A New Clustering Algorithm Using Message Passing and its Applications in Analyzing Microarray Data

Huimin Geng[1], Xutao Deng[2] and Hesham Ali[2]
[1]Department of Pathology and Microbiology, University of Nebraska Medical Center, Omaha, NE 68198, [2]Colleage of Information Science and Technology, University of Nebraska at Omaha, Omaha, NE 68182
Email: huimingeng@unmc.edu, {xdeng, hali}@mail.unomaha.edu

Abstract

In this paper, we proposed a new clustering algorithm that employs the concept of message passing to describe parallel and spontaneous biological processes. Inspired by real-life situations in which people in large gatherings form groups by exchanging messages, Message Passing Clustering (MPC) allows data objects to communicate with each other and produces clusters in parallel, thereby making the clustering process intrinsic and improving the clustering performance. We have proved that MPC shares similarity with hierarchical clustering but offers significantly improved performance because it takes into account both local and global structure. MPC can be easily implemented in a parallel computing platform for the purpose of speed-up. To validate the MPC method, we applied MPC to microarray data from the Stanford yeast cell-cycle database. The results show that MPC gave better clustering solutions in terms of homogeneity and separation values than other clustering methods.

1. Introduction

Clustering algorithms are widely used in bioinformatics to classify data, as in the analysis of gene expression and the building of phylogenetic trees [1,2]. In the literature, a vast amount of clustering algorithms exists, including optimization methods such as k-means [3], agglomerative algorithms such as Hierarchical Clustering (HC) [1] and Super Paramagnetic Clustering (SPC) [4,5], graph theoretical based algorithms such as CLICK [6] and CAST [7], and neural network approaches such as Self Organizing Maps (SOM) [8]. Refer to [9] for the description and the characteristics of each of the algorithms and [10] for a comparative review of those methods in expression profile.

Among the best-known and most widely accepted clustering algorithms, are those involving HC. HC algorithms (the agglomerative manner) proceed from an initial partition into singleton clusters by successive merging of clusters until all elements belong to the same cluster. Different types of linkage (i.e. single, complete, average, and centroid) can be chosen by the user to compute the distance of two clusters [11]. HC arranges the data into a tree structure which can be easily viewed and understood; and the hierarchical structure provides potentially useful information about the relationships between clusters. However, HC has some inherent problems: HC always puts a priority on global distance but not honors local structures; that is, in each step, only the two clusters which have the globally lowest distance (or highest similarity) are merged. This mechanism of HC may produce clusters against intuition and with more singletons (see section 2.3), and may not be suitable for biological data sets as biological processes are often *parallel* by nature. As an example, behavior of multiple genes in a regulatory network, displayed by microarray technology, is a parallel process. At a specific time, each gene interacts only with its local environment. We believe that HC may not be effective in this case, because the parallel process and local interaction will not be represented by HC, which is designed to sequentially merge objects by global measurements.

To overcome the problems, we propose a new clustering algorithm, employing the concept of message passing. Message Passing Clustering (MPC) allows data objects to communicate with each other, generates clusters in parallel so that the global distances and local distances are well-balanced, and hence improves the performance of clustering. Also, MPC leads itself very well to the parallel implementation. It does not have a sequential bottleneck. In addition, the proposed message passing technique can be extended in a number of ways and opens the door for further development to address more challenging clustering problems.

This article is organized as follows. The sequential algorithms of MPC, the software package, the properties of MPC and the proofs are given in section 2. The parallel implementation of MPC is illustrated in Section 3. Section 4 presents artificial and real data studies. Section 5 summarizes the MPC method and provides a discussion.

2. Methods

The computing model of the MPC method is inspired by a common real-life situation. Suppose we have a social event where people don't know one another at the beginning of the event. One person may look around and talk to another person to see if they share some common interest. If so, they will continue the conversation and other people with the same interest may join this group as time passes. It is often the case that after a while a set of talking groups are formed at this event. This communication model shows parallel and spontaneous clustering processes by exchanging information between people. In the MPC algorithm, we employ the concept of message passing to represent the information exchange processes between data objects. An abstract of a preliminary version was presented in Geng *et al.* [12].

