiguous genes/conditions.

The objective of this experiment is to determine if an algorithm is able to extract exactly all the embedded biclusters.

Comparison Criteria: Following (14), we have used two ratios given below to evaluate our biclustering algorithm:

$$\theta_{Shared} = \frac{S_{cb}}{Tot_{size}} * 100 \tag{3}$$

with
S_{cb} = Portion size of biclusters correctly extracted
Tot_{size} = Total size of correct biclusters

$$\theta_{NotShared} = \frac{S_{ncb}}{Tot_{size}} * 100 \tag{4}$$

with
S_{ncb} = Portion size of biclusters not correctly extracted
Tot_{size} = Total size of corrected biclusters

The ratio θ_{Shared} (resp. $\theta_{NotShared}$) expresses the percentage of shared (resp. not shared) biclusters volume which corresponds (resp. not corresponds) with the real biclusters. In fact, when θ_{Shared} (resp. $\theta_{NotShared}$) is equal to 100% the algorithm extracts the correct (resp. not correct) biclusters. A perfect solution has θ_{Shared} =100% and $\theta_{NotShared}$=0% representing, thus, the exact number of genes and conditions of implanted biclusters.

Protocol for Experiments: We have fixed, in this experimental study, δ at 5 (resp. 6) and β at 0.1 for biclusters without (resp. with) overlap. For the four reference algorithms, we used the default values for different parameters as used in (35). We run all the algorithms and we select the 4 biclusters obtained by each algorithm which best fit the 4 real biclusters. We compute the θ_{Shared} and the $\theta_{NotShared}$ for each algorithm to show the averaged percentage of volume of the resulting biclusters which is shared and not shared with the real biclusters.

4.1.2. Results

Table 3 shows the best biclusters provided by each algorithm for the first dataset.

As we can see in Table 3, *BiMine* can extract 100% of implanted biclusters with an extra volume that represents 33.03% of implanted biclusters. On the other side, *BiMine+* extracts almost all types of biclusters, i.e., 93.34% of implanted biclusters with an extra volume that represents 39.17%.

In fact, when *BiMine+* constructs MBET, 100% of implanted biclusters are represented, but the strategy of *BiMine+* is to extract the maximum size bicluster for every subtree rooted by a node of the first level to maximize the volume of each obtained bicluster and to avoid highly overlapped biclusters.

Table 3: *BiMine+* results and comparison with other algorithms in synthetic data without overlapped biclusters.

Algorithms	θ_{Shared}	$\theta_{NotShared}$
CC	18.21	36.57
OPSM	46.39	74.42
ISA	39.38	5.31
Bimax	58.18	21.39
BiMine	100	33.03
BiMine+	93.34	39.17
BILS	61.27	76.86
BicFinder	100	36.18

Hence, if several biclusters rooted by the same node exist, then only the biggest is considered, this may not be exactly the implanted bicluster. That is why, *BiMine+* is marginally affected. *BiMine+* outperforms BILS on θ_{Shared} and $\theta_{NotShared}$. However, *BicFinder* is slightly better than *BiMine+*.

On the other hand, the best of the reference algorithms, i.e., *Bimax*, can extract only 58.18% of implanted biclusters with 21.39 % of extra volume. CC uses the MSR function of the selected elements as the biclustering criterion. When the signal of the implanted biclusters is weak, the greedy nature of CC may delete some rows and columns of the implanted biclusters in the beginning of the algorithm and miss the deleted rows and columns in the output biclusters. ISA uses only up-regulated and down-regulated constant expression values in its biclustering algorithm. When coherent biclusters exist, ISA may miss some rows and columns of the implanted biclusters. OPSM seeks only up and down regulation expression values with coherent evolution. Its performance decreases when there exist scenarios constant biclusters. The discretization preprocessing used by *Bimax* cannot identify the elements in the coherent biclusters. Hence, the algorithm cannot find exactly the implanted biclusters.

Table 4 shows the best biclusters provided by each algorithm for the second dataset.

As we can see in Table 4, the results with *BiMine+* present the highest coverage of the correct biclusters. In fact, *BiMine+* (resp. *BiMine*) can extract 89.17 % (resp. 85.35 %) of implanted biclusters with an extra volume that represents 44.16 % (resp. 41.78 %). *BiMine+* outperforms BILS and

Table 4: *BiMine+* results and comparison with other algorithms in synthetic data with overlapped biclusters.

