Biclustering of Gene Expression Data Based on Local Nearness

Jesus S. Aguilar-Ruiz*, Domingo Savio Rodriguez**
Dan A. Simovici***

*BIGS BioInformatics Group Seville, University of Seville, Spain
aguilar@lsi.us.es,
**BIGS BioInformatics Group Seville, University of Seville, Spain
dsavio@lsi.us.es,
***Univ. of Massachusetts Boston, Massachusetts 02125, USA
dsim@cs.umb.edu

Résumé. L'analyse des données d'expression de génes dans les fragments d'ADN est un outil important utilisé dans la recherche genomique dont les objectifs principaux s'étendent de l'étude du caractére fonctionnel des génes spécifiques et leur participation dans les processus biologiques à la reconstruction de conditions des maladies et leur pronostique. Les données d'expression des génes sont arrangées dans les matrices où chaque géne corresponde à une ligne et chaque colonne représente une condition expérimentale spécifique. Les techniques de biclustering ont le but de trouver de sous-ensembles de génes qui montrent les modéles d'activité similaires sous un sous-ensemble des conditions. Notre approche consiste en un algorithme de biclustering basé sur la proximité locale. L'algorithme cherche biclusters dans une manière greedy, commençant avec biclusters qui contiennent deux génes et incluant autant de génes que possible dépendant d'un seuil de distance qui garantit la similarité de comportements des génes.

1 Introduction

The DNA Microarray technology represents a great opportunity of studying the genomic information as a whole, so we can analyze the relations among thousands of genes simultaneously. The experiments carried out on genes under different conditions produce the expression levels of their transcribed mRNA and this information is stored in DNA chips.

A *bicluster* is a subset of genes that show similar activity patterns under a subset of conditions. The research on biclustering started in 1972 with Hartigan's work, in which the way of dividing a matrix in sub–matrices with the minimum variance was studied (Hartigan *et al.*, 1972). In that approach the perfect bicluster was the submatrix formed by constant values, i.e., with variance equal to zero. Hartigan's algorithm, named *direct clustering*, divides the data matrix into a certain number of biclusters, with the minimum variance value, so the fact of finding a number of sub-matrices equal to the number of elements of the matrix is avoided. Another way of searching biclusters is to measure the coherence between their genes and conditions.

Cheng & Church (Cheng *et al.*, 2000) introduced a measure, the *mean squared residue* (MSR), that computes the similarity among the expression values within the bicluster. The ideas of Cheng and Church were further developed by Yang (Yang *et al.*, 2002, 2003) who dealt with missing values in the matrices. As a result of this approach an algorithm named FLOC was designed. Other works (e.g (Wang *et al.*, 2002)) are based in a quality value as well, calculated using the expression values of biclusters, so to measure their coherence.

Other alternatives in the searching for biclusters have been studied in the last years. We might also consider that a value in the data matrix is the sum of the contributions of different biclusters. Based on the previous idea, Lazzeroni (Lazzeroni *et al.*, 2000) presents the *plaid models*, in which the data matrix is described as a linear function of layers corresponding to its biclusters and shows how to estimate a model using an iterative maximization process. Shamir (Shamir *et al.*, 2002) proposes a new method to obtain biclusters based on a combination of graph theoretical and statistical modelling of data. In this model, a gene responds to a condition if its expression level changes significantly at that condition witch respect to its normal level. In a recent work (Liu *et al.*, 2004), a generalization of OPSM model, introduced by (Ben *et al.*, 2002), is presented. The OPSM model is based on the search of biclusters in which a subset of genes induce a similar linear ordering along a subset of conditions. Some techniques search for specific structures in data matrix to find biclusters : (Gerstein *et al.*, 2003) creates a method for clustering genes and conditions simultaneously based on the search of "checkerboard" patterns in matrices of gene expression data. Previously the data is processed by normalization in a spectral framework model (several schemes all built around the idea of putting the genes on the same scale so that they have the same average level of expression across conditions, and the same for the conditions). Evolutionary computation techniques have also been used in this research area. These techniques use aspects from natural selection within computer science, including genetic algorithms, genetic programming and evolutionary strategies. In (Aguilar *et al.*, 2005) an evolutionary technique, based on the search of biclusters following a sequential covering strategy and measuring the mean squared residue, is used.