2.1 Basic Algorithm

The key idea of MPC is to allow vectors to communicate with each other so that the clusters can be formed in parallel. Initially each vector is a cluster by itself. During the clustering process, each cluster will send a message to its nearest neighbor (the cluster which has the maximum similarity to the sending cluster) by calling function *Msg_Send.* Each cluster C_i is associated with two special memory cells, $C_i.TO$ and $C_i.FROM$. These two cells function as a message box with $C_i.TO$ storing the outgoing address and $C_j.FROM$ storing the incoming address. After sending messages, each cluster checks the message box by calling *Msg_Rcv*. If the outgoing and incoming addresses are the same, a mutually nearest neighbor (C_i is the nearest neighbor of C_j and vice versa) is found such that the two clusters merge to one. This process is repeated until the number of clusters K is reached, where K can be any specified number from 1 to n and K is 1 by default.

Algorithm 1. MPC main function

```
Inputs: m-dimensional vectors $X_1, X_2, ..., X_n$
Outputs: A partition of $X_1, X_2, ..., X_n$ into K clusters
MPC ($X_1$, X2,...,Xn)
    Initialize ($X_1$, X2... Xn);
    while (Num_Clusters>K)
        for (each cluster $C_i$) //message sending
            $C_j$=Find_Nearest_Neighbor ($C_i$);
            Msg_Send ($C_i$,$C_j$);
        for (each cluster Ci) //message receiving
            $C_j$=Msg_Rcv ($C_i$) // check message
            if ($C_j$ not equal to NULL)
                New Cluster C'= Merge ($C_i$, $C_j$);
                Num_Clusters= Num_Clusters-1;
Msg_Send ($C_i$, $C_j$) //send a message from $C_i$ to $C_j$
    $C_i$.TO=j;
    $C_j$.FROM=i;
Msg_Rcv ($C_i$) //check the message box
    if ($C_i$.FROM=$C_i$.TO=j) //find mutually
        return $C_j$;
    else
        return NULL;
```

2.2 Software Package of MPC

The software of MPC is written in C++ and can be downloaded in Windows™ and UNIX platforms at http://bioinformatics.ist.unomaha.edu/~hgeng/. The program input is a distance matrix. Users can select: clustering algorithms— MPC or HC; similarity metrics— correlation coefficient or Euclidean distance; similarity linkage— single, complete, average, or centroids; and cluster display— with or without branch length. Two files are outputted: "tree.txt" is used as an input to tree view software, NJplot [13], to display the clustering dendrogram; "cluster.txt" contains the information of clustering processes, including the number of clusters, the size of each cluster, the data objects in each cluster, and the average homogeneity and separation values for evaluating the solution.

2.3. Properties of MPC

While it may appear that MPC and HC produce similar outputs, we show that, more often than not, the output clusters of the two approaches differ in favor of the MPC method.

Definition*:* A *couple* is a pair of clusters a and b such that they are mutually nearest:

$$b = \arg\min_{i\in P,\ i\neq a} D(a,i) \quad \text{and} \quad a = \arg\min_{i\in P,\ i\neq b} D(b,i),$$

where i is any cluster in the current cluster pool P and $D(a, b)$ is the distance between the cluster a and b.

Theorem 1**:** The number of new clusters produced at each round of message exchanging is from 1 to *n*/2, where *n* is the number of clusters at the previous step.

