A Novel Evolutionary Algorithm for Bi-clustering of Gene Expression Data based on the Order Preserving Sub-Matrix (OPSM) Constraint

Hongchan Roh and Sanghyun Park

***Abstract*—Biclustering is a popular method which can reveal unknown genetic pathways. However, even though many algorithms have been suggested, no overwhelming algorithm has been suggested, due to its significant search space, until now. In this respect, several evolutionary algorithms tried to address this problem utilizing the powerful search capability of Evolutionary Computation (EC). However, most algorithms focused on exploiting the Mean Square Residue (MSR) measure which was proposed by Cheng and Church. The Order Preserving Sub-Matrix (OPSM) constraint was rarely considered even though it promises more biologically relevant biclusters than the MSR measure. The goal of this paper is to design an EC algorithm which ensures biologically significant biclusters by using the OPSM constraint and better biclusters than the original OPSM algorithm. We designed a novel encoding method and evolutionary operators suitable for the OPSM constraint. To efficiently explore the search space, we modulized our evolutionary algorithm and applied the co-evolution concept. Through a set of experiments, it was confirmed that our algorithm outperformed a representative EC biclustering algorithm based on CC and the original OPSM algorithm.**

I. Introduction

Recently, biclustering methods have been vigorously researched to discover local patterns in gene expression data. Whereas traditional clustering techniques such as hierarchical clustering [1] and k-means clustering [2] requires clustered genes to behave similarly over all the experimental conditions, biclustering requires genes in the same cluster to behave similarly over a subset of the conditions of gene expression data. This specified clustering concept is useful to uncover genetic pathways that are activated only over some of the conditions. The problem of finding a minimum set of biclusters is a generalization of another problem such as covering a bipartite graph, which has been shown to be NP-hard [15].

Hartigan [3] first introduced the biclustering concept and later, in 2000, Cheng and Church [4] first used this concept in biclustering gene expression data. After Cheng and Church, various methods regarding biclustering gene expression have been suggested. Among them, the Order Preserving Sub-Matrix model (OPSM) [5] is a representative biclustering method concerning the discovery of one or several submatrices in a gene expression matrix in which the expression levels of the selected genes induce the same linear ordering of the selected conditions. Later, [6] extended OPSM by assigning similar expression levels equal ranks.

Evolutionary Computation (EC) performs well in addressing complex optimization problems. It has excellent exploration power that provides the capability of escaping from local optima and working well when solutions to a problem contain complex interacting parts [16]. In addition, EC has been applied for problem solving in various domains such as planning [17], design [18], scheduling [19]-[20], simulation and identification [21], control [22], and classification [23]-[26].

In this respect, EC is very suitable for searching biclusters, and thus it has been applied to several biclustering approaches, combined with the previous well known biclustering methods such as Cheng and Church (CC), and the OPSM.

Several EC-based biclustering algorithms [7]-[10], which utilize CC's mean squared residue (MSR) measure as a fitness function score, have been proposed. An EC-based biclustering algorithm [12] with its own measure which has many common aspects with Sequential Evolutionary Biclustering (SEBI) [10] has been proposed. An EC-based biclustering algorithm [11] which utilizes the OPSM constraint has been also suggested. However, in the true sense of the word, [11] is not a biclustering algorithm because it can operate with not a subset of conditions but all the conditions. It's specially designed for gene expression data whose conditions are time series. Among them, SEBI is a comprehensive algorithm for the other algorithms. It tried to find biclusters which have an MSR score lower than the user-defined threshold while reducing overlapped areas between biclusters, growing bicluster size, and maximizing row variance.

Most of the previous EC algorithms for biclustering have not provided biologically significant biclusters and also cannot fully utilize the search power of EC. Biclustering algorithms' performance depends on how much information about genetic pathways their results, such as biclusters, can provide. Therefore, previous non-EC-based biclustering algorithms are mostly evaluated by biological significance of biclusters they found. However, previous EC algorithms

Manuscript received July 5, 2008. This work was supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MOST) (No. R01-2006-000-11106-0).