In this work we propose an algorithm to obtain high quality biclusters based on local nearness, i.e., biclusters with the maximum number of genes and in which the absolute value of the difference between two expression values of any pair of genes under the same condition is not greater than a certain value. Therefore, we consider that two genes whose distance between each other is lower than a threshold with respect to certain experimental conditions, have similar behavior regarding those conditions. To find the appropriate distance threshold, we carried out a preliminary statistical study in order to allow the algorithm to discover high–quality biclusters. The quality is further evaluated by means of the mean squared residue, so that a comparative analysis with other techniques is possible.

The paper is organized as follows : in Section 2 the definitions related to biclustering are presented ; the algorithm is shown in Section 3 ; in Section 4, we describe the method used and illustrate it with a simple example ; experimental results are discussed in Section 5, comparing the quality of those generated by Cheng & Church's and Aguilar & Divina's algorithms ; finally, the most interesting conclusions and future research directions are summarized in Section 6.

2 Definitions

The gene expression data are arranged in matrices. A matrix is defined as a triple $M = (G, C, \ell)$, where G, C are two finite sets referred to as the *the set of genes* and *the set of conditions* respectively, and $\ell : G \times C \longrightarrow \Re$ is the *level* function. The real number $\ell(g, c)$ is also denoted by $\langle g, c\rangle$ and represents the level of expression of the gene g under the specific condition c.

Next the main concepts of our approach are defined and some examples to clarify them are provided.

Definition 1 *Let $M = (G, C, \ell)$ be a matrix formed by a set of genes, G, and a set of conditions,C. We say that a pair of non-empty sets (I, J) is a δ–bicluster, if $I \subseteq G$, $J \subseteq C$ and*

$$J = \{c \in C \mid \forall g, g' \in I, \mid \langle g, c\rangle - \langle g', c\rangle \mid \leq \delta\}$$

The absolute value of the difference between two expression values of any pair of genes in the bicluster under the same condition is not greater than a threshold δ.

Definition 2 *Let (I, J) be a bicluster. The residue R of an element a_{ij} of the bicluster (I, J) is*

$$R(a_{ij}) = a_{ij} - a_{iJ} - a_{Ij} - a_{IJ}$$

where a_{iJ} is the mean of the i*th row in the bicluster, a_{Ij} the mean of the* j*th column in the bicluster, and a_{IJ} is the mean of all the elements within the bicluster.*

The mean squared residue MSR of a bicluster (I, J) is

$$MSR(I, J) = \frac{1}{\mid I \mid\mid J \mid} \sum_{i \in I, j \in J} R^2(a_{ij})$$

The MSR is the variance of the set of all the values in the bicluster, plus the mean row variance and the mean column variance. This value is indicative of the coherence of values across both rows and columns. The lower the MSR is, the stronger the coherence.

Example 1 *Consider the matrix $M = (G, C, \ell)$, where $G = \{g_1, g_2, g_3, g_4, g_5\}$, $C = \{c_1, c_2, c_3, c_4\}$ and ℓ is defined by :*

$$(\ell(g_i, c_j)) = \begin{pmatrix} 1 & 9 & 3 & 2 \\ 7 & 8 & 1 & 4 \\ 1 & 6 & 2 & 2 \\ 5 & 1 & 5 & 7 \\ 2 & 1 & 3 & 1 \end{pmatrix}$$