Algorithms	θ_{Shared}	$\theta_{NotShared}$
CC	9.21	47.94
OPSM	42.87	49.31
ISA	23.28	23.97
Bimax	34.07	3.43
BiMine	85.35	41.78
BiMine+	89.17	44.16
BILS	54.92	67.25
BicFinder	79.94	46.11

BicFinder on θ_{Shared} and $\theta_{NotShared}$. The best of the reference algorithms, i.e., OPSM, can extract only 42.87 % of implanted biclusters with 49.31 % of extra volume. To find overlapped biclusters in a given matrix, some algorithms, e.g., CC, need to mask the discovered biclusters with random values which is not necessary for *BiMine+*. ISA and OPSM are sensitive to overlapping biclusters due to the normalization step used in the preprocessing phase. With overlapping biclusters, the expression value range after normalization becomes narrower. Table 4 shows that *BiMine+* is marginally affected by the implanted overlap biclusters compared to other algorithms.

4.2. Real data

The synthetic datasets are always biased regarding the underlying model and only reflect some aspects of biological reality. In this section, we show computational results on two well-known real datasets: Saccharomyces cerevisiae dataset and yeast cell cycle dataset. Missing values are replaced by random ones (16).

To fix the two parameters δ (threshold of conditions number) and β (ASR quality threshold), we use a tuning rule based on the p-value. In fact, the p-value uses a cumulative hypergeometric distribution. It implies the probability of observing the number of genes from a particular Gene Ontology (GO) category, i.e., Biological Process, Molecular Function, Cellular Component, within each bicluster. The probability p for detecting at least q genes, from a particular category within a cluster of size n, is defined as follows:

$$p = 1 - \sum_{i=0}^{q-1} \frac{\begin{pmatrix} g_c \\ i \end{pmatrix} \begin{pmatrix} g_g - g_c \\ n-i \end{pmatrix}}{\begin{pmatrix} g_g \\ n \end{pmatrix}} \tag{5}$$

where g_c is the number of genes within a category and g_g is the number of genes within the genome (49). The p-values are computed for each functional category in each cluster. They are used to evaluate the statistical significance for the genes in each bicluster, which means how well the genes match with the different GO categories. Notice that a smaller p-value, close to 0, is indicative of a better match (49).

To tune δ and β, we first fix one threshold and tune the other, and vice-versa (initially, we set δ to 1 and β to 0). For each experiment, ten values are tested between 0.1 and 1 with a stepwise of 0.1 for β, and $|J|$ values are tested between 1 and $|J|$ with a stepwise of 1 for δ. For each combination, we compute the p-values of the obtained biclusters. We pick the combination with the lowest p-value for the final experiment. The procedure stops when the p-value becomes high.

4.2.1. Saccharomyces cerevisiae dataset

The Saccharomyces Cerevisiae dataset[3] contains the expression levels of 2993 genes under 173 experimental conditions. In order to evaluate the biological relevance of *BiMine+*, we compute the p-values to indicate the quality of the extracted biclusters. Following the same process as in (18; 35; 44), we extract the 100 largest biclusters out of 1441 found on this dataset. These 100 biclusters are obtained after a post-filtering procedure in order to eliminate insignificant and small biclusters like Cheng *et al.* (15). The results of *BiMine+* are compared against *BiMine* and reported scores of *Bimax*, OPSM, ISA, Samba and CC from (44). The idea is to determine whether the set of genes discovered by biclustering algorithms shows significant enrichment with respect to a specific Gene Ontology (GO) annotation. We use the web-tool *FuncAssociate* (11) to evaluate the discovered biclusters. *FuncAssociate* computes the adjusted significance scores for each bicluster. Indeed, the adjusted significance scores assess genes in each bicluster by computing adjusted p-values (p), which indicates how well they match with the different

[3]Available at http://www.tik.ethz.ch/sop/bimax/

GO categories. For this experiment, the two parameters of *BiMine+* δ and β are set to 9 and 0.2. The running time of *BiMine+* on this test was 10 minutes. Note that the running time of *BiMine* was approximatively 5 days. In fact, the tree pruning rule used by *BiMine* consumes much more time than the rule used by *BiMine+*.

