BICLUSTERING GENE EXPRESSION DATA BASED ON A HIGH DIMENSIONAL GEOMETRIC METHOD

XIANG-CHAO GAN[1], ALAN WEE-CHUNG LIEW[2] , HONG YAN[1,3]

[1]Department of Computer Engineering and Information Technology
City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong
[2]Department of Computer Science and Engineering
Chinese University of Hong Kong, Shatin, Hong Kong
[3]School of Electrical and Information Engineering
University of Sydney, NSW 2006, Austra
E-MAIL: 50004098@student.cityu.edu.hk, wcliew@cse.cuhk.edu.hk, h.yan@cityu.edu.hk

Abstract:

In gene expression data, a bicluster is a subset of genes exhibiting a consistent pattern over a subset of the conditions. In this paper, we propose a new method to detect biclusters in gene expression data. Our approach is based on the high dimensional geometric property of biclusters and it avoids dependence on specific patterns, which degrade many available biclustering algorithms. Furthermore, we illustrate that a bilclustering algorithm can be decomposed into two independent steps and this not only helps to build up a hierarchical structure but also provides a coarse-to-fine mechanism and overcome the effect of the inherent noise in gene expression data. The simulated experiments demonstrate that our algorithm is very promising.

Keywords:

Biclustering; gene expression data; superplanes

1. Introduction

In DNA microarray experiments, a key step in the analysis of gene expression data is to discover groups of genes that share similar transcriptional behavior. Clustering gene expression data into homogeneous groups is instrumental in functional annotation, tissue classification, motif identification. A review can be found in [1]. However, standard clustering methods, such as the k-means, hierarchical, or self-organizing map algorithms, have their limitations. They require that the related genes behave similarly across all measurement conditions. When a database includes many heterogeneous conditions from many experiments, clustering algorithms often cannot produce a satisfactory solution.

In this case, biclustering algorithms are preferable. In gene expression data, a bicluster is a subset of genes exhibiting a consistent pattern over a subset of conditions. This means that biclustering performs clustering simultaneously in two dimensions. In some situations, where an interesting cellular process is active only in a subset of conditions, or a single gene may participate in multiple pathways that may, or may not, be co-active under all conditions; biclustering approaches are a key technique to use.

When evaluating a biclustering algorithm, one of the key measurements is the patterns it can detect. There are many different patterns useful for gene expression data. This will be explained in detail in the following section. A good algorithm should incorporate as many as possible patterns and be flexible and extendable. However, most available algorithms are based on specific patterns and this limits their application. The Double Conjugated Clustering (DCC) [2] and Block clustering [3] are designed to detect constant values. The Coupled Two-Way Clustering (CTWC) [4] and Sheng el al.'s algorithm [5] are interesting in their account of the constant rows or columns bicluster. Segal et al. [6] assume the additive model in their algorithm. Lazzeroni and Owen (2000) introduce the notion of a plaid model using general additive model. Wang et al. [7] and Yuval et al. [8] develop their algorithms based on a multiplicative model.

In practice, a perfect bicluster with constant columns or coherent values seldom exists in gene expression data due to noise in experiments. A good biclustering algorithm should be able to adapt to noise situations and find the most feasible solution. To overcome the effect of noise, many biclustering algorithms use parametric method and assume that both noise and gene data values that do not belong to the target bicluster satisfy a certain statistical distribution. This limits the application of their algorithms.

Based on the above analysis, we require a good gene expression data biclustering algorithm be flexible and noise immune. In this paper, we develop a novel biclustering algorithm based on the high dimensional geometric property of the biclusters. We decompose the biclustering algorithm into two independent steps and this facilitates our algorithm to easily incorporate all possible patterns. In addition, we use a coarse-to-fine mechanism to overcome the effect of noise in gene expression data, which has been proven to be very efficient for overcoming the noise effect in the image processing and pattern recognition fields.