If the threshold δ is 5, some of the biclusters are $B_1 = \{\{g_1, g_3, g_4\}, \{c_1, c_3, c_4\}\}$, $B_2 = \{\{g_1, g_2, g_3\}, \{c_2, c_3, c_4\}\}$, and $B_3 = \{\{g_3, g_4, g_5\}, \{c_1, c_2, c_3\}\}$, where

$$B_1(g_i, c_j) = \begin{pmatrix} 1 & 3 & 2 \\ 1 & 2 & 2 \\ 5 & 5 & 7 \end{pmatrix} \quad B_2(g_i, c_j) = \begin{pmatrix} 9 & 3 & 2 \\ 8 & 1 & 4 \\ 6 & 2 & 2 \end{pmatrix} \quad B_3(g_i, c_j) = \begin{pmatrix} 1 & 6 & 2 \\ 5 & 1 & 5 \\ 2 & 1 & 3 \end{pmatrix}$$

In all biclusters, the absolute value of the difference among the expression values of every pair of genes is lower than or equal to the threshold $\delta = 5$, under the same condition.

Our goal is to obtain biclusters with the maximum number of genes and conditions and with the minimum value of MSR.

3 Algorithm

Our approach, named BLN (Biclustering by Local Nearness), is based on local nearness, i.e., in the definition of δ–bicluster provided above, in Def. 1. Its goal is to obtain different biclusters with the maximum number of genes so that all of them have the next two properties :

– the difference between two expression values of any pair of these genes under the same condition in the bicluster is not greater than δ ;
– the number of conditions of each bicluster is not lower than λ.

The algorithm, illustrated in Figure 1, consists of two phases. In the first part we obtain a set of valid biclusters with only two genes (*2g_Bicluster*), so that the algorithm analyzes all possible pairs of genes in the data matrix in order to find them, as we can see in Figure 1 (line 9). This first bicluster set will be used by the algorithm in the second part. We have designed a special tree data structure in which those biclusters are stored. In this tree, the nodes represent conditions and leaves are groups of genes of *2g_Biclusters* that have a common group of conditions, ie., to reach them we have to follow the same path in the tree. The reason why we use this structure is to minimize the amount of memory used for storing these first biclusters (they can be many thousands) and also to reduce the running time. The aim of the second part

```
1 **Procedure :** BLN
2 **Input :**
3   $M$ (data matrix $M = (G, C, \ell)$)
4   $\delta$ (maximum difference between two expression values)
5   $\lambda$ (minimum length of the set of conditions for every bicluster)
6 **Output :**
7   $B$ (final set of biclusters)
8 **Method :**
9 Initialize $B$ to contain all the biclusters $(I, J)$
    with $\mid I \mid= 2$ and $\mid J \mid\geq \lambda$
10 $S = \emptyset$
11 **repeat**
12   $B' = B - S$
13   Useful_Genes= $\{g \in I \mid \forall (I, J) \in B'\}$
14   $S = B$
15   **for each** bicluster $(I, J) \in B'$
16     **for each** $g' \in$ (Useful_Genes - $I$)
17       **if for all** $g'' \in I$ there is a bicluster $(\{g', g''\}, J')$ with $J \subset J'$
18         Add $(I \cup \{g'\}, J)$ to $B$
19 **until** $\mid B \mid=\mid S \mid$
20 **end** BLN
```

FIG. 1 – *The BLN algorithm.*

is to create new biclusters containing more than two genes. This part is iterative and follows a greedy methodology to prune the search space. At each iteration a group of new biclusters is created. The process ends when no new bicluster is obtained (line 19, the number of biclusters has not altered). In every iteration, the set of biclusters that has been obtained in the previous one is analyzed, as we can see in line 15 (in the first iteration BLN starts working with the biclusters obtained in the first phase). To create a new bicluster from another one we have to add it new genes. These new genes are gathered from a set called *Useful Genes*, which is formed by the genes from all the biclusters of the set being analyzed minus the genes from the bicluster which is being processed at that moment (lines 16 to 18). To add one gene, *g'*, to a bicluster (I, J), BLN has to check that for every gene in the bicluster, *g"* , a *2g_Bicluster* formed by *g'* and *g"* exists and its group of associated conditions has to be greater than or equal to the conditions of J, i.e., the conditions of the new possible bicluster J' includes that of the former I (line 17). In this way, BLN ensures that the difference between the expression value of the new gen and the others is always lower than δ.