Figure 6 shows, for each significant score p (p=5%, 1%, 0.5%, 0.1% and 0.001%) and for each compared algorithm, the percentage of the total extracted biclusters by the algorithm reaching the indicated p-value.

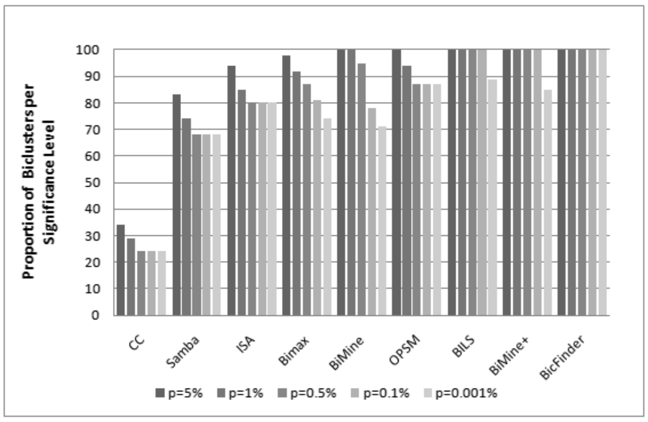


Figure 6: Proportions of biclusters significantly enriched by GO annotations on Saccharomyces Cerevisiae dataset.

From Figure 6, we observe that for each p-value (p=5%, 1%, 0.5% and 0.1%), 100% of extracted biclusters by *BiMine+* reaches the score. For p=0.001%, the percentage drops to 85%. *BicFinder* and BILS are slightly better than *BiMine+* only on p=0.001%, but have the same performance for the other p-values. On the other hand, even the best competing method OPSM cannot achieve such a performance. Indeed, the percentage of the biclusters extracted by OPSM reaching a score of p = 5%, 1%, 0.5% and 0.1% is respectively 100%, 94%, 87%, and 87%. Yet, OPSM performs slightly better for p=0.001 with a percentage of 87% against 85% for *BiMine+*. We also note that globally *BiMine+* performs better for all p-values compared to CC, Samba, ISA and *Bimax*. Finally, *BiMine+* outperforms *BiMine* whose performance is close to Bimax.

4.2.2. Yeast Cell-Cycle dataset

The Yeast Cell-Cycle dataset is described in (49). This dataset is processed in (16) and publicly available from (17). It contains the expression profiles of more than 6000 yeast genes measured at 17 conditions over two complete cell cycles. In our experiments we use 2884 genes selected by (16). For this dataset, three criteria are used. First, we assess the coverage. Second, we evaluate the biological relevance of the extracted biclusters like used for the saccharomyces cerevisiae dataset. Finally, we identify the biological annotations for the obtained biclusters. For this experiment, the two parameters of $BiMine+$ δ and β are experimentally set to 5 and 0.2. Running $BiMine+$ on this dataset leads to a group of 883 biclusters on this dataset. Following (15), a post-filtering procedure is applied to eliminate insignificant and small biclusters and retain 100 largest biclusters. The running time of *BiMine+* on this dataset was 38 minutes (The running time of *BiMine* was approximatively 2 days).

Coverage measurement

To evaluate the performance of $BiMine+$, we compute the total number of cells in the dataset that are covered by the biclusters like used in (12; 40). Our 100 biclusters selected cover 51.76% cells of the initial dataset, while this coverage is 13.36% for *BiMine* (4), 51.34% for MOEA (40) and 50.99% for EA FRAMEWORK (12). The poor coverage of *BiMine* can be explained by its tree pruning rule based on the ASR function. $BiMine+$ avoids this problem and can extract biclusters offering a much better coverage.

Biological relevance

In order to evaluate the biological relevance of $BiMine+$, we use again the p-values and apply the web-tool *FuncAssociate* (11). The results of $BiMine+$ are compared against reported scores of CC, ISA, *Bimax*, OPSM and *BiMine* on this dataset from (4).

