Evolutionary Biclustering based on Expression Patterns

Beatriz Pontes
Department of Computer Science
University of Seville
Seville, Spain
Email: bepontes@us.es

Raúl Giráldez, Jesús S. Aguilar-Ruiz
School of Engineering
Pablo de Olavide University
Seville, Spain
Email: {giraldez, aguilar}@upo.es

***Abstract*—The majority of the biclustering approaches for microarray data analysis use the Mean Squared Residue (MSR) as the main evaluation measure for guiding the heuristic. MSR has been proven to be inefficient to recognize several kind of interesting patterns for biclusters. Transposed Virtual Error (VE^t) has recently been discovered to overcome MSR drawbacks, being able to recognize shifting and/or scaling patterns. In this work we propose a parallel evolutionary biclustering algorithm which uses VE^t as the main part of the fitness function, which has been designed using the volume and overlapping as other objectives to optimize. The resulting algorithm has been tested on both synthetic and benchmark real data producing satisfactory results. These results has been compared to those of the most popular biclustering algorithm developed by Cheng and Church and based in the use of MSR.**

***Keywords*-genetic algorithm; parallel computing; gene expression data; microarray analysis; biclustering;**

I. INTRODUCTION

Microarray technologies allow to simultaneously measure the expression level of thousand of genes, producing large numerical data matrices in which rows represent genes under study and columns refer to the experimental conditions the genes have been put through [1]. One of the main characteristics of microarrays of gene expression data is that they are unbalanced. This means that the number of genes under study is much greater than the number of experimental conditions. Gene expression matrices from microarrays would contain thousands of rows but only a few columns (usually less than a hundred). In order to discover relevant information from these matrices, several data mining techniques have already been applied. Clustering techniques aim at finding groups of genes that present a similar variation of expression level under all the experimental conditions [2]. However, relevant genes are not necessarily related to every condition. Biclustering techniques [3] outperform traditional clustering in two main aspects: firstly, they simultaneously group genes and conditions, thus a bicluster will group not only genes but also the conditions under which the genes are related. Secondly, while overlapping is rarely admitted among different clusters, intersection among biclusters is permitted, where several genes and conditions may be grouped together in more than one bicluster.

The biclustering problem has been proven to be NP-hard [4]. It consist of selecting a number of sub-matrices from a microarray, using both dimensions simultaneously. Due to the complexity of this problem, an exhaustive search is impracticable and therefore it is essential the application of any heuristic methodology, in combination with an evaluation measure for biclusters. Cheng and Church [5] were the first in applying biclustering to microarray data, using a greedy search algorithm combined with random replacement in order to render it non-deterministic. They also developed the well-known evaluation measure *Mean Squared Residue* (MSR). Both heuristic and evaluation measure have been proven to be inefficient for the biclustering problem [6], [7]. Since 2000, some others heuristics have been proposed [8], [9], [10], some of them using evolutionary approaches [11], [12], [13], although the majority of them use MSR as the main guidance for the algorithm.

When looking for a coherent behaviour of a group of genes, the different possibilities can be grouped in two main patterns: shifting and scaling behaviours [6]. They can also be summarized into a third one, the combined shifting and scaling pattern, and which is the most interesting one since it is the most probable in real gene expression data [14]. In 2010, Pontes et al. [15] proposed a new evaluation measure for biclusters, named *Transposed Virtual Error* (VE^t) and based on these expression patterns concepts. VE^t has been proven to be efficient in order to identify all these kind of patterns in biclusters. Nevertheless, when using any evaluation measure in a specific heuristic for biclustering, there exist other objectives that are needed to be taken into account, such as the size of the biclusters, gene expression variance or overlapping amount, among others.

In this work we present a new biclustering approach combining a genetic algorithm with VE^t [15]. Our algorithm presents the advantage of being configurable with parameters. These parameters may be setting in order to modify the characteristic of the results.We conducted several experiments on both synthetic and real data from microarrays. Since many generations are needed in order to come up to good solutions, we have implemented our evolutionary algorithm parallel computation, making thus good use of the computer resources.