A simple example based on the data matrix $M = (G, C, \ell)$ described in Example 1. We apply BLN to M using $\delta = 5$ and $\lambda = 2$. After the algorithm's first phase we have obtained a group of 2-gene biclusters shown in Table 1. The second part will analyze every bicluster of

TAB. 1 – *Example of* 2g_Biclusters *generation.*

$Genes$	$Conditions$	$Genes$	$Conditions$
$\{1,2\}$	$\{2,3,4\}$	$\{2,4\}$	$\{1,3,4\}$
$\{1,3\}$	$\{1,2,3,4\}$	$\{2,5\}$	$\{1,3,4\}$
$\{1,4\}$	$\{1,3,4\}$	$\{3,4\}$	$\{1,2,3,4\}$
$\{1,5\}$	$\{1,3,4\}$	$\{3,5\}$	$\{1,2,3,4\}$
$\{2,3\}$	$\{2,3,4\}$	$\{4,5\}$	$\{1,2,3\}$

that group trying to generate new ones. We choose one of this preliminary bicluster : $B_1 = \{\{g_1, g_5\}, \{c_1, c_3, c_4\}\}$

$$B_1(g_i, c_j) = \begin{pmatrix} 1 & 3 & 2 \\ 2 & 3 & 1 \end{pmatrix}$$

In this case the group of $Useful\_Genes$ will be $\{g2, g3, g4\}$. For every gene of this set we try to create a new bicluster checking if it can be part of the bicluster we are processing, B_1. We start with $g2$, and as the biclustering being considered is $\{g1, g5\}$, we check if $\{g2, g1\}$ and $\{g2, g5\}$ are biclusters, and then if these biclusters have the right conditions to include $g2$ in $\{g1, g5\}$.

$Genes$	$Conditions$
$\{2,1\}$	$\{2,3,4\}$
$\{2,5\}$	$\{1,3,4\}$
{1,5}	**{1,3,4}**

The first combination of genes, $(g2, g1)$, generates a bicluster with a group of conditions incompatible with the conditions of B_1, so we cannot add $g2$ to the bicluster. The same happens with the gene $g4$.

$Genes$	$Conditions$
$\{4,1\}$	$\{1,3,4\}$
$\{4,5\}$	$\{1,2,3\}$
{1,5}	**{1,3,4}**

However, the gene $g3$ is appropriate to be included in the bicluster.

$Genes$	$Conditions$
$\{3,1\}$	$\{1,2,3,4\}$
$\{3,5\}$	$\{1,2,3,4\}$
{1,5}	**{1,3,4}**

The group of conditions $\{c1, c2, c3, c4\}$ is compatible with the conditions of $B_1 : \{c1, c3, c4\}$ for every pair of genes formed by each gene of B_1 and the gene $g3$ of the $Useful\_Genes$ set. As a result, a new bicluster is created, $Bnew = \{\{g_1, g_3, g_5\}, \{c_1, c_3, c_4\}\}$, where

$$(Bnew(g_i, c_j) = \begin{pmatrix} 1 & 3 & 2 \\ 1 & 2 & 2 \\ 2 & 3 & 1 \end{pmatrix}$$

4 Method

In this section we describe the method used to obtain biclusters based on local nearness using the algorithm BLN. The aim is to generate the greatest number of biclusters with the maximum number of genes and a low value of MSR. To do this we have to determine first the correct value of the parameters of our algorithm, paying special attention to the distance threshold between expression values : δ. As this value is critical, we have designed carefully the method in order to provide valid and high–quality biclusters.

We have developed our experiments with a well known dataset : the *Saccharomyces Cerevisiae* cell cycle expression dataset. The *Yeast* dataset consist of a data matrix composed by 2884 genes (rows) and 17 experimental conditions (columns). We have to gathered statistical information about this data matrix to decide which parameters are suitable for our purposes. If we analyze the *yeast* data matrix we find that the maximum distance between expression values is 596. We consider the half of that value, that is, 298, as the maximum distance that we are going to allow, δ_max. We divided this value by 100, creating 100 different intervals for δ. The next step is to run an special version of BLN which goal is to obtain statistical information about the number of biclusters generated after the first and second parts. This version carried out a test for every $\delta : 2, 98 * i$, with i ranging from 1 to 100 and for every number of minimum conditions allowed, λ, from 1 to 17. These runs are only a simulation so the computational cost is lower than a normal BLN run. We recorded the number of biclusters generated after the first and second phases in $100 * 17$ different situations. Also, we compiled information about the mean of the number of genes and the maximum number of genes after the second part. To reduce this amount of data we focus on a minimum number of conditions ranging from 10 to 17.