Figure 7 shows, for each significant score p (p=5%, 1%, 0.5%, 0.1% and 0.001%) and for each compared algorithm, the percentage of the statistically significant biclusters extracted by the algorithm with the indicated p-value. We observe that $BiMine+$ is highly competitive to the other algorithms on this dataset. 100% of discovered biclusters of $BiMine+$ are statistically significant with p=5%, 1%, 0.5% and 0.1%. Even with $p \leq 0.001\%$, 89% of discovered biclusters of $BiMine+$ are statistically significant against 51% for *BiMine*, 64% for *Bimax*, 86% for BILS and 91% for *BicFinder*.

Analysis of biological annotation enrichment of biclusters

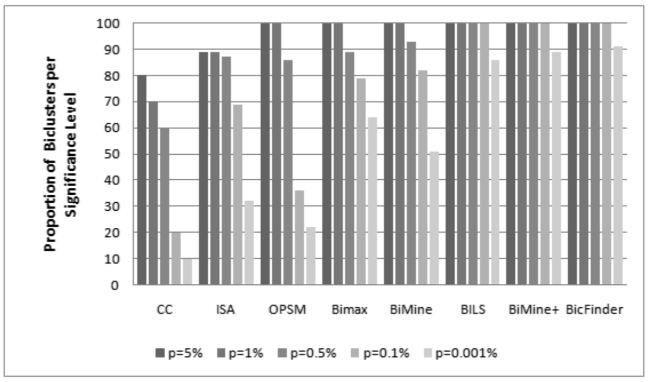


Figure 7: Proportions of Biclusters significantly enriched by GO annotations on Yeast Cell-Cycle dataset.

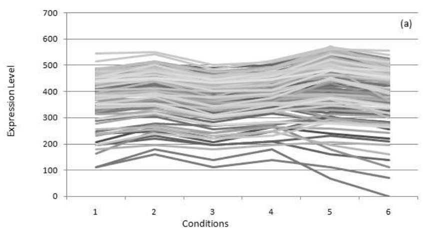
In order to identify the biological annotations for the biclusters, we use GOTermFinder (http://db.yeastgenome.org/cgi-bin/GO/goTermFinder) which is a tool available in the Saccharomyces Genome Database (SGD). GOTermFinder is designed to search for the significant shared GO terms of the groups of genes and provides the user with the means to identify the characteristics that the genes may have in common. We present the significant shared GO terms (or parent of GO terms) used to describe two selected set of genes (extracted by *BiMine+* on yeast cell-cycle dataset) with 136 genes $\times$ 6 conditions and 131 genes $\times$ 7 conditions in each bicluster. These two biclusters have respectively an ASR value equal to 0.24 and 0.73. Following (39), we report the most significant GO terms shared by these biclusters in terms of biological process, molecular function and cellular component. For example, with the first bicluster (Table 5), the genes (YAL059W, YBL072C, YBR048W, YBR181C, YBR189W, YCR031C, YDL083C, YDL208W, YDR025W, YDR064W, YDR418W, YDR447C, YDR450W, YER074W, YER131W, YGR214W, YJR123W, YLR048W, YLR068W, YLR167W, YLR192C, YLR441C, YML026C, YMR143W, YMR230W, YMR269W, YNL096C, YNL112W, YNL302C, YOL040C, YOL127W, YOR056C, YOR293W, YPL090C, YPR102C) are particularly involved in the ribosome biogenesis and ribonucleoprotein complex biogenesis. The values within parentheses after each GO term in Table 5, such as (43.4%, 5.05e-23) in the first bicluster, indicate the cluster frequency and the statistical significance. The cluster frequency (43.4%) shows that out of 136 genes in the first bicluster 59 belong to this process, and the statistical significance is provided

Table 5: Most significant shared GO terms (process, function, component) for two biclusters on yeast cell-cycle dataset.