Proof of theorem 1: The number of potential couples in the cluster pool is from 1 to *n*/2, so the proof follows immediately. Theorem 1 shows clusters merge in an ideally exponential fashion so that MPC is a conceptually fast algorithm which reflects its parallel process. ■

Lemma: Let (*a, b*) be a couple in the current cluster pool *P*, and the similarity criteria be single, complete, or average linkage, then *D(a,b)* is bounded by the following inequalities: (1) $D(a,b) \le D(a,(b \cup c))$, (2) $D(a,b) \le D((a \cup b),c)$, and (3) $D(a,b) \le D(a,(c \cup d))$. where *c* and *d* can be any clusters except *a* and *b* in *P*.

Proof of the lemma: Since *couple* is a symmetric definition, we can exchange *a* and *b* in the preceding lemma. Without any loss of generality, we assume that the similarity criterion is complete linkage. For single and average linkage, the lemma can be proved in similar fashion. We can prove this lemma as follows:

(1) $D(a,(b \cup c)) = \max(D(a,b), D(a,c)) \ge D(a,b)$ by the definition of complete linkage. Similarly,

(2) $D((a \cup b),c) = \max(D(a,c), D(b,c)) \ge D(a,c) \ge D(a,b)$

(3) $D(a,(c \cup d)) = \max(D(a,c), D(a,d)) \ge D(a,c) \ge D(a,b)$

It is important to note that the lemma does not apply to the case of centroid linkage because the centroid of a cluster is able to "move" during the clustering process by merging with other clusters and hence the distance between two clusters is not "fixed". ■

Theorem 2**:** Comparing MPC and HC in various similarity criteria, the following propositions hold:
i). In the case of centroid linkage, the clustering dendrogram from MPC and HC are generally different.
ii). In the case of single, complete, or average linkage, **a).** the final clustering dendrogram from MPC and HC are equivalent; **b)**. But intermediate clusters from MPC and HC are generally different. If the number of final clusters is specified in advance, MPC and HC may yield different solutions.

Proof of i): Due to the different merging strategies between MPC and HC, the centroids in the two methods are generally different, hence the clustering dendrogram are generally different. We want to show one example. Suppose there are four two-dimensional data objects *a, b, c* and *d* with the coordinates given in Figure 1. MPC merges (*a b*) and (*c d*) at the same step, while HC merges only (*a b*) at the first step, and then updates the distance table. Since the distance between *c* and the centroid of (*ab*) is less than the distance between *c* and *d, c* is grouped with (*ab*) at the second step (Figure 2). The solution given by MPC conforms to our intuition, while the solution from HC does not. ■

Proof of ii)-a): A dendrogram is a tree representing the merging processes controlled by the clustering algorithm. A branch in a dendrogram represents a "marriage" of two data objects in a couple. Now we show by induction that MPC and HC produce the same dendrogram in the case of single, complete and average linkage.

Basis: In the leaves level, each data object is a cluster. MPC and HC have the same dendrogram.

Induction: Assume that at some step, MPC and HC has exactly the same dendrogram. We want to show that a branch produced by merging a couple in MPC should have a correspondence in HC. Suppose we identify a couple (*a, b*) which is about to merge in MPC. From (3) in the lemma, $D(a,b) \le D(a,(c \cup d))$, we see that *(a, b)* still remains a couple even if other couple *(c, d)* may be formed before the marriage of *(a, b)* in HC. So the same branch representing the merging of *(a, b)* will be preserved in HC.

Proof of ii)-b): Figure 3 shows an example of different solutions from MPC and HC when the number of clusters is three. In HC, there are two singletons, *d* and *e*. While in MPC, though the distance between *d* and *e* are large, they are, relatively, closest to each other as compared to other nodes, so *d* and *e* are grouped if considering not only global distance but also local distance.■

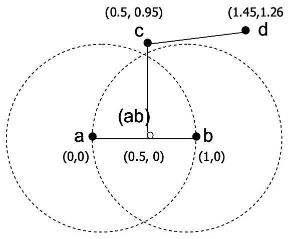

Figure 1. Coordinates.