The number of bicluster generated with these tests values are illustrated in the graphics of Fig. 2. The first graph (left) shows the evolution of the number of different biclusters created

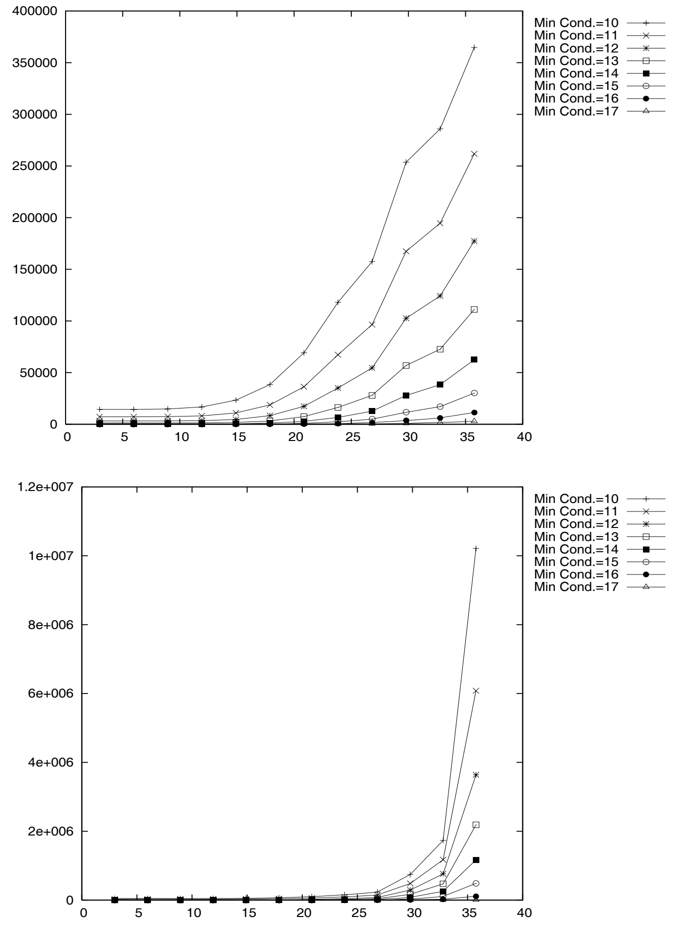

FIG. 2 – *Number of biclusters with different values of δ (in X–axis) and λ (curves identify the minimum number of conditions) : after the first phase of BLN (left) and after the second phase of BLN (right), respectively. When the threshold δ increases, the number of biclusters also does. When the minimum number of conditions increases, the number of biclusters decreases.*

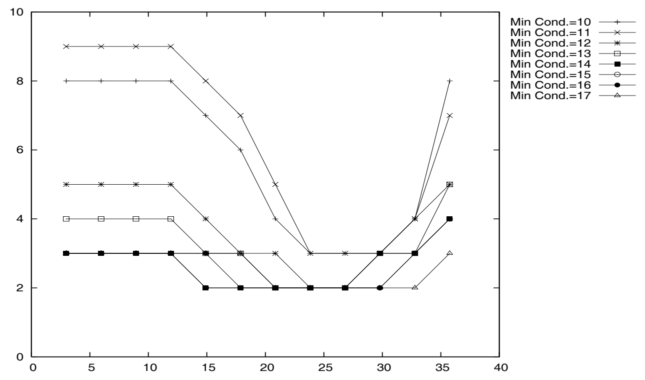

FIG. 3 – *Mean of the number of biclusters with different values of δ (X–axis) and λ.*