H. Roh is with the Computer Science Department, Yonsei University, Seoul, Korea (corresponding author to provide phone: 82-2-2123-7757, fax: 82-2-365-2579, e-mail: fallsmal@cs.yonsei.ac.kr).

S. Park is with the Computer Science Department, Yonsei University, Seoul, Korea (e-mail: sanghyun@cs.yonsei.ac.kr).

except only one EC-based biclustering algorithm [8] didn't even try to measure the biological significance of their results. We can expect that their results, except the algorithms [11]-[12], would not have good biological significance because they used MSR measure of CC method as their fitness score. CC is revealed as a not ideal method for finding biologically significant biclusters in [13]. Its gene ontology (GO) annotation evaluation result was the worst among 5 tested biclustering algorithms. Every EC-based algorithm has the problem that its search space is so significantly large that evolutionary algorithms cannot efficiently work.

It can be expected that an EC algorithm using the OPSM constraint probably provides more biologically significant biclusters than the EC algorithms based on CC, since in [13], OPSM was the best algorithm in GO p-value evaluation. However, there is no OPSM based EC biclustering algorithm in the true sense of the word biclustering, even though OPSM showed very good performance in finding biologically significant biclusters in [13]. In addition, the original OPSM algorithm [5] cannot search globally because of its pruning process, so we expect that EC-based biclustering algorithms utilizing the OPSM constraint can find biclusters that the original OPSM cannot find.

One goal of this paper is to design an EC algorithm which can not only provide biologically significant biclusters satisfying the OPSM constraint, but can also find bigger biclusters than the original OPSM algorithm. The other goal is to evaluate our algorithm with the original OPSM in the aspect of bicluster size and with a representative EC algorithm based on CC in the aspect of biological significance.

Using the OPSM constraint, we designed an EC algorithm including an efficient encoding method that can reduce the search space, and genetic operators suitable for the OPSM constraint. In order to efficiently search biclusters, we modulized our EC algorithm for each module to handle searching biclusters having a certain condition length. In addition, using the concept of co-evolution, we maximized the EC's search capability.

In Section 2, the survey of related work is given. Section 3 describes our algorithm. Section 4 provides the experimental result. Finally, we conclude this paper in Section 5.

II. PRELIMINARIES

A. Introduction to Evolutionary Computation

EC is a population-based stochastic generate-and-test technique inspired by Darwinian evolution. Like survival of the fittest and selective pressure, EC addresses complex problems by evolving candidate solutions through computational approaches [10].

EC typically needs a population such as a set of candidate solutions. These solutions are evolved by the repeated mutations and repeated selections of more fit solutions. The candidate solutions in the population can be referred as *individuals*, or as chromosomes. The solutions can be encoded in different ways. A popular method is the binary string encoding, where each bit of the string has a particular meaning.

In each generation, selection is performed among the individuals in the generation's population according to the quality of each solution. The quality is assessed by a score function referred as the *fitness function*. Therefore, the better the fitness of an individual, the more possibilities the individual has of being selected for reproduction and the more parts of its genetic material will be passed on to the next generations.

After selection is performed, the survived individuals reproduce their offspring through the *crossover* and *mutation* process. In the crossover process, EC makes offspring by swapping genetic information between the selected individuals. The mutation process changes a very small part of the genetic information of each offspring to a random value.

By doing so, EC can efficiently explore the space of candidate solutions to a certain problem. Typically, this space is denoted as *search space*, which represents all the possible solutions each of which can be encoded as an individual.