Biclusters	Biological Process	Molecular function	Cellular component
136 genes × 6 conditions	translation (43.4%,5.05e-23) maturation of SSU-rRNA (14.7%, 1.73e-13) ribosome biogenesis (25.7%, 1.11e-12) maturation of SSU-rRNA from tricistronic rRNA transcript (SSU-rRNA, 5.8S rRNA, LSU-rRNA) (14.0%, 1.78e-12) ribonucleoprotein complex biogenesis(25.7%, 6.44e-11)	structural constituent of ribosome (36.8%, 1.84e-39) structural molecule activity (36.8%, 6.09e-30)	cytosolic ribosome (39.0%, 5.98e-50) cytosolic part (37.5%, 8.78e-43) ribosome (42.6%, 5.99e-39) ribosomal subunit (36.8%, 3.79e-38) cytosolic small ribosomal subunit (19.9%, 9.20e-29)
131 genes × 7 conditions	DNA-dependent DNA replication (12.2% ,2.42e-09) DNA strand elongation (8.4% , 4.03e-09) DNA strand elongation during DNA replication (8.4% , 4.03e-09) lagging strand elongation (6.9%, 9.90e-09) DNA metabolic process (22.1%, 2.74e-08) DNA replication (13.0%, 7.38e-08)	double-stranded DNA binding (5.3%, 0.00035) structure-specific DNA binding (6.1% , 0.00276) structural constituent of ribosome (10.7%, 0.00612)	replication fork (11.5% , 1.65e-12) nuclear replication fork (9.9% , 4.04e-11) non-membrane-bounded organelle (40.5%, 8.17e-10) intracellular non-membrane-bounded organelle (40.5%, 8.17e-10)

by a p-value of 5.05e-23 (highly significant).

In microarray data analysis, genes are considered to be in the same cluster if their curves of genes expression levels are similar across a set of conditions (36; 42; 45). In Figure 8, we show the two biclusters of Table 5 found by *BiMine+*. From a visual inspection of the biclusters presented, we can notice that the genes do present a similar behaviour under the selected conditions.



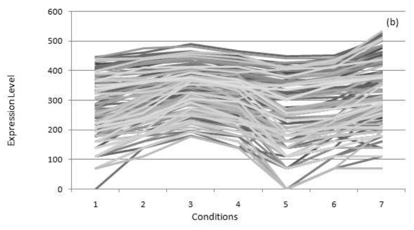


Figure 8: Two Biclusters found by *BiMine+* on yeast cell-cycle dataset: (a) 136 genes × 6 conditions with ASR = 0.24 (b) 131 genes × 7 conditions with ASR = 0.73.

All these experiments tend to suggest that the proposed approach is able to detect biologically significant and functionally enriched biclusters with low p-value.

5. Conclusion

In this paper, we have proposed a novel enumeration algorithm, called *BiMine+*, for biclustering of gene expression data. *BiMine+* is designed to extract coherent and maximum size biclusters with little overlap. The performances of the *BiMine+* algorithm is assessed on synthetic datasets as well as two real DNA microarray datasets. Computational experiments show highly competitive results of *BiMine+* in comparison with our three biclustering algorithms (*BiMine*, BILS and *BicFinder*) and other popular biclustering algorithms. Comparative study shows that the *BiMine+* algorithm can find statistical and biological significant biclusters.

Acknowledgement

We are grateful to the reviewers for their careful reviews and highly helpful comments. The work is partially supported by the Region "Pays de La Loire" (France) *via* the following projects: Bioinformatique Ligrienne, Radapop and LigeRO.

References

[1] J. S. Aguilar-Ruiz. Shifting and scaling patterns from gene expression data. *Bioinformatics*, 21:3840–3845, 2005.

[2] F. Angiulli, E. Cesario, and C. Pizzuti. Random walk biclustering for microarray data. *Journal of Information Sciences*, pages 1479–1497, 2008.

[3] W. Ayadi and M. Elloumi. *Algorithms in Computational Molecular Biology: Techniques, Approaches and Applications*, chapter Biclustering of Microarray Data, pages 651–664. Wiley Book Series on Bioinformatics : Computational Techniques and Engineering, Wiley-Blackwell, John Wiley & Sons Ltd., New Jersey, USA (Publish.), 2011.

[4] W. Ayadi, M. Elloumi, and J. K. Hao. A biclustering algorithm based on a bicluster enumeration tree: application to dna microarray data. *BioData Mining*, 2(1):9, 2009.

[5] W. Ayadi, M. Elloumi, and J.K. Hao. Iterated local search for biclustering of microarray data. In *Proceedings of 5th IAPR International Conference on Pattern Recognition in Bioinformatics. Volume 6282 of Lecture Notes in Computer Science*, pages 219–229. Springer-Verlag, 2010.