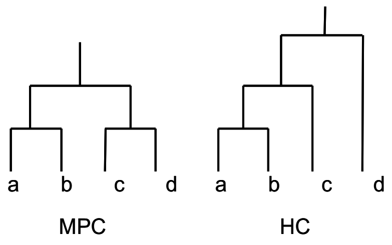

Figure 2. Different dendrogram.

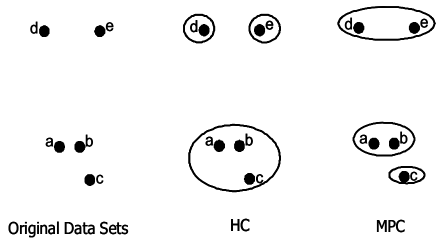

Figure 3. Clustering results of MPC, HC.

3. Parallel Implementation of MPC

Clustering algorithms are often used to analyze large biological data. Hence efficient implementations of clustering algorithms are highly desirable. Additionally, with the recent advances in computer hardwires, massive parallel machines (such as cluster computing facilities) are currently available and relatively inexpensive. The basic MPC algorithm can be implemented as a sequential algorithm in a single processor machine, as shown in 2.1. However, due to the parallel nature of MPC, we can easily implement it on a multiprocessor platform to enhance the speed of processing. As an example, we show a parallel implementation of the single-linkage on Concurrent Read Concurrent Write (CRCW) Parallel Random Access Machines (PRAMs) to emulate the message passing scenario. PRAMs allow multiprocessors to access a single shared parallel memory simultaneously. CRCW allows processors to concurrently write to or read from any location on the shared memory. The parallel computing model is illustrated in Figure 4.

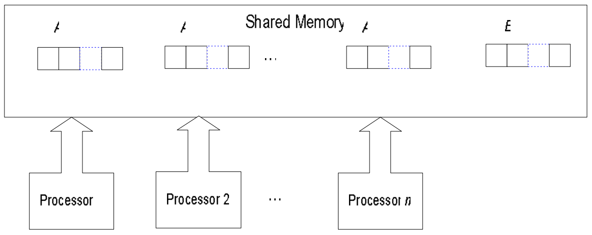

Figure 4. Parallel computing model of MPC.

In this model, each cluster will be the responsibility of one processor, so *n* processors are needed to perform the clustering of *n* data objects. When two clusters are agglomerated, the lower numbered processor corresponding to the two clusters takes over full responsibility for the new cluster. If one processor no longer has any clusters in its responsibility it becomes idle. In addition to the mail box memory, each processor *i* maintains an array A_i of the distances between the cluster C_i and the other clusters. We use a special array *B* to store the nearest neighbor for each cluster to speed up the searching. The parallel algorithm is described in Algorithm 2.

To update the data structure, each processor *k* updates the inter-cluster distance array A_k to reflect the new distance between its clusters and the newly agglomerated cluster. For each cluster it is responsible for, each processor must thus update a single location in the array. No operation need be performed for the newly agglomerated clusters, since the new distances will be determined by the remaining clusters. Step 1 is performed once and step 2, 3, 4 and 5 are performed *n* times, so the parallel algorithm has a time complexity *O(n).* The implementation of MPC using other similarity criteria can be handled in the similar fashion with complexity *O(n).*

Algorithm 2: Parallel Algorithm of MPC

```
1. Generate distance array $A_i$ on each processor $i$ and store the nearest neighbors in the special array $B$.  $O(n)$
2. Obtain the nearest neighbor on each processor $i$.  $O(1)$
3. Identify couples by message sending and receiving on each processor $i$. $O(1)$
4. For each couple $(i, j)$, do the following:
    4.1 Update distance array $A_k$ on each processor $k$, where $k$ is any processor other than $i$ and $j$. $O(1)$
    4.2 Update $B$ on each processor.  $O(1)$
    4.3 Merge couple $(i,j)$ by dropping processor $j$ $(i<j)$. $O(1)$
5. If two or more processors remain, go to step 2.
```

4. Results

We test the validity of the MPC method with both artificial and real microarray gene expression data. Centroid criterion was used in both studies. We used Euclidean distance in 4.1 and the correlation coefficient in 4.2.