B. Previous EC algorithms for Biclustering Problem

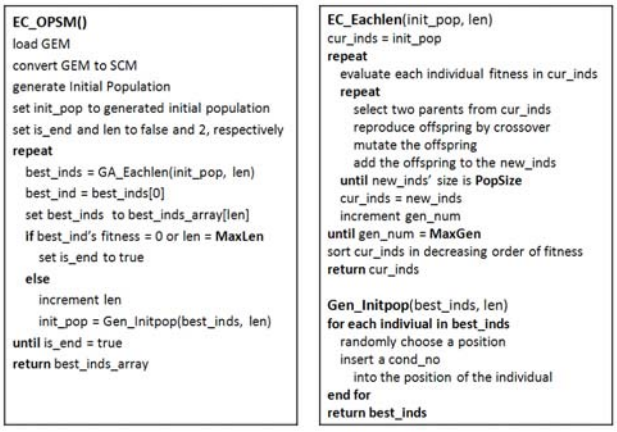
In the remainder of this paper, we will assume that the

1010100000|01010010

Fig. 1. SEBI's encoding

measurements of several experiments are given in terms of an m×n-matrix, E, where m is the number of considered genes and n the number of experiments. A cell e_{ij} of E contains a real value that reflects the abundance of mRNA for gene i under a experimental condition j.

SEBI [10] tried to find biclusters satisfying the following conditions: Bigger bicluster size which means including more genes and conditions; Smaller MSR score, at least lower than ð, a threshold defined by users; Relatively high row variance which means that a bicluster has high variance within a row; Low level of overlapping among biclusters in order to cover as more global search space.



(a) (b)

Fig. 2. The overall algorithm

In SEBI's EA, each individual represents a bicluster. SEBI encoded each individual by using the binary encoding where m bits and the following n bits respectively represent the selection of genes and conditions as demonstrated in Fig. 1. This means that if i_{th} bit is set among the first m bits, i_{th} gene is included in the bicluster, and if j_{th} bit is set among the following n bits, j_{th} condition is included in the bicluster.

Each individual is encoded by (m+n) bits, so the possible number of individuals which is equal to the size of search space is $2^{(m+n)}$.

C. OPSM

Given the above notation, the OPSM constraint can be described as follows: A submatrix is order preserving if there is a permutation of its columns (conditions) such that the sequence of (gene expression) values for each row (gene) is strictly increasing.

The original OPSM algorithm [5] finds a subset G of genes (|G| m) and a subset C of experiments (|C| n) such that the submatrix D of E defined by G and C maximizes a given score f(G,C) and is an Order Preserving Sub-Matrix (OPSM), see below. The score f reflects the probability for D to participate in the best OPSM. This algorithm evaluates from 2-length permutation to n-length permutation by this score function.

After sorting permutations by this score, the algorithm leaves the highest T permutations pruning the other permutations at each step. The T is the pruning threshold given by users. Permutations having (L+1)-length are generated by adding one condition to L-length permutations.

III. Algorithm Description

A. Encoding Method

Each individual represented as an L-length character string encodes an L-length permutation of condition numbers. Each character of the string represents a condition number. The character string is enough to express the permutations of condition numbers because typical values for m and n are in the ranges from 500 m 15000 and 10 n 150, given a m×n gene expression matrix (GEM) [5]. The search space generated by this encoding scheme is as follows:

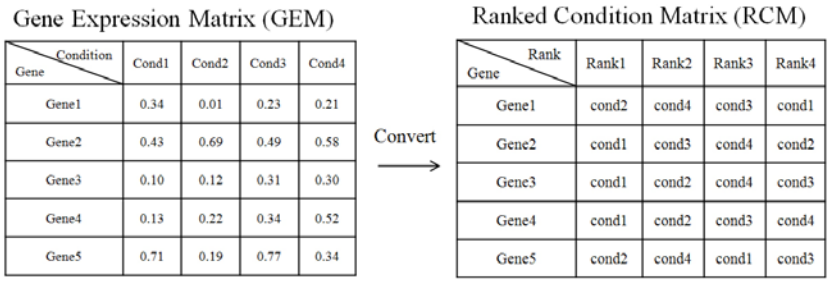
$$\sum_{k=2}^{n} {}_nP_k \tag{1}$$

In the range of typical genes and conditions, the search space generated by our encoding method is always smaller than the search space of the other EC-based biclustering algorithms [7]-[12] because all the algorithms uses binary encoding for the selection of genes and conditions. As mentioned above, the binary encoding scheme generates $2^{(m+n)}$ search space. Due to the fact that m is usually greater than n, our encoding scheme's search space is significantly smaller than the binary encoding scheme in general. In this respect, this encoding method helps our EC algorithm to efficiently work on finding biclusters.