978-1-4577-1676-8/11/$26.00 ©2011 IEEE

II. PARALLEL GENETIC ALGORITHM

Genetic algorithms are classified as population-based meta-heuristics for combinatorial optimization, iteratively trying to improve a candidate solution with regard to a given measure of quality. Genetic algorithms start with a set of possible solutions instead of a single one. This characteristic allow genetic algorithms to explore a larger subset of the whole space of solutions, at the same time as it helps them to avoid becoming trapped at a local optimum. These reasons made genetic algorithms very suited to the biclustering problem.

Regarding encoding, we have adopted the same individual representation in other evolutionary biclustering works [12], where each bicluster is represented by a fixed sized binary string in which a bit is set to one if the corresponding gene or condition is present in the bicluster, and set to zero otherwise.

Starting by an initial population, genetic algorithms select some individuals and recombine them to generate a new population of individuals. This process is repeated for a number of generations until the algorithm converges or a certain stop criteria is met. Our biclustering algorithm comprises two processes, one inner to the other. The inner procedure corresponds to the genetic algorithm, which starts with an initial population and iteratively improves it in order to come up to a good bicluster solution. A coverage sequential procedure is used to produce a predefined number of biclusters n, invoking n times the inner algorithm.

In the following subsections we explain the initial population and generational change strategies, which have also been parallelized to improve time performance. In both cases, there exists no communication among the different activities, thus reducing synchronization and coordination times. Furthermore, we have no shared resources for writing, thus avoiding concurrent accesses.

A. Initial Population

In order to design the initial population strategy, we conducted several tests in which perfect biclusters were hidden in random data matrices. Using random strategies similar to the ones in [13] or [11], the algorithm did not always find the best solution. Nevertheless, our algorithm always converged to the best solution when the initial population contained at least one 3×3 sized bicluster representing a partial solution.

The initial population we have adopted consist in randomly generating 3×3 sub-matrices, henceforth seeds. The key is to generate much more seeds than the size of the population and then select the best ones. The probability of a randomly generated seed to be part of the solution can be computed as the number of possible seeds in the solution divided by the number of possible seeds in the whole data matrix, as in equation 1, where M, N, $|I|$ and $|J|$ are the number of rows and columns of the microarray data matrix and the solution, respectively.

$$\frac{Favorable\_seeds}{Total\_seeds} = \frac{\binom{|I|}{3} \times \binom{|J|}{3}}{\binom{M}{3} \times \binom{N}{3}} \quad (1)$$

Thus, our algorithm computes the number or seeds that needed to be generated in order to have at least one of them included in the solution. This can only be done with synthetic data, but it also gives us an idea of the number of seeds to generate in the case of real data.

The initialization procedure is carried out once per bicluster. Nevertheless, when looking for very small biclusters, or when the size of the microarray is very big, this initialization could take a very long time. We have therefore parallelized this phase using a pool of threads the size of the available cores in the computer, creating afterwards as many activities as cores. The task of each activity will be to generate a number of seeds such that the total number of seeds are generated by all the activities.

B. Generational change

Generational change is the mechanism that allows the population to improve its individuals, according to the fitness function and trying to converge to the optimal solution. Several operators are involved in this process. Firstly, a selector is needed in order to choose the individuals from one generation to the next. These individuals can be incorporated in the next generation in several ways: replicating themselves, being mutated, being crossed with other(s) individual(s) or by the combination of some of this operators.

Elitism is applied in order to ensure the convergence of the algorithm [16]. Also, a mutated copy of the best individual is incorporated into the next population. The rest of individuals are computed by selecting one or two individuals and applying crossover or/and mutation. We have used tournament of size 3 as selection mechanism. The 80% of the remaining individuals are generated by the crossover of two previously selected chromosomes, while the other 20% individuals correspond to replications. The resulting offspring is mutated with a certain probability in both cases.

Three distinct crossover operators are used in our algorithm with equal probability: one-point crossover, two-points crossover, and uniform crossover. We have also applied two different mutation operators: the simple one and the uniform one, with 0.2 and 0.0001 probabilities, respectively.