for every test values after the first phase, with a maximum number of biclusters of 364.713 with $\delta = 35.76$ and $\lambda = 10$ and a minimum number of biclusters of 10 with $\delta = 2.98$ and $\lambda = 17$. The second graph (right) shows the same information with respect to the second part of the algorithm, generating the next extreme values : 10.215.022 with $\delta = 35.76$ and $\lambda = 10$, and 21 with $\delta = 2.98$ and $\lambda = 17$. Obviously, the restrictive conditions that make the number of biclusters lower are a low δ value and high value of λ. In fact, if λ is equal to the number of experimental conditions, then we would find clusters instead of biclusters, although allowing overlapping among clusters, in contrast with traditional clustering techniques. It is worth to note that for every pair of test values, δ_i and λ_i, the number of biclusters generated after the first and second parts are related, in such a way that the last one is directly proportional to the first one in all the cases. We gathered information about the number of genes, although only for the second part of the process (in the first part we only generate *2g_Biclusters*).

In Figure 3 we can observe the average number of genes of the biclusters for our test values. It is interesting that the mean of genes decreases whereas the value of δ grows for all the minimum number of conditions until the value 25. This is because the number of biclusters increases with δ while the number of their genes does not change. From the moment δ has the value of 25 the number of genes in the biclusters grows and the mean as well. From $\delta = 34$, the mean starts increasing. For this reason, we have chosen the value 35.76 for our experiments, as there are more choices to select good biclusters. With $\delta = 35.76$ and $\lambda = 10$ the mean of genes reaches to its biggest value.

We can see the evolution of the maximum number of genes in the biclusters in Figure 4. This value is constant for all the test values until the δ value is about 27. From this moment, we find biclusters with more genes. The number of genes reaches its maximum value, 17, with $\delta = 35.76$ and $\lambda = 11$ or $\lambda = 10$ (both graphs are identical).

The main conclusion from this previous analysis is that the maximum number of genes is reached with the value $\delta = 35.76$. The correct value for λ is 10 or 11. We finally choose 10 as the minimum number of conditions allowed because with $\delta = 35.76$ and $\lambda = 10$ the algorithm generates more biclusters than with $\lambda = 11$. Once the correct parameters have been chosen the next step is to run BLN and study the results.

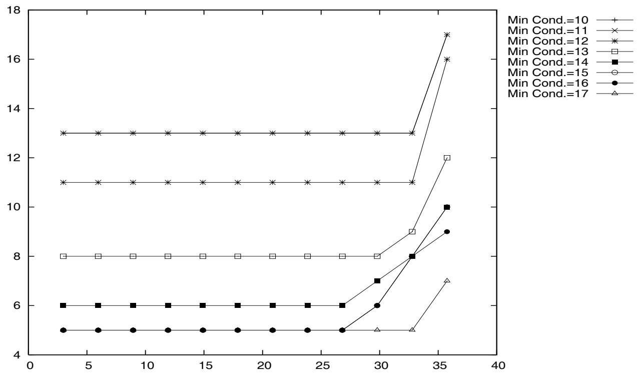

FIG. 4 – *Maximum number of genes in biclusters varying the values of δ (X–axis) and λ.*

TAB. 2 – *Information about biclusters generated by BLN. In the first column every bicluster is identified by its generation number. The second and third columns show the number of genes and conditions, respectively. The MSR value is reported in fourth column and the last column is related to the row variance.*

Bicluster	Genes	Conditions	MSR	Row Variance
433315	17	10	105.87	231.52
462795	15	10	95.69	166.08
548356	15	10	81.56	173.35
682571	16	12	87.10	292.20
755312	14	10	69.85	153.78
847161	14	10	91.65	355.04
1031000	14	10	89.62	159.25
8604074	14	10	73.84	127.69
9509611	15	10	73.93	182.27

5 Experimental Results

We compare BLN with the Cheng and Church algorithm (CC) and with the SEBI algorithm which is an evolutionary–based algorithm that extracts biclusters following a sequential covering strategy (Aguilar *et al.*, 2005).