B. Overall Algorithm

Fig. 2 shows the overall process of our algorithm consisting of the main process such as EC_OPSM, and sub-routines such as EC_Eachlen and Gen_Initpop. There are 3 global variables defined as parameters by users such as MaxLen, PopSize, and MaxGen which represents the maximum condition length, population size, and the number of maximum generation, respectively.

Our algorithm named Evolutionary Computation based on the Order Preserving Sub-Matrix constraint (ECOPSM) is modulized to efficiently apply EC to our biclustering problem as shown in Fig. 2a. In the EC_OPSM process, it executes an



(a) (b)

Fig. 3. Converting process from GEM to RCM

EC algorithm per each length permutation by calling EC_Eachlen function. This can reduce the search space and increase the search performance. It utilizes the previous EC result having L-length individuals in generating the initial population of (L+1)-length individuals.

As a first step of the EC_OPSM process, this algorithm converts a GEM to a Ranked Condition Matrix (RCM). As represented in Fig. 3, the GEM is first ranked by its expression value in each row. Then, each condition number is written to the RCM substituted for its expression value in the rank position of the row. Next, RCM's each row is transformed to a character string such that each character represents the condition number. Finally, RCM is stored as an array of character strings. This array size is equal to m, the total number of the genes in the GEM.

By converting the GEM to the RCM, ECOPSM can easily find out biclusters satisfying the OPSM constraint by examining whether the current individual string is a subsequence of each row string in RCM.

After successfully generating RCM, ECOPSM generates an initial population which has the PopSize of 2-length individual strings. The 2-length individuals are completely randomly generated by selecting 2 random numbers among 0 to n-1 without duplication.

In the loop of the EC_OPSM, the ECOPSM obtains the final population evolved by the GA_Eachlen function. Storing the returned population to L_{th} slot of the array of the final populations (best_inds_array[L]), it generates the initial population of (L+1)-length individuals by calling the Gen_Initpop function. The Gen_Initpop function makes each of (L+1)-length individuals by inserting a random number, that is not already included in each L-length individual, into a random position of the individual of the final popluation as shown in Fig. 2b. By doing so, the initial population of 3-length individuals is generated by the 2-length individuals, the initial population of 4-length individuals by the 3-length individuals, and so on.

If the best individual of the returned population's fitness is 0 or the EC_OPSM finally finishes finding the best n-length individuals, then it returns the array of the final populations.

C. *Evolutionary Computation*

The EC_Eachlen function in Fig. 2b retrieves the best individuals which has the specified length by the input parameter, Len. First, it evaluates the fitness of each individual in the initial population obtained by the input parameter. The fitness function is simply defined as follows:

$$f(X) = count(RCM, X) \qquad (2)$$

X is an individual, and count(RCM, X) is the function that counts the number of genes each of which has a subsequence equal to the string of X by using the stored string-array of RCM in the first step of EC_OPSM process. Using the counted genes and the conditions encoded in X, a bicluster is defined. This fitness evaluation is clear and reasonable, since OPSM's performance depends on the size of biclusters, and in this case the bicluster size is determined solely by the number of genes due to the condition length fixed as the given parameter, Len.

Parent selection is performed by Rank based Selection [27] such that the individuals in the population are ranked according to their fitness, and the expected value for each individual to be selected as parents depends on its rank rather than its absolute fitness. Offspring are reproduced by performing 1-point crossover between the selected two parents. If the crossover operation makes the individual have duplicated conditions, mandatory mutation is performed in order to replace the duplicated condition number with another condition number not included in the individual.

After the crossover operation, this function executes a slightly changed mutation, which is not toggling a bit in binary encoding but exchanging one character with the unused condition number in the individual string, to the individual.