The number of generations (iterations) is 1500 generations, although if there is no significant improvement after 150 consecutive generations, the iterations are stopped. This number of generations has been set experimentally using synthetic data. In our algorithm, the evaluation is performed at the same time as the creation of new offsprings, allowing us to parallelize the whole generational change process. We have used again a pool of threads the size the number of cores in the computer. In this case, each activity submitted to the pool would be responsible for the creation of a new chromosome, together with its evaluation.

III. FITNESS FUNCTION

This section presents the distinct objectives used in our algorithm, as well as the reasoning on how they have been put together into a single fitness function.

We have used VE^t [15] as the main part of the fitness. It has to be minimized since its optimal value is 0, and it has a linear increasing behaviour when the amount of error in a bicluster gets bigger, measured according to the distance from its nearest perfect pattern.

The final fitness function is shown in equation 2 and is made up of three different terms, referring to the quality metric, volume and overlap. Each term is multiplied by a weight which defines its importance when evaluating a candidate solution. The default values for the weights in equation 2 have been obtained experimentally, but they can be modified according to the user preferences. Next subsections explain the terms involved in the evaluation of the solutions.

$$Phi(\mathcal{B}) = \frac{VE^t(\mathcal{B})}{VE^t(\mathcal{M})} + w_v \cdot Vol(\mathcal{B}) + w_{ov} \cdot Overlap(\mathcal{B}) \quad (2)$$

A. VE^t for the Evaluation of Biclusters

Transposed Virtual Error (VE^t) [15] is an evaluation measure for biclusters based on the concepts of expression patterns. It is computed by first creating a *Virtual Condition*, which is a vector containing the means of every row in the bicluster. Afterwards, a process of standardization is carried out both the bicluster data and the virtual condition. In the case of the bicluster the standardization is computed by using the means and deviations per column. Finally, VE^t measures the differences between the standardized values for every experimental condition and the standardized virtual condition.

The range of values of VE^t depends of the values in a microarray. Although the algorithm pursuit to minimize it, the weight of the other terms of the fitness function would have to be recomputed when using a different microarray. In order to avoid this situation, we divide it by the VE^t value of the whole microarray. This value is not necessarily the maximum VE^t, but it is certainly a good upper limit. First term in equation 2 represents the evaluation metric for biclusters.

B. Volume Term in the Fitness Function

In this subsection we study the different possibilities for tackling with biclusters sizes in the fitness function. At this point we have two contrary objectives to be optimized. On the one hand, VE^t has to be minimized and normally the smaller a bicluster is, the lower VE^t will be. On the other hand, the volume has to be maximized and the general tendency is that bigger biclusters will have bigger values for VE^t. In order to design the volume term for the fitness we took into account the following issues:

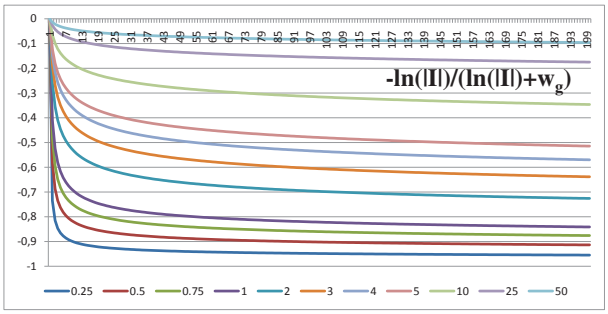

Figure 1: Final volume term representation

- Use of a *logarithmic scale*. Little changes in the number of rows or columns would not have a significant effect.
- *Two separated terms* for number of genes and conditions. This is necessary for avoiding too unbalanced biclusters, but also desirable in order to allow to configure each dimension size independently.
- *Fixed range*. Minimum and maximum values for both terms (gene and condition sizes) should not be dependant on any parameter value.