4.1. Illustrative Data Sets

An online simulator, eXPatGen [14], was used to generate artificial gene expression data, for which the classes are known. Figure 5 demonstrates the methodology of evaluating MPC with the artificial data. Employing the user-defined inputs (such as gene groups) to the simulator, dynamic mRNA profiles similar to those produced from microarray experiments were generated. Then the proposed method was applied in the analysis of the data and the results were directly compared to the initial input.

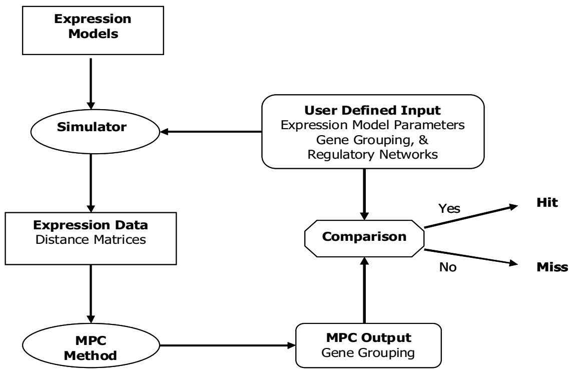

Figure 5. Methodology of evaluating MPC with the simulated data.

We show one example which contains 10 groups of genes, with 10 genes in each group. By setting parameters, we can control which genes are induced or repressed by which genes, define the transcription dynamics for each gene, and hence determine the gene expression patterns (Figure 6). Applying the proposed method to this data set, we found that the clustered display (Figure 7) includes exactly nine clusters, with each cluster presenting one gene group. Note, A and B belong to one cluster since they have the same dynamic expression profile. In total, we tested 35 data sets. For each data set, 10 to 100 genes with dimension ranging from 20 to 40 were included. A 95% hit rate was achieved, in which 639 of 674 genes were correctly clustered. This study shows that MPC has high accuracy and stability.

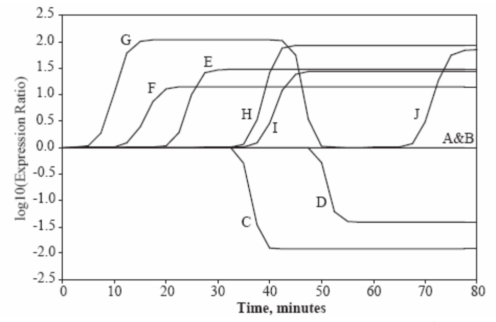

Figure 6. Dynamic expression profiles [14].

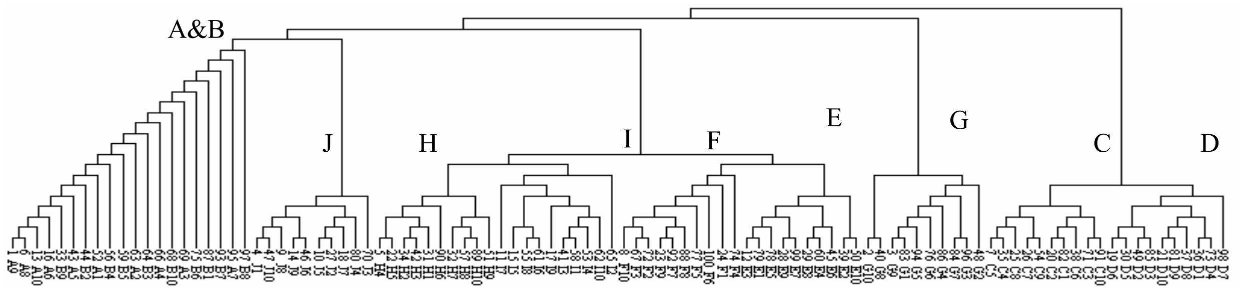

Figure 7. Clustered display of the simulated data set using the MPC method.