2. Geometric Characteristic of A Bicluster

An interesting criterion when evaluating a biclustering algorithm is the identification of the type of biclusters the algorithm is able to find. There are three major classes of biclusters that are well known to be related to gene expression data: (a) Biclusters with constant values; (b) Biclusters with constant values on columns or rows; (c) Biclusters with coherent values on columns or rows. An example of column-oriented biclusters is presented in Figure. 1. Note there are also some bicluster approaches that view the elements of the matrix as symbolic values regardless of the exact numeric values. Since it is not the emphasis of our paper, we omit it for simplicity. A detailed survey of the most available biclustering algorithms based on the type of biclusters detected can be found in [9].

1.2	1.2	1.2	1.2
1.2	1.2	1.2	1.2
1.2	1.2	1.2	1.2
1.2	1.2	1.2	1.2

(a)

1.2	2.0	1.5	3.0
1.2	2.0	1.5	3.0
1.2	2.0	1.5	3.0
1.2	2.0	1.5	3.0

(b)

1.2	2.2	0.2	3.2
2.0	3.0	1.0	4.0
1.4	2.4	0.4	3.4
2.4	3.4	1.4	4.4

(c)

1.0	2.0	0.5	1.5
2.0	4.0	1.0	3.0
1.4	3.0	0.7	2.1
2.4	5.0	1.2	3.6

(d)

Figure 1: Examples of three types of biclusters: (a) Constant Bicluster, (b) Constant Columns, (c) Coherent values by additive models, where each row and column can be obtained by adding a constant to each of the others, and (d) Coherent values by multiplicative models, where each row and column can be obtained by multiplying each of the others by a constant value.

As mentioned before, most available algorithms are based on specific patterns and this limits their applications. The Double Conjugated Clustering (DCC) [2] and Block clustering [3] methods are designed to detect constant values, (Fig.1.a). The Coupled Two-Way Clustering (CTWC) [4] and Sheng el al.'s [5] algorithms extract constant rows or column biclusters, (Fig.1.b). Segal et al. [6] assume the additive model in their algorithm, (Fig.1.c), and Yuval et al. [9] develop their algorithms based on a multiplicative model, (Fig.1.c).

To avoid dependence of biclustering algorithms to a specific pattern, we investigate the common property of a bicluster first in this paper. From a geometric viewpoint, a bicluster in Fig. 1 (b), (c), and (d) is denoted by a single line in a high dimensional space. Each gene in the bicluster is a point lying in this line when we only consider the conditions selected by this bicluster. For example, if we denote the four conditions in Fig.1 as *x*, *y*, *m* and *n*, the bicluster in Fig.1(c) can be denoted as $x = y - 1 = m + 1 = n - 2$ and $x = 0.5y = 2m = \frac{2}{3}n$ for the bicluster in Fig. 1(d).

However, when all conditions, not only the conditions selected by corresponding biclusters but also the conditions do not belong to the bicluster, are considered, this geometric property changes. We cannot denote a bicluster with a line any more. Without loss of generality, assume a three-conditioned experiment with the conditions denoted as *x*, *y*, *z respectively*. If a bicluster covers conditions *x* and *z*, there exists a bundle of hyperplanes that pass through all points in this bicluster. All these hyperplanes conform to the following equation:

$$a_0 + a_1 x + a_3 z = 0 \tag{1}$$

where a_i, (i = 0, 1, 3) are constant and $a_2 y$ is omitted since $a_2 = 0$. A demonstration is given in Fig. 2.

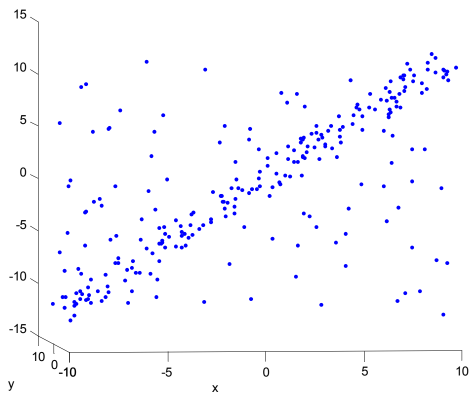

Figure 2: A demonstration of a bicluster's geometric property: the bicluster covering *x,* and *z* conditions lies in a plane in (*x*, *y*, *z*) space.