The final design of the term for the volume is the one shown in equation 3, where $|I|$ and $|J|$ refer to the number of genes and conditions, respectively, and w_g, w_c are the configuring parameters.

$$Vol(\mathcal{B}) = \left(\frac{-\ln(|I|)}{\ln(|I|) + w_g}\right) + \left(\frac{-\ln(|J|)}{\ln(|J|) + w_c}\right) \quad (3)$$

The graphical representation of one of the terms in equation 3 can be seen in figure 1. Although the range of all the functions represented are the same, those functions whose constant value is greater decrease slower. Depending on the value of the constant used, the term will have more influence over the fitness function at the beginning of the algorithm. At a certain point, increasing the number of rows or columns for a certain solution would not compensate the lose of quality, according to VE^t. The moment in which the algorithm stop increasing the size of the solutions and focus on improving the quality depends on the value of the constant used. The smaller the constant is, the sooner the algorithm will stop increasing the size. In figure 1 it can be clearly seen that for the smaller value of w_g represented ($w_g = 0.25$), the function decreases slower from a smaller value of m than for greater values of w_g.

Increasing the constant associated to rows (w_g) will produce biclusters with a greater number of genes, while increasing the constant associated to columns (w_c) will produce biclusters with more experimental conditions. Moreover, when adding equation 3 to the fitness function it is also necessary to include a weight w_v which will put the different terms involved on the same level.

C. Controlling the Overlapping

Overlapping among biclusters is usually permitted but controlled in the literature [17]. Overlapping differs from VE^t and volume in the sense that it cannot be evaluated on a bicluster by itself. The way in which overlapping is controlled differs from ones authors to others. Cheng and Church [5] try to avoid overlapping by replacing in the microarray data those values contained in each found bicluster by random ones. The main drawback of this strategy is that the replacement does not really avoid including those values in future biclusters. Therefore, if a bicluster is overlapped with a former one, that means that this new bicluster has been found using random values instead of the real ones.

In our work, we have adopted a strategy similar to the one used in [11], where a matrix of weights the size of the microarray is initialized with zero values at the beginning of the algorithm. Every time a bicluster is found, the weight matrix is updated increasing by one those elements contained in the bicluster.

$$Overlap(\mathcal{B}) = \frac{\sum_{i \in I, j \in J} W(e_{ij})}{|I| \times |J| \times (n-1)} \quad (4)$$

We have incorporated into the fitness function the last term shown in equation 4 in order to limit the overlap among biclusters. I and J refers to the sets of rows and columns in the bicluster $\mathcal{B}$, respectively. W represents the weight matrix and e_{ij} values corresponds to the elements of the bicluster $\mathcal{B}$. This equation computes how many times had the elements of $\mathcal{B}$ appeared in any former biclusters, and divides it by the size of $\mathcal{B}$ and the order of the solution minus one. This way, we are being more permissive with the latest solutions, and also enclosing the overlapping factor in the interval [0,1]. Furthermore, a weight w_{ov} is used to control the level of overlapping that the user is intended to admit.

IV. EXPERIMENTS

Synthetic data experiments were carried out with two purposes: tune the algorithm configuration and stablish comparisons when the solution is known. Experiments on real data have also been carried out with two well-known microarray benchmark data sets.

The experiments conducted for time comparison purposes have been carried out in a personal computer with an Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40GHz processor, 4GB of main memory and Windows 7 operating system.

A. Synthetic Data Experiments

We have randomly generated matrices the size of one of the most tested benchmark microarrays in biclustering: yeast *Saccharomyces cerevisiae* cell cycle expression dataset [18], made up of 2884 rows and 17 columns. We have also defined several sizes for the inclusion of perfect biclusters: 20×10, 60×12, 100×13, 150×15 and 200×16. For each of these sizes we have generated a perfect bicluster according to a combined shifting and scaling patters and 5 different random artificial microarrays (2884×17). Thus, each of the 5 different sized perfect biclusters has been inserted into 5 different microarrays in random positions. Furthermore, we have also generated the same number of microarrays adding noise to the data with random values generated from normal distribution, with mean equals to 0 and deviation equals to 0.25. All in all, there are 50 different experiments, 25 in which the biclusters follow a perfect pattern and 25 where noise has been included in the data. Each experiment has been run twice, with both the parallel algorithm and the non-parallel one, in order to establish a time performance comparison among them. We have also run each experiment several times and computed the mean of the obtained results.