[6] W. Ayadi, M. Elloumi, and J.K. Hao. Bicfinder: a biclustering algorithm for microarray data analysis. *Knowledge and Information Systems: An International Journal*, 30:341–358, 2012.

[7] Z. Bar-Joseph. Analyzing time series gene expression data. *Bioinformatics*, 20(16):2493–2503, 2004.

[8] S. Barkow, S. Bleuler, A. Prelic, P. Zimmermann, and E. Zitzler. Bicat: a biclustering analysis toolbox. *Bioinformatics*, 22(10):1282–1283, 2006.

[9] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. In *Proceedings of the sixth annual international conference on Computational biology*, pages 49–57, New York, NY, USA, 2002. ACM.

[10] S. Bergmann, J. Ihmels, and N. Barkai. Defining transcription modules using large-scale gene expression data. *Bioinformatics*, 20(13):1993–2003, 2004.

[11] G.F Berriz, O.D. King, B. Bryant, C. Sander, and F.P. Roth. Characterizing gene sets with funcassociate. *Bioinformatics*, 19(18):2502–2504, 2003.

[12] S. Bleuler, A. Prelic, and E. Zitzler. An ea framework for biclustering of gene expression data. In *Proceedings of Congress on Evolutionary Computation*, pages 166–173, 2004.

[13] K. Bryan, P. Cunningham, and N. Bolshakova. Application of simulated annealing to the biclustering of gene expression data. In *IEEE Transactions on Information Technology on Biomedicine, 10(3)*, pages 519–525, 2006.

[14] C. Cano, L. Adarve, J. Lopez, and A. Blanco. Possibilistic approach for biclustering microarray data. In *Computers in Biology and Medicine, 37*, pages 1426–1436, 2007.

[15] K.O. Cheng, N.F. Law, W.C. Siu, and A.W. Liew. Identification of coherent patterns in gene expression data using an efficient biclustering algorithm and parallel coordinate visualization. *BMC Bioinformatics*, 9(210):1282–1283, 2008.

[16] Y. Cheng and G. M. Church. Biclustering of expression data. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 93–103. AAAI Press, 2000.

[17] Y. Cheng and G.M. Church. Biclustering of expression data. Technical report, (supplementary information), 2006.

[18] Y. Christinat, B. Wachmann, and L. Zhang. Gene expression data analysis using a novel approach to biclustering combining discrete and continuous data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 5(4):583–593, 2008.

[19] I.G. Costa, A. Schonhuth, and A. Schliep. The graphical query language: A tool for analysis of gene expression time-courses. *Bioinformatics*, 21(10):2544–2545, 2004.

[20] S. Das and S.M. Idicula. Application of reactive grasp to the biclustering of gene expression data. In *Proceedings of the International Symposium on Biocomputing*, pages 1–8, New York, NY, USA, 2010. ACM.

[21] A. Dharan and A.S. Nair. Biclustering of gene expression data using reactive greedy randomized adaptive search procedure. *BMC Bioinformatics*, 10(Suppl 1):S27, 2009.

[22] F. Divina and J.S. Aguilar-Ruiz. A multi-objective approach to discover biclusters in microarray data. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 385–392, New York, NY, USA, 2007. ACM.

[23] Béatrice Duval and Jin-Kao Hao. Advances in metaheuristics for gene selection and classification of microarray data. *Briefings in Bioinformatics*, 11(1):127–141, 2010.

[24] S. Erdal, O. Ozturk, D. Armbruster, H. Ferhatosmanoglu, and W.C. Ray. A time series analysis of microarray data. In *Proceedings of the Fourth IEEE Symp. Bioinformatics and Bioeng*, pages 366–374, 2004.

[25] C.A. Gallo, J.A. Carballido, and I. Ponzoni. Microarray biclustering: A novel memetic approach based on the pisa platform. In *Proceedings of the 7th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, pages 44–55, Berlin, Heidelberg, 2009. Springer-Verlag.

[26] J. Guan, Y. Gan, and H. Wang. Discovering pattern-based subspace clusters by pattern tree. *Knowledge-Based Systems*, 22(8):569–579, 2009.

[27] L. Han and H. Yan. Hybrid method for the analysis of time series gene expression data. *Knowledge-Based Systems, In Press, Accepted Manuscript, Available online 12 April 2012*.