In Equation (1), there exist coordinates *x*, *z* while y, which does not belong to the bicluster, has disappeared. We find that a hyperplane denotes a possible bicluster: The coordinates appearing in its equation denote the conditions the bicluster covers and the points in the hyperplane denote the genes in the bicluster.

Based on the above analysis, instead of directly seeking specific bicluster patterns, which have been proven to be a NP-complete problem, in this paper we decompose a biclustering procedure into two steps: First, we detect the hyperplanes existing in the gene expression data; then we analyze whether a required pattern exists for the genes which lie on these hyperplanes. The block diagram of our algorithm is as follows.

.

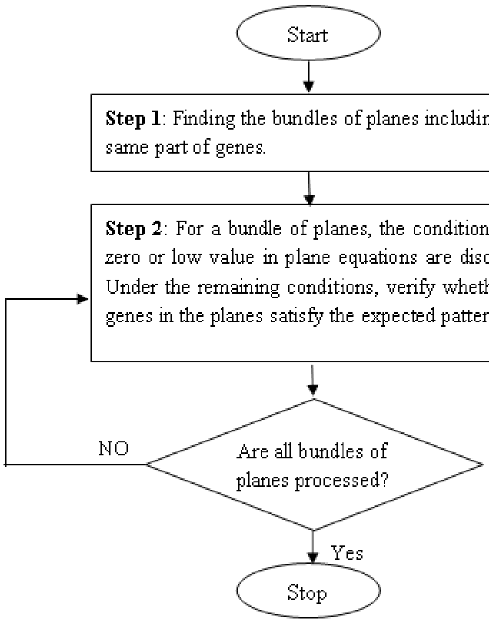
Start
Step 1: Finding the bundles of planes including the same part of genes.
Step 2: For a bundle of planes, the conditions with zero or low value in plane equations are discarded. Under the remaining conditions, verify whether the genes in the planes satisfy the expected pattern.
NO
Are all bundles of planes processed?
Yes
Stop

Figure3: The block diagram of the proposed biclustering algorithm using high dimensional geometric method.

3. Plane Fitting by Fast Hough Transform Method

In the block diagram of our algorithm (see Figure 3), a robust high dimensional plane fitting method is a key step in our algorithm. To achieve this goal, we make use of the Hough Transform (HT). Hough Transform is a powerful technique for line extraction and pattern detection in image processing and computer vision [11]. However, the standard Hough Transform may not be feasible for high dimensional data because of the computational complexity and storage requirement.

Here we use the Fast Hough Transform (FHT) [12] since it has easy high-dimensional extension and gives considerable speed and less storage requirement than the conventional methods. Furthermore, FHT is also a coarse-to-fine method and is noise insensitive. We here review the basic principles of the FHT.

As with a Hough transform, the FHT is also a mapping from an observed *data space*, which is often called a *feature space* in image processing, into a *parameter space*. Each feature point in data space (In gene expression data, a point is data values of a gene) generates "votes" for a set of parameter-space points. An area in the parameter space containing many mapped points reveals the feature of interest.

Hyperplane formulation

The FHT is applied to problems in which points in the parameter space are *hyperplane* represented as

$$a_{0j} + \sum_{i=1}^{k} a_{ij} X_i = 0 \quad \text{for } j = 1,2,\cdots,n. \tag{2}$$

where X_i is the i-th dimension of the parameter space. Each a_{ij} is a function of observed feature points and is normalized such that $\sum_{i=1}^{k} a_{ij}^2 = 1$. For parameter space ($X_1, X_2, \ldots, X_k$), the bound and the desired quantization of X_i are given. Since the value interval of each X_i is given, all k intervals will form a hypercube. We can use a *hypercube* to represent the parameter space.