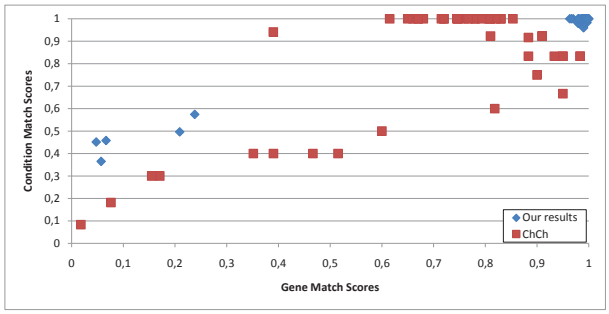

Figure 2: Gene and Condition Match Scores

We have compared the obtained results of all the set of experiments with those obtained using Cheng and Church Algorithm [5]. Match score performance measure [3] has been used to measure the extend to which the found bicluster adjusts to the inserted one.

Let $\mathcal{B}_1(I_1, J_1)$ and $\mathcal{B}_2(I_2, J_2)$ be two biclusters. Gene match score is defined as $S_I(I_1, I_2) = \frac{|I_1 \cap I_2|}{|I_1 \cup I_2|}$ and condition match score is defined as $S_J(J_1, J_2) = \frac{|J_1 \cap J_2|}{|J_1 \cup J_2|}$, varying from 0, where the two biclusters have no elements in common, to 1, when both biclusters are the same.

Figure 2 displays the gene and condition match scores of the executions of both our algorithm and Cheng and Churchs one. X-axis represents gene match scores and Y-axis represents condition match scores. Each square refers the comparison of a bicluster found by Cheng and Church algorithm and the equivalent solution, while each rhombus refers to the comparison of a bicluster found by our algorithm and its corresponding solution. Most of the rhombus are located in the right top corner, which means that most of the biclusters found by our algorithm match almost completely the solution. Only five biclusters have very low gene match score, below 0.3, corresponding to biclusters in which noise had been added. Many of the biclusters found by Cheng and Church algorithm are also located near the right top part of the graph, although not so close to the corner.

Table I: Paralellization factors for synthetic data

BicSize	MinGenerations	GenChangeF	InitializationF
20x10	428 (P)	3.47±0.81	3.56±0.65
60x12	637 (NP)	3.72±0.68	3.93±0.06
100x13	556 (NP)	3.97±0.71	3.74±0.02
150x15	689 (NP)	3.82±0.54	3.63±0.09
200x16	844 (NP)	3.89±0.54	3.59±0.18

Table II: Summary of the results for Yeast and Lymphoma datasets

Average	Yeast		Lymphoma	
	EA(0.5)	Ch&Ch	EA(0.5)	Ch&Ch
VE^t	0.051±0.02	0.10±0.11	0.215±0.04	0.57±0.10
Genes	54.32±28.77	166.71±226.4	57.5±41.11	269.22±204.71
Conds.	11.21±3.23	12.09±4.39	18.44±7.01	24.5±20.92
Overlap	0.017±0.009	0.064±0.08	0.004±0.002	0.064±0.043

Table I shows the parallelization factor on both population initialization and generational change. Column *MinGenerations* shows the minimum number of generations of both parallel (P) and non-parallel (NP) algorithms needed to reach the stop criteria for one execution. The means and deviations of the parallelization factors for generational changes are shown in the third column, while last column shows the means and deviations of the parallelization factors for several executions of the initialization procedure. The means are close to the number of processors (4) in both cases, meaning that we have successfully minimized synchronization and coordination times. Deviation values are higher for the situations in which the number of executions are higher (generational change) or where the process takes longer to end (initialization for the smallest bicluster), due to the intervention of the operating system.

B. Real Data Experiments

The proposed algorithm has been applied to two benchmark real-life data sets, yeast *Saccharomyces cerevisiae* cell cycle dataset [18], containing 2884 genes and 17 conditions, and human large B-cell Lymphoma [19], consisting of 4026 genes and 96 conditions. Both datasets can be downloaded from http://arep.med.harvard.edu/biclustering, as well as 100 biclusters found for each dataset by Cheng and Church [5].