D. *An Extension using Co-evolution Concept*

Co-evolution is a promising method that can increase the efficiency of evolutionary exploration in Evolutionary Computation.

In our algorithm, co-evolution between L-length individuals and (L+1)-length individuals is possible because if it generates L-length initial population from (L+1)-length individuals, then the better biclusters that have not been

searched by the previous evolutionary process can be found.

Therefore, we appended the co-evolution operation named Pair-Iteration in the loop of the EC_OPSM process. In the loop, EC_OPSM iterates the following co-evolution process Max Iteration (MaxIter) times. First, the (L+1)-length initial population is generated from the L-length final population returned by the EC_Eachlen function. Second, the (L+1)-length population is evolved through the process of the EC_Eachlen function with the generated (L+1)-length initial population given as the parameter of the function. Next, the L-length initial population is generated from the (L+1)-length final population returned by the EC_Eachlen function. Then, L-length population is evolved through the process of the EC_Eachlen function with the generated L-length initial population given as the parameter. In addition, the array of the final populations (best_inds_array) is updated with the returned final population at each step.

Through several experiments, this co-evolution process was empirically proven to be more cost-effective than merely increasing the number of max generation, MaxGen and the population size, PopSize.

TABLE I
ECOPSM PARAMETERS

Parameter	Description	Quantity
MaxLen	Maximum condition length	n
PopSize	Number of individuals in a population	300
MaxGen	Maximum generation number	300
Pc	Crossover rate	0.6
Pm	Mutation rate	0.01
Min	Parameter for Rank based selection	0 to 0.9
Max	Parameter for Rank based selection	1.1 to 2.0
MaxIter	Maximum pair-iteration count	15 to 20

IV. EXPERIMENTAL RESULT

A. Experimental Environment

We conducted experiments on two well-known data sets in order to assess the performance of the proposed method for finding biclusters. Both data sets are the yeast Saccharomyces cerevisiae cell cycle expression data sets. The first expression matrix is originated by Cho et al. [28] consisting of 2,884 genes and 17 experimental conditions. The second data set is the one provided by Gasch et al. [14], which contains 2993 genes and 173 conditions.

To assess the biological relevance of biclusters on the GEM for Saccharomyces, a quantitative measure is introduced that relates the biclustering outcomes to annotations by Gene Ontology (GO) Consortium. We measured the p-value for overrepresented GO categories using the latest updated (2008.5.22) gene annotation data provided by GO Consortium.

In order to compare our algorithm with the original OPSM algorithm, the bicluster size is measured with the row size and column size reported separately. In this case, the bicluster size is more important as a performance measure than the p-value, since both algorithms satisfy the OPSM constraint, and the only difference is the searching method.

The ECOPSM's parameters used by these experiments are represented in Table 1.

B. Performance Comparison with SEBI

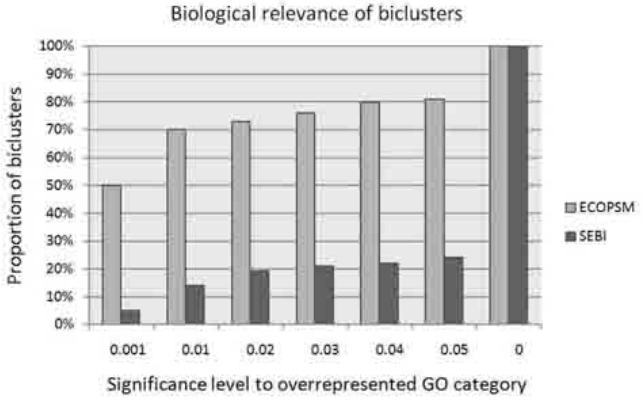


Fig. 4. Biological relevance of biclusters found by ECOPSM and SEBI

We conducted experiments on Cho et al.'s data to assess the biological significance of biclusters found by ECOPSM and SEBI algorithms. Since the biological relevance of biclusters found by the SEBI algorithm has not been reported, we implemented the SEBI algorithm based on the Bleuler and Zitzler's paper [10]. Cho et al.'s data were used since we needed to test our own implemented version of SEBI using the data used by the research [10]. We obtained 100 biclusters whose average gene number, condition number, and mean square residue were 13.03, 15.84, and 248.29, respectively, which were similar results to the experimental result in the research.