The FHT algorithm recursively divides the parameter space into hypercubes from low to high resolution. It performs the subdivision and subsequent "vote counting" only on hypercubes with votes exceeding a selected threshold. This hierarchical approach leads to a significant reduction in both computation and storage compared to the conventional Hough Transform.

Hyperplane/hypercube intersection test criterion.

The FHT needs to determine whether a hypercube receives a vote from a particular hyperplane. We can use a simple, conservative test to see whether the hyperplane intersect the hypercube's circumscribing hypersphere, that is, if

$$a_0 + \sum_{i=1}^{k} a_i C_i \le r \tag{3}$$

where [$C_1, \ldots, C_k$] are the coordinates of the hypercube's center and r is the radius of the hypersphere.

K-tree representation

For the FHT, we represent the parameter space as a nested hierarchy hypercube. We can associate a K-tree with the representation. The root node of the tree corresponds to a hypercube with side-length S_0 having one vertex at the origin [0,…,0] and the diagonally opposite vertex at [S_0,…, S_0]. Each node of the tree has 2^k sons arising when that node's hypercube is halved along each of its k dimensions. Each child has a child index, a vector $b = [b_1, \ldots, b_k]$, where each b_i is -1 or 1. The child index is interpreted as follows: if a node at level l of the tree has center C_l then the center of its child node with index [$b_1, \ldots, b_k$] is

$$C_l + \frac{S_{l+1}}{2}[b_1, \ldots, b_k] \tag{4}$$

where S_{l+1} is the side length of the son at level l+1 and $S_{l+1} = S_l / 2$.

We here present an incremental formula for evaluating the test in (3), The normalized distance can be computed incrementally for a child node at level l+1 with child index [b_1, …, b_k] as follows,

$$R_0 = \frac{a_0}{S_0} + \frac{1}{2}\sum_{i=1}^{k} a_i \qquad (5)$$

$$R_{l+1} = 2R_l + \frac{1}{2}\sum_{i=1}^{k} a_i b_i \qquad (6)$$

The test (3) can be expressed as: A hyperplane intersects a hypercube if

$$|R| \le \sqrt{k} \ / \ 2 \qquad (7)$$

To facilitate the understanding of the FHT, we here provide a plane detection example in 3D data. Given M range data points, (x_j, y_j, z_j), j = 1,…,M, the plane detection problem is to find one or more planes that best fit these feature points. If the plane can be represented as

$$x = m_y y + m_x z + b_x \qquad (8)$$

where m_x and m_y are the directional normals of the plane. By choosing (m_y, m_z, b_x), we get a 3-parameter space. A feature point (x_j, y_j, z_j) is transformed into the parameter spaces

$$a_{0j} + a_{1j} m_y + a_{2j} m_z + a_{3j} b_x = 0$$

where

$a_{0j} = -x_j / L_j$, $a_{1j} = y_j / L_j$, $a_{2j} = z_j / L_j$, $a_{3j} = 1/L_j$,and $L_j = \sqrt{y_j + z_j + 1}$.

Our biclustering algorithm

To summarize, when given a set of genes expression data {F_j} under diverse experimental conditions, the high dimensional geometric biclustering method can be summarized as:

Parameters or function needs to be predetermined:

1. A minimum vote count "T" as threshold and the desired resolution "q".
2. A transformation that maps each gene expression data F_j into a hyperplane in parameter space represented by $a_{0j} + \sum_{i=1}^{k} a_{ij} X_i = 0$ for $j = 1,2,\cdots,n$. where X_i is bounded by [0, S_0] . The root of K-tree at [S_0/2, …, S_0/2]

Procedure:

1. Transform genes expression data into parameter space.
2. Compute the initial normalized distances from the hyperplane to the root node and perform the vote procedure for the root node. For each set of gene expression data, if Equation (7) is satisfied, add one to the vote number of the root node. If the vote number for root node is bigger than the threshold T and resolution is less than q, subdivide the root node into the K-tree child nodes.
3. Vote for each child node and subdivide them if possible. A similar vote-and-subdivide mechanism is performed for each new node until no new nodes appear.
4. When there is no node with resolution equal to q and the vote number is bigger than T, record the node with the biggest resolution as it is the most probable solution. When there are several nodes with resolution equal to q and a vote number bigger than T, incorporate the planes covering the same parts of the genes.
5. For each bundle of planes, the conditions with zero or low value in the plane equations are discarded. Under the remaining conditions, verify whether the genes in the planes satisfy the expected pattern. If yes, we get a bicluster; otherwise just discard it. Same processing continues until all bundles are processed.

The above procedure is very efficient for a modest database. Since the computational complexity of the FHT algorithms change more dramatically with the dimension increase than with the change of gene numbers, we here provide a simple divide-and-conquer mechanism for large datasets. First, divide the conditions into several non-overlapping blocks and each block includes all genes but different conditions. Then, we perform the proposed biclustering algorithm for each block. For a detected bicluster in one block, we test whether the other conditions can be incorporated into it. Lastly, we delete the same bicluster. Using this simple extension, we actually transform the biclustering problem which performs clustering on the two dimensions simultaneously into a low-dimensional biclustering problem and a simple clustering problem in one dimension.

4. Experiment results

We analyze the performance of our algorithm on several datasets. We verify that the proposed algorithm is not limited to a specific pattern by using our algorithm to detect three different patterns: constant columns, constant rows and coherent values by multiplicative models. All three patterns are very useful and frequently mentioned in biclustering algorithms for gene expression data. For a clear evaluation, we also generate a synthetic dataset with three overlapping biclusters to examine the ability of our algorithm to find multiple biclusters, especially when overlaps between biclusters are present. For these synthetic datasets, since the genes and conditions covered by each bicluster are known, they will give a detailed look at our algorithm.

Synthetic data with one embedded bicluster

Biclusters with constant columns. We embed a pattern of 25 rows by columns 8 into a dataset of size 100 by 30. In this experiment, the pattern is constant columns and the value of each row is produced from a uniform distribution *U*(-5, 5). The background is also generated from the uniform distribution *U*(-5, 5). A similar experiment was performed by Sheng *et al*. (2003).

The final pattern of the bicluster revealed by our algorithm is shown in Fig. 4. We see that all the columns where the embedded pattern locations were correctly found. In addition, all embedded rows were recovered. We can compare our performance to that of (Sheng *et al*. 2003). For their algorithm, all columns where the embedded pattern locations were correctly found and most of the embedded rows were recovered.

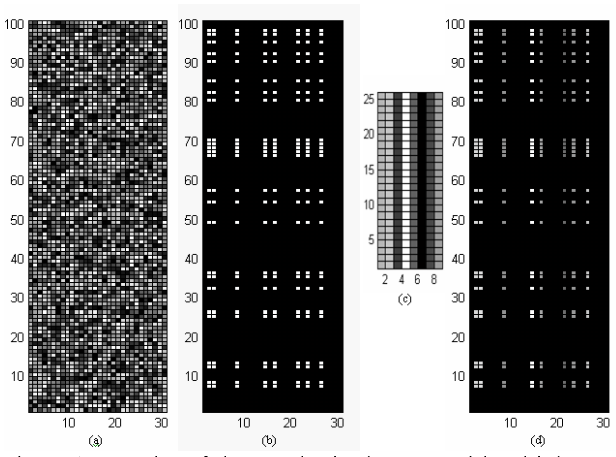

Figure4: Results of the synthetic data set with a bicluster of constant columns. (a) The data matrix. (b) The position of the data matrix belongs to the bicluster. (c) The pattern of the bicluster. (d) Pattern and the position of the bicluster revealed by the proposed high geometric method.