We provide default values for the different weights in the fitness function, obtained experimentally. w_v and w_{ov} have been set to 5.0, while w_g and w_c have been set to 0.25 and 0.5, respectively. Increasing these two weights leads the algorithm towards bigger biclusters, or vice versa, as explained in section III.

Table II summarizes the results for both datasets. For each dataset, first column corresponds to one execution of our evolutionary algorithm (EA). For each execution, the mean of VE^t, number of genes, number of conditions and overlap of 100 biclusters are reported, and also the standard deviations are given. The same metrics of the 100 biclusters obtained by Cheng&Church algorithm are reported in the second column for each dataset.

Regarding yeast dataset, it can be seen in table II that the mean of the VE^t values in the Ch&Ch algorithm is almost the double of the mean of VE^t for all our executions. Also the size of the biclusters found by Ch&Ch algorithm is much bigger that the ones found by our proposal.In fact, the number of genes vary from 2 to 989, being the first reported biclusters much greater than the last ones. Also, many of these biclusters are really clusters, since they contains the whole set of conditions. Our approach also produces biclusters of different sizes, but the range of the number of genes and conditions is not so wide, the minimum number of genes in a bicluster was 7, and the maximum 182. The minimum number of conditions was 6, and the maximum 17, although only 11 biclusters were obtained with the whole set of conditions (against 37 biclusters in Ch&Ch). The amount of overlap in the biclusters found by Ch&Ch is also much greater than for the evolutionary algorithm, almost the triple. It is important to notice that Ch&Ch algorithm tries to avoid overlapping among biclusters by introducing random values once a bicluster is found. This means that biclusters containing any amount of overlap with the previous ones has been computed using random values instead the original ones.

Table III: Paralellization factors for real data

Dataset	MinGenerations	GenChangeF	InitializationF
Yeast	838(NP)	3.66±0.57	3.30±0.66
Lymphoma	1216(NP)	3.92±0.7	3.75±0.05

The results for the lymphoma dataset can also be seen in the two last columns of table II. We can draw similar conclusions than for the yeast dataset. In general, our algorithm produces steadier results, in the sense that the biclusters obtained are diverse in volume but no so disparate as for Ch&Ch. Note that the deviation of our executions are much lower than for Ch&Ch. The minimum and maximum genes in a bicluster produced by Ch&Ch are 2 and 757, respectively, while our algorithms produces biclusters between 4 and 310. The number of conditions varies from 7 to 96 for Ch&Ch and from 10 o 61 in our approach. Also, the mean of VE^t values for our biclusters is almost the half than the mean of Ch&Ch biclusters, and the overlapping factors are up to six times the overlapping factors of our algorithm.

Table III summarizes the time comparison for the real datasets, in the same format as in table I. Note that initialization factor is slightly lower for the yeast dataset. This can be explained by the fact that in this dataset there exist much more seeds with VE^t equals to zero. The initialization strategy for the real datasets is to generate seeds until a predefined number of seeds is reached (one million for yeast and lymphoma), or until it finds the population size number of seeds with VE^t equals to zero. Since the seeds are generated independently in the different processors, this second conditions is easier to be met in the non-parallel algorithm. Nevertheless, the time difference given by the initialization factor is still important.

V. CONCLUSIONS

In this work we present a genetic parallel algorithm for the biclustering problem, in which the fitness function has been carefully studied in order to guide the algorithm towards the best possible solutions attending to three different objectives, two of them to be minimized (overlap and evaluation measure) and the other to be maximized (volume). We also provide guidelines for its personalization. VE^t has been used as the metric for quantifying the goodness of the candidate solutions. VE^t [15] is based on the concepts of behavioural patterns and overcomes MSR drawbacks, being able to recognize both shifting and scaling patterns in gene expression data either separately or simultaneously. The parallelization of the algorithm has been implemented in two different stages, corresponding to the most computationally expensive phases: initialization and generational change. We have performed several experiments on both synthetic and benchmark real data, confronting the results with those obtained by Cheng and Church, and also computing the time performance with regard to a non-parallel version of our algorithm, where the parallelization factor is almost reached.

ACKNOWLEDGMENT

This research is supported by the Spanish Ministry of Science and Technology under grant TIN2007-68084-C02-00 and the Junta de Andalucía Research Program.