After the analysis done in the previous section, the algorithm BLN is run using $\delta = 35.76$ and $\lambda = 10$. As a result, BLN generates 10.215.022 different biclusters with various gene set sizes. The overlapping between biclusters is obvious, and even many of them are included in others, so only 9.147 biclusters were finally generated containing a number of genes greater than or equal to 14. The first phase of BLN took 41 seconds. The second phase, 12.200 seconds (9.684 without graphics and genetic files generation). The average number of genes in biclusters was 8. The biclusters generated covered 93.93% of the genes and 100% of the conditions, i.e., 93.93% of the elements of the expression matrix. The BLN algorithm has been implemented in Java and run on a Win–Xp platform.

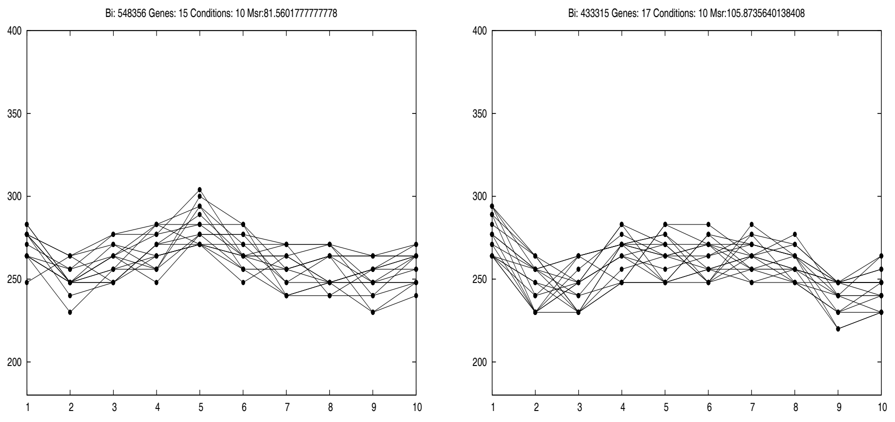

FIG. 5 – *An example of biclusters generated by BLN. The shapes of the graphs show the quality of biclusters.*

TAB. 3 – *Performance comparison between CC, SEBI and BLN. First column (algorithm), second column (average mean squared residue), third column (average volume), fourth column (average number of genes), fifth column (average number of genes) and sixth column (average number of conditions).*

Alg.	MSR	Vol.	Num. of Genes	Num. of Cond.
CC	204.29	1576.98	166.71	12.09
SEBI	202.68	204.67	13.20	15.44
BLN	70.02	166.84	15.34	10.88

The criteria used to measure the quality of biclusters are the maximum number of genes, the minimum mean squared residue, and the maximum number of conditions, in this order. Following these criteria we have selected the best 100 biclusters among all having a number of genes between 14 and 17. The features of some of them can be observed in Table 2.

In Table 2, the biclusters selected present a small MSR value, i.e., it exists a great coherence across both genes and conditions. This similar behavior can be observed in the graphics on Figure 5. These graphics show the evolution of the expression values of the set of genes under the set of conditions. We obtain biclusters with high number of genes as well, being 17 the maximum value. In the *Yeast* data set this number of genes and the MSR value vary with the parameters, i.e., when the distance threshold grows and the minimum number of conditions decreases (less restrictive experimental conditions), then the number of genes increases substantially.

In Table 3 we compare our 100 best biclusters and their average values with those obtained by the algorithms of Cheng & Church (CC) and Aguilar & Divina (SEBI).

BLN obtains better results with regard to the MSR value than the other two algorithms. The averaged volume, i.e., the number of genes multiplies by the number of conditions, and the average of conditions in biclusters are lower. The average number of genes is 15.34, greater than that of SEBI's. The most interesting property of biclusters found by BLN is that it provides biclusters with very low mean squared residue in comparison to the other techniques while maintaining the number of genes between 14 and 17. The row variance shows that the algorithm is able to find interesting patterns, which are illustrated in Figure 5. In Figure 5 are represented six biclusters, with genes ranging from 14 to 17. The values of expression level for the genes in the biclusters are very close to each other, preserving the threshold set by the algorithm. It is specially interesting that the range of values is very small regarding the range of expression levels, as it is shown in the graphs. In addition, some shapes show significant patterns in data, as the bicluster at the top–right corner of Figure 5.