4.2. Gene Expression Data

Spellman *et al.* [15] identified 800 genes that are cell-cycle regulated from the Stanford yeast cell-cycle database (http://cellcycle-www.stanford.edu), and Shamir *et al.* [9] selected 698 of those 800 genes, which have no missing entries, over 72 conditions, to evaluate different clustering algorithms: *K*-means, CAST, SOM, CLICK and "Heuristic". In order to compare MPC with those methods, we analyzed this 698×72 data set and set up the experiment with the same parameters as those chosen in [9]. Based on the analysis conducted by Spellman *et al.*, we expect to find in the data five main clusters (G1-peaking, S-peaking, G2-peaking, M-peaking, and M/G1-peaking genes), and compare different solutions in terms of the homogeneity and separation values since the true clusters are unknown.

Homogeneity is an intra-cluster measure; a good homogeneity indicates objects in the same cluster are highly similar to each other. Separation is an inter-cluster measure; a good separation means objects from different clusters have low similarity to each other. The average homogeneity and average separation are defined as:

$$H_{Ave} = \frac{1}{|N|} \sum_{X \in N} S(X, Cl(X))$$

$$S_{Ave} = \frac{1}{\sum_{i \neq j} |C_i||C_j|} \sum_{i \neq j} |C_i||C_j| S(C_i, C_j)$$

where *X* is a data object, *Cl(X)* is a cluster which *X* belongs to, *N* is the set of all data objects in the cluster, and C_i and C_j denote disjoint clusters. Recall that *S* is the similarity between two data objects.

Table 2 shows the solutions produced by each program and their homogeneity and separation parameters. GeneCluster is clustering software which uses the SOM algorithm. The so-called "Heuristic" solution is not the true solution, but it is obtained manually by inspecting the expression patterns and comparing to the literature [15]. We found that MPC identified five main clusters, as it should be, and gave relatively low homogeneity and high separation values.

Table 2. A Comparison of Clustering Solutions

Program	# Clusters	Homogeneity	Separation
K-Means	49	0.629	0.086
CAST	5	0.6	-0.146
GeneCluster	6	0.617	-0.073
CLICK	6	0.656	-0.098
"Heuristic"	5	0.572	-0.133
MPC	5	0.593	-0.152

Figure 8 gives a comparison of homogeneity and separation values for all solutions. We say a clustering solution has high quality if H_{Ave} is relatively high and/or S_{Ave} is relatively low. So the ideal point is the lower right corner where both are good. However, homogeneity and separation are two conflicting parameters; Improvement of one will deteriorate the other. In Figure 8, we find MPC gives a competitive solution which is closest to the "Heuristic" solution.

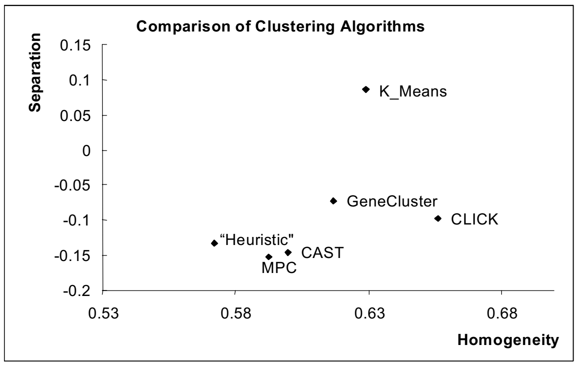

Figure 8. A comparison of homogeneity and separation values for all solutions.

5. Conclusions and Discussion

In this paper we addressed the intrinsic problems of HC by proposing a new clustering algorithm which employs the concept of message passing. By taking advantage of the communication among data objects and by taking into account both local and global structure, MPC can describe parallel and spontaneous biological processes more precisely. In addition to be performed in a single processor machine, MPC can be easily implemented in powerful and efficient parallel computing platforms. In the testing results, a 95% hit rate was achieved for the simulated gene expression data, and a partition closest to the "Heuristic" solution was obtained for the real yeast cell-cycle gene expression data.