We measured the adjusted p-value of the found biclusters (The adjusted p-value represents the fraction (as a %) of 1000 null-hypothesis simulations having the genes of the tested bicluster with this single-hypothesis p-value or smaller). The adjusted p-value was compared by the adjusted p-value of ECOPSM's biclusters as shown in Fig. 4. The 100 biclusters found by SEBI algorithms participated in this comparison. To be compared with these biclusters, 100 biclusters were randomly chosen among the biclusters found by ECOPSM, which have 5 to 7 conditions. The reason why we didn't used all the found biclusters but randomly chosen 100 biclusters was to fairly compare them with the biclusters found by SEBI. ECOPSM found 300 biclusters every length and consequently, 900 biclusters were found, which have 5 to 7 conditions. Therefore, it would be a slight advantage for ECOPSM to use the whole 900 biclusters in comparison, so we applied the approach choosing randomly.

Fig. 4 demonstrates the proportion of biclusters which has the p-value lower than each values represented in the x-axis of the graph. 50 % of ECOPSM' biclusters have the adjusted

p-value lower than 0.001, whereas only 5 % of the SEBI's biclusters have the adjusted p-value lower than 0.001.

By this result, it was revealed that the ECOPSM generated more biologically significant biclusters than the SEBI algorithm.

C. Performance comparison with the original OPSM

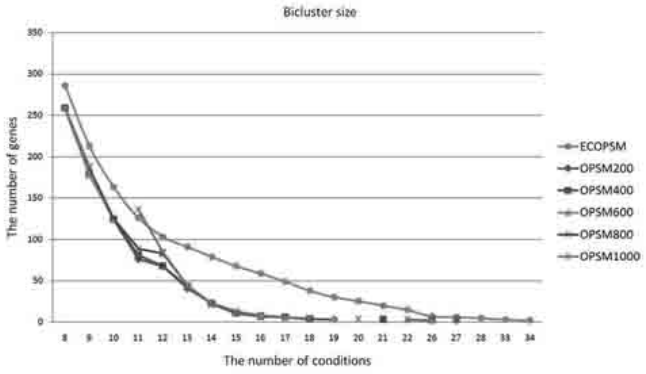


Fig. 5. Bicluster size of ECOPSM and OPSM

We conducted experiments on Gasch et al.'s data set and Cho et al.'s data set, comparing our algorithm with the original OPSM algorithm by measuring the size of biclusters. The size of the biclusters can be the measure of the biclusters' quality, since finding bigger biclusters is the same goal of the paper [5] that originally proposed the OPSM. For each OPSM having a certain number of conditions, the number of genes was measured. Since the original OPSM's performance varies with the pruning threshold, we executed the OPSM algorithms 10 times, increasing the threshold by 100 from 100 to 1000. We used the same OPSM program as Prelic et al. [13] utilized in their experiment.

In the experiment of Cho et al.'s data set, the ECOPSM and OPSM demonstrated the same performance in finding larger biclusters, since the data set was too small to differentiate the performance of both.

Fig. 5 shows the experimental result conducted on Gasch et al.'s data set. Except 11-length condition, the ECOPSM locates larger biclusters than the original OPSM. The number of genes included in biclusters decreased according to the increase in the number of conditions. It can be noticed that the threshold could not significantly increase the performance of the OPSM. ECOPSM's longest condition length having at least 2 genes was 34 whereas the original OPSM's longest condition length was 27.

D. Comprehensive Performance Comparison

The comprehensive comparison between the 3 algorithms was needed. Therefore, we compared the 3 algorithms by using the Gasch et al.'s data and GO p-value evaluation.