6 Conclusions

Our technique for finding biclusters is based on local nearness, i.e., on the distance between the expression values of genes, under the same condition, within the biclusters. We find groups of genes that have a similar behavior under a subset of conditions. A previous statistical study of the data determine the suitable parameters for our approach, named BLN, which provides a group of different biclusters with highly–related genes and very low mean squared residue. Compared to previous algorithms our approach obtains biclusters with fewer genes than CC and very low mean squared residue. Relative to SEBI, BLN improves the average number of genes maintaining its very low mean squared residue. Graphical examples of biclusters are provided and they show similar behavior for the genes in biclusters.

Future research will focus on improving the efficiency of the algorithm by means of pruning techniques and to validating the biclusters with biological knowledge, such as Gene Ontology.

Acknowledgement

This research was supported by the Spanish Research Agency CICYT under grant TIN2004-001.

Références

Aguilar,J.S., Divina,F. (2005) Evolutionary Biclustering of Microarray Data, *3rd European Workshop on Evolutionary Bioinformatics*.

Ben-Dor,A., Chor,B., Karp,R., Yakhini,Z. (2002) Discovering local structure in gene expression data : The Order Preserving Submatrix Problem, *6th ACM International Conference on Research in Computational Molecular Biology, RECOMB2002*.

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

Dougherty,E., Barrera,J., Marcel,B.,Kim,S., Cesar,R., Chen,Y., Bittner,M., Trent,J. (2002) Inference from Clustering with Application to Gene-Expression Microarrays, *Journal of Computational Biology*, vol. 9(1), 105-126.

Gerstein,M., Chang,J., Basri,R. ,Kluger,Y. (2003) Spectral Biclustering of Microarray Data : Coclustering Genes and Conditions, *Journal Genome Research*, vol. 13(4), 703-716.

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

Lazzeroni,L., Owen, A. (2000) Plaid models for gene expression data, *Technical Report Stanford University*.

Liu,J., Yang, J., Wang,W. (2004) Biclustering in Gene Expression Data by Tendency, *IEEE Computational Systems Bioinformatics Conference 2004*, 183-193.

Madeira,S., Oliveira,A. (2004) Biclustering Algorithms for Biological Data Analysis : A Survey, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 1(1), 24-45.

Shamir,R., Sharan,R., Tanay,A. (2002) Discovering statistically significant biclusters in gene expression data, *Bioinformatics*, vol. 19, Suppl. 1 2002, 136-144.

Wang,H. ,Wang,W., Haixun,W., Yu,P. (2002) Clustering by pattern similarity in large data sets, *ACM SIGMOD International Conference on Management of Data*, 394-405.

Yang,J. ,Wang,W., Haixun,W., Yu,P. (2002) Improving Performance of Bicluster Discovery in a Large Data Set, *6th ACM International Conference on Research in Computational Molecular Biology, RECOMB2002*, Poster.

Yang,J. ,Wang,W., Haixun,W., Yu,P. (2003) Enhanced biclustering on expression data, *3rd IEEE Conference on Bioinformatics and Bioengineering*, 321-327.

Summary

The analysis of gene expression data in DNA chips is an important tool used in genomic research whose main objectives range from the study of the functionality of specific genes and their participation in biological process to the reconstruction of diseases's conditions and their subsequent prognosis. Gene expression data are arranged in matrices where each gene corresponds to one row and every column represents one specific experimental condition. The biclustering techniques have the purpose of finding subsets of genes that show similar activity patterns under a subset of conditions. Our approach consists of a biclustering algorithm based on local nearness. The algorithm searches for biclusters in a greedy fashion, starting with two–genes biclusters and including as much as possible depending on a distance threshold which guarantees the similarity of gene behaviors.