One advantage of MPC is its flexible structure. We could change the message type and the way the message is handled to deal with more challenging situations. For example, MPC can easily deal with the data set consisting of a mixture of data types. Another advantage of MPC is that it can be developed further to incorporate additional advanced features, such as undoing clusters. We could send messages from each cluster to multiple clusters and assign merging probabilities to those messages. In the receiver side, we recalculate the probability of each node which was already in the cluster and kick out those no longer having good probability, so that the irreversible problem in HC is solved.

Acknowledgements

This work was supported by the NIH grant number P20 RR16469 from the INBRE program of the National Center for Research Resources.

References

[1] Eisen M.B., Spellman P.T., Brown P.O., and Botstein D. "Cluster analysis and display of genome-wide expression patterns", *PNAS, U S A*, **95**, 14863-14868, 1998.

[2] Nei M. and Kumar S. *Molecular Evolution and Phylogenetics.* Oxford University Press, New York, 2000.

[3] Herwig R., Poustka A. J., Meuller C., Lehrach H., and O'Brien J. "Large-scale clustering of cDNA-fingerprinting data", *Genome Research*, **9**(11), 1093-1105, 1999.

[4] Getz, G., Levine, E. and Domany, E. "Coupled two-way clustering analysis of gene microarray data", *PNAS*, **97**, 12079-12084, 2000.

[5] Blatt M., Wiseman S., and Domany E. "Super-paramagnetic clustering of data", *Physical Review Letters* **76**, 3251-3254, 1996.

[6] Sharan R. and Shamir R. "CLICK: A clustering algorithm with applications to gene expression analysis", In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB),* 307-316, 2000.

[7] Ben-Dor, Shamir R., and Yakhini Z. "Clustering Gene Expression Patterns", *J. Comput. Biol.*, **6**, 281-297, 1999.

[8] Tamayo P., Slonim D., Mesirov J., Zhu Q., Kitareewan S., Dmitrovsky E., Lander E. S., and Golub T.R. "Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation", *PNAS*, **96**, 2907-2912, 1999.

[9] Shamir R., and Sharan R. "Algorithmic Approaches to Clustering Gene Expression Data", *Current Topics in Computational Molecular Biology*, 269-300, MIT Press, 2000.

[10] Tseng. G.C. "A Comparative Review of Gene Clustering in Expression Profile". *Proceedings of the 8th International Conference on Control, Automation, Robotics and Vision (ICARCV),* 1320-1324, 2004.

[11] Gibbons F.D. and Roth F.P. "Judging the quality of gene expression-based clustering methods using gene annotation". *Genome Res.*, **12**, 1574–1581, 2002.

[12] Geng H., Bastola D. and Ali H.H. "A New Approach to Clustering Biological Data Using Message Passing", In *Proceedings of 2004 IEEE Computer Society Bioinformatics Conference (CSB)*, 493-494, best poster awards, 2004.

[13] Perrière G. and Gouy M. "WWW-Query: An on-line retrieval system for biological sequence banks", *Biochimie*, **78**, 364-369, 1996.

[14] Michaud D.J., Marsh A.G., and Dhurjati P.S. "eXPatGen: generating dynamic expression patterns for the systematic evaluation of analytical methods", *Bioinformatics,* **19**, 1140-1146, 2003.

[15] Spellman P.T., Sherlock G., Zhang M.Q., Iyer V.R., Anders K., Eisen M.B., Brown P.O., Botstein D., and Futcher B. "Comprehensive Identification of Cell Cycle-regulated Genes of the Yeast *Saccharomyces cerevisiae* by Microarray Hybridization", *Molecular Biology of the Cell*, **9**, 3273-3297, 1998.