Biclusters with constant row and biclusters with coherent values by multiplicative models. Our algorithm is very flexible at detecting different patterns, besides the pattern of constant columns. We now also test the performance of our algorithm using following two patterns: constant rows and coherent values by multiplicative models. The experimental results are provided in Figure 5 and Figure 6, respectively. We found that for all these three datasets, the resulting biclusters are correctly found with the same conditions and same genes as the original known bicluster.

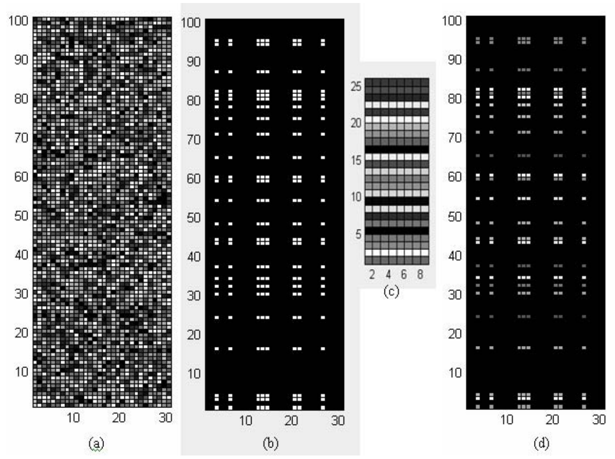

Figure 5: Results of the synthetic data set with a bicluster of constant rows. (a) The data matrix. (b) The position of the data matrix belongs to the bicluster. (c) The pattern of the bicluster. (d) pattern and the position of the bicluster revealed by the proposed high geometric method.

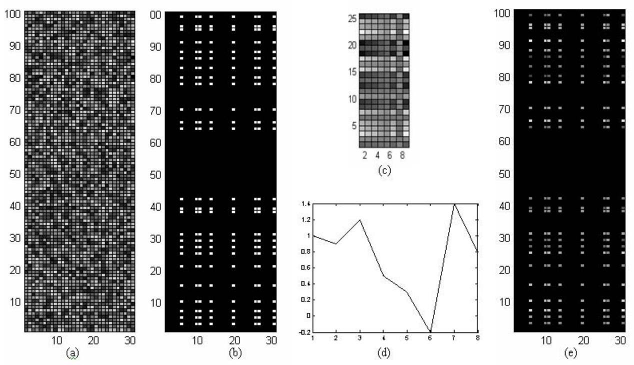

Figure 6: Results of the synthetic data set with a bicluster with coherent values by multiplicative models. (a) The data matrix. (b) The position of the data matrix belongs to the bicluster. (c) The pattern of the bicluster. (d) The multiplicative coefficients of each row in the bicluster. (e) pattern and position of the bicluster revealed by the proposed high geometric method.

Synthetic data with multiple overlapping biclusters

To examine the ability of our algorithm to find multiple biclusters, especially when overlap between biclusters is present, we embed three biclusters into a noisy background described by a uniform distribution *U*(-5, 5). The dataset is of size 200 rows by 40 columns, and the three embedded biclusters are of the following sizes, 40 by 7 for Bicluster 1, 25 by 10 for Bicluster 2, and 35 by 8 for Bicluster 3. As can be seen in the main plot of Figure 7(a), Bicluster 1 overlaps with Bicluster 2 at two columns, and Bicluster 3 overlaps with Bicluster2 at five rows and three columns.

In this experiment, we divide the dataset into 4 blocks, each with 10 conditions and 200 rows. In the first block, the bicluster 1 was found with all 7 conditions and 40 rows. Bicluster 2 was also found with the first 3 conditions and 25 rows, and the other 7 conditions was tested and then added into the bicluster. In the 2nd block, bicluster 3 was found with 8 conditions and 35 rows. Bicluster 2 with 7 conditions missed in Block 1 is detected and then the other 3 conditions are found back in the subsequent analysis. Since Bicluster 2 is detected twice, one is deleted.