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

[29] L. Ji and K. Tan. Mining gene expression data for positive and negative co-regulated gene clusters. *Bioinformatics*, 20(16):2711–2718, 2004.

[30] L. Ji and K. Tan. Identifying time-lagged gene clusters using gene expression data. *Bioinformatics*, 21(4):509–516, 2005.

[31] M. Khashei, A.Z. Hamadani, and M. Bijari. A fuzzy intelligent approach to the classification problem in gene expression data analysis. *Knowledge-Based Systems*, 27:465–474, 2012.

[32] A. Kwon, H. Hoos, and R. Ng. Inference of transcriptional regulation relationships from gene expression data. *Bioinformatics*, 19(8):905–912, 2003.

[33] E.L. Lehmann and H.J.M. D'Abrera. *Nonparametrics: Statistical Methods Based on Ranks*, pages 292–323. rev. ed. Englewood Cliffs, NJ: Prentice-Hall, 1998.

[34] J. Liu and W. Wang. Op-cluster: Clustering by tendency in high dimensional space. *IEEE International Conference on Data Mining*, pages 187–194, 2003.

[35] X. Liu and L. Wang. Computing the maximum similarity bi-clusters of gene expression data. *Bioinformatics*, 23(1):50–56, 2007.

[36] Y. Luan and H. Li. Clustering of time-course gene expression data using a mixed-effects model with b-splines. *Bioinformatics*, 19:474–482, 2003.

[37] S. C. Madeira, M. C. Teixeira, I. Sa-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*, 7(1):153–165, 2010.

[38] Sara C. Madeira and Arlindo L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

[39] U. Maulik, A. Mukhopadhyay, and S. Bandyopadhyay. Combining pareto-optimal clusters using supervised learning for identifying co-expressed genes. *BMC Bioinformatics*, 10:27, 2009.

[40] S. Mitra and H. Banka. Multi-objective evolutionary biclustering of gene expression data. *Pattern Recogn.*, 39(12):2464–2477, 2006.

[41] C. Mollër-Levet, S. Cho, and O. Wolkenhauer. dna microarray data clustering based on temporal variation: Fcv and tsd preclustering. *Applied Bioinformatics*, 2(1):35–45, 2003.

[42] S.D. Peddada, E.K. Lobenhofer, L. Li, C.A. Afshari, C.R. Weinberg, and D.M. Umbach. Gene selection and clustering for time-course and dose-response microarray experiments using order-restricted inference. *Bioinformatics*, 19:834–841, 2003.

[43] B. Pontes, F. Divina, R. Giráldez, and J.S. Aguilar-Ruiz. Virtual error: A new measure for evolutionary biclustering. In *Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, pages 217–226, 2007.

[44] A. Prelic, S. Bleuler, P. Zimmermann, P. Buhlmann, W. Gruissem, L. Hennig, L. Thiele, and E. Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, 2006.

[45] A. Schliep, A. Schonhuth, and C. Steinhoff. Using hidden markov models to analyze gene expression time course data. *Bioinformatics*, 19:i255–i263, 2003.

[46] D. J. Slotta. *Evaluating Biological Data Using Rank Correlation Methods*. PhD thesis, Faculty of the Virginia Polytechnic Institute and State University, May 2005.

[47] R.B. Stoughton. Applications of dna microarrays in biology. *Annual Review of Biochemistry*, 74:53–82, 2005.

[48] A. Tanay, R. Sharan, and R. Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18:S136–S144, 2002.

[49] 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.

[50] L. Teng and L. Chan. Discovering biclusters by iteratively sorting with weighted correlation coefficient in gene expression data. *J. Signal Process. Syst.*, 50(3):267–280, 2008.

[51] J. Yang, H. Wang, W. Wang, and P. Yu. Enhanced biclustering on expression data. In *Proceedings of the 3rd IEEE Symposium on BioInformatics and BioEngineering*, pages 321–327, Washington, DC, USA, 2003. IEEE Computer Society.

[52] Z. Zhang, A. Teo, B.C. Ooi, and K.L. Tan. Mining deterministic biclusters in gene expression data. *IEEE International Symposium on Bioinformatic and Bioengineering*, pages 283–290, 2004.