REFERENCES

[1] C. Tilstone, "Dna microarrays: Vital statistics," *Nature*, vol. 424, pp. 610–612, 2003.

[2] M.B. Eisen, P.T. Spellman, P.O. Brown and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," *PNAS*, vol. 95, pp. 14 863–14 868, 1998.

[3] A. Prelić, S. Bleuler, P. Zimmermann, and et al., "A systematic comparison and evaluation of biclustering methods for gene expression data," *Bioinformatics*, vol. 22, pp. 1122–1129, 2006.

[4] A. Tanay, R. Sharan, and R. Shamir, "Discovering statistically significant biclusters in gene expression data," *Bioinformatics*, vol. 18, pp. 136–144, 2002.

[5] Y. Cheng and G. M. Church, "Biclustering of expression data," in *Proceedings of the 8th International Conference on Intellingent Systemns for Molecular Biology*, La Jolla, CA, 2000, pp. 93–103.

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

[7] A. Mukhopadhyay, U. Maulik, and S. Bandyopadhyay, "A novel coherence measure for discovering scaling biclusters from gene expression data," *Journal of Bioinformatics and Computational Biology*, vol. 7, no. 5, pp. 853–868, 2009.

[8] X. Liu and L. Wang, "Computing the maximum similarity bi-clusters of gene expression data," *Bioinformatics*, vol. 23, pp. 50–56, 2007.

[9] J. Liu, Z. Li, X. Hu, and Y. Chen, "Biclustering of microarray data with mospo based on crowding distance." *BMC bioinformatics*, vol. 10 Suppl 4, no. Suppl 4, pp. S9+, 2009. [Online]. Available: http://dx.doi.org/10.1186/1471-2105-10-S4-S9

[10] W.-H. Yang, D.-Q. Dai, and H. Yan, "Finding correlated biclusters from gene expression data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, pp. 568–584, 2011.

[11] F. Divina and J. S. Aguilar-Ruiz, "Biclustering of expression data with evolutionary computation," *IEEE Transactions on Knowledge & Data Engineering*, vol. 18, no. 5, pp. 590–602, 2006.

[12] S. Mitra and H. Banka, "Multi-objective evolutionary biclustering of gene expression data," *Pattern Recognition*, vol. 39, no. 12, pp. 2464 – 2477, 2006, bioinformatics. [Online]. Available: http://www.sciencedirect.com/science/article/B6V14-4JRVB39-1/2/bb276259c35cdbcaa8a90a50d9d0c66f

[13] A. Mukhopadhyay, U. Maulik, and S. Bandyopadhyay, "Finding multiple coherent biclusters in microarray data using varible string length multiobjective genetic algorithm," *IEEE Transactions on Information Technology in Biomedicine*, vol. 13, no. 6, 2009.

[14] X. Xu, Y. Lu, A. K. H. Tung, and W. Wang, "Mining shifting-and-scaling co-regulation patterns on gene expression profiles," in *22nd International Conference on Data Engineering (ICDE'06)*, 2006, pp. 89–99.

[15] B. Pontes, R. Girldez, and J. S. Aguilar-Ruiz, "Measuring the quality of shifting and scaling patterns in biclusters," in *Lecture Notes in Computer Science*, vol. 6268, 2010, pp. 242–252.

[16] C. Coello Coello, "Evolutionary multi-objective optimization: a historical view of the field," *Computational Intelligence Magazine, IEEE*, vol. 1, no. 1, pp. 28 – 36, feb. 2006.

[17] A. P. Gasch and M. B. Eisen, "Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering," *Genome Biology*, vol. 3(11), p. research0059.10059.22, 2002.

[18] R. Cho, M. Campbell, E. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T. Wolfsberg, A. Gabrielian, D. Landsman, D. Lockhart, and R. Davis, "A genome-wide transcriptional analysis of the mitotic cell cycle," *Molecular Cell*, vol. 2, pp. 65–73, 1998.

[19] A. A. Alizadeh, M. B. Eisen, R. E. Davis, and et al., "Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling," *Nature*, vol. 403, pp. 503–511, 2000.