Based on the above analysis, three biclusters are perfectly detected. An unexpected outcome of our algorithm is in Block 2, a bicluster with 3 conditions overlapping with Bicluster 2 and Bicluster 3 and 60 rows comprises all rows of Bicluster 2 and 3 is detected as a new bicluster. After a careful analysis, we think it is a reasonable result. Furthermore, a good algorithm should detect many possible patterns and let the users decide which is preferable.

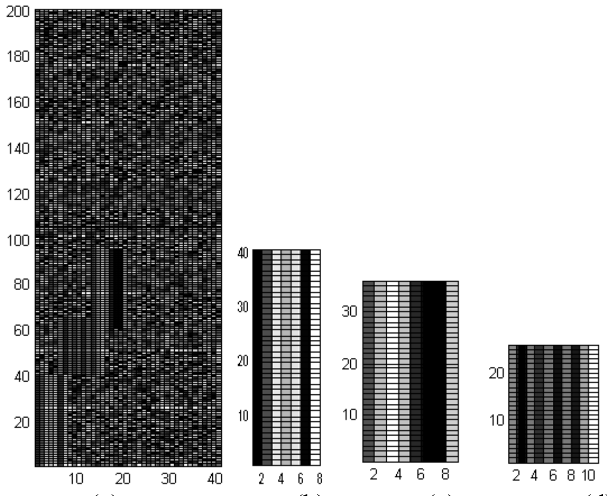

Figure 7: Results form the synthetic data set with multiple overlapping biclusters. (a) The data matrix, (b), (c) and (d) The three biclusters found by the proposed method.

5. Conclusions

We have developed a new high dimensional geometric method for gene expression data biclustering. Our algorithm is significantly different from the available algorithms. We illustrate that a biclustering algorithm can be decomposed into two independent steps and this not only helps it to build up a hierarchical structure but also provides coarse-to-fine mechanism to overcome the effect of the noise in gene expression data. The simulated experiments demonstrated that our algorithm is very promising.

Acknowledgements

The work described in this paper was fully supported by an interdisciplinary grant (Project 9010003) and a strategic research grant (Project 7001706) from City University of Hong Kong.

References

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

[2] Busygin,S., Jacobsen,G. and Kramer,E. (2002) Class discovery in gene expression data. *Proc. 5th Annual Intl. Conf. on Computational Biology*, 5, 31-38

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

[4] Getz,G., Levine,E., and Domany,E. (2000) Coupled two-way clustering analysis of gene microarray data. *Proceedings of the Natural Academy of Sciences USA*, 97, 12079-12084.

[5] Sheng,Q., Moreau,Y., and Moor,B.D.(2003) Biclustering microarray data by Gibbs sampling. *Bioinformatics*, 19(Suppl. 2), ii196-ii205.

[6] Segal,.E., Taslar,B., Gasch,A., Friedman,N., and Koller,D. (2001) Rich probabilistic models for gene expression. *Bioinformatics*, 17, S243-S252.

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

[8] Wang,H., Wang,W., Yang,J., and Yu, P.S. (2002) Clustering by pattern similarity in large data sets. *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, 394-405.

[9] Yuval,K., Basri,R., Chang, J.T., and Gerstein., Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Research*, 13, 703-716, 2003.

[10] Madeira,S.C., and Oliveira,A.C. (2004) Biclustering algorithms for biological data analysis: a survery. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*.

[11] Ballard,D.H., and Brown,C.M., (1982) *Computer Vision*, Prentice Hall, Englewood Cliffs, NJ.

[12] Li,H., Lavin,M.A., and Master,R.J.L. (1986) Fast Hough Transform: a hierarchical approach. *Computer Vision, Graphics, and Image Processing*, 36,139-161.