The 100 biclusters found by one run of SEBI were totally used in this comparison, with the MSR threshold 300 given as in the paper, and in the case of ECOPSM, for the comparison, the 100 biclusters were randomly chosen from the found biclusters which have 10 to 13 conditions. In the case of the original OPSM, the pruning threshold T was chosen as 1000 and all the found biclusters having 10 to 13 conditions participated in the comparison.

Fig. 6 represents the proportion of the biclusters found by each algorithm which has lower values than the specified values in the x-axis. It's remarkable that more than 30% of biclusters found by the ECOPSM have p-value lower than 10^{-130}. All the biclusters of the ECOPSM are distributed in the range of p-value from 10^{-58} to 10^{-157}. However, OPSM's biclusters are distributed in the range of p-value from 10^{-37} to 10^{-100}. Moreover, SEBI's biclusters are distributed in the very short range of p-value from 1 to 10^{-10}.

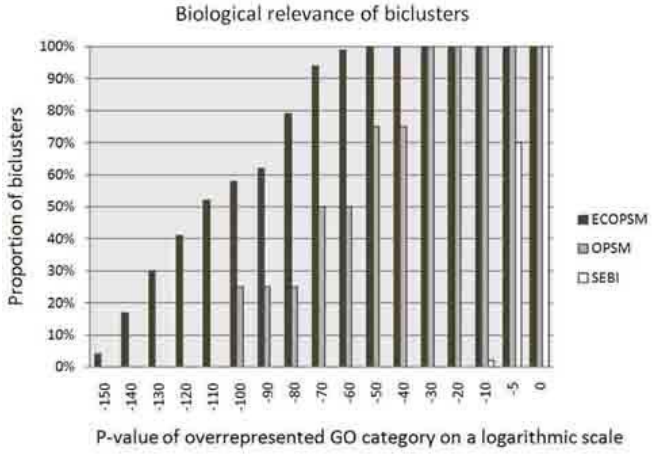


Fig. 6. Biological relevance of biclustering found by each algorithm

E. Various biclusters found by ECOPSM

The following graphs show the several biclusters chosen among the biclusters that ECOPSM found from Gasch et al.'s data.

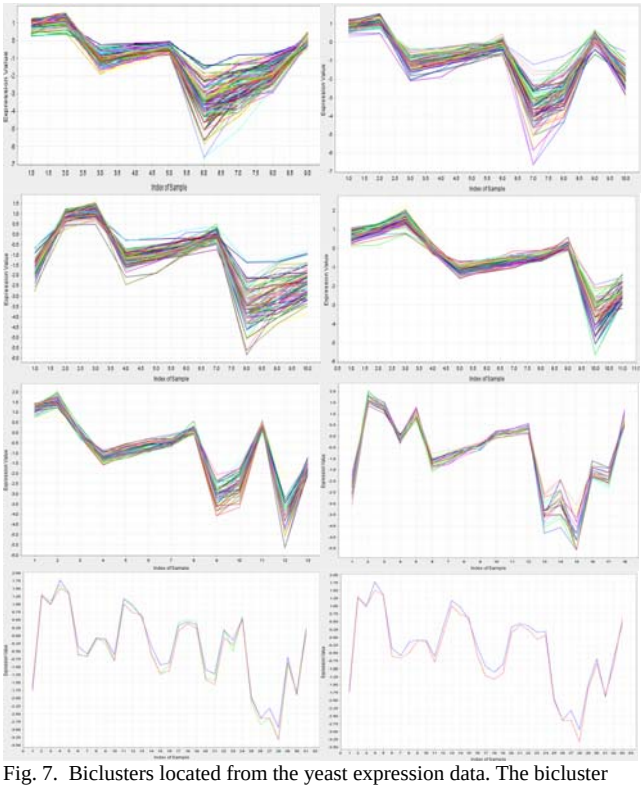


Fig. 7. Biclusters located from the yeast expression data. The bicluster features in each are reported in the following format (number of genes,

number of conditions, p-value) as follows in the order of left-to-right and top-to-bottom. (143, 10, $6.6*10^{-72}$), (102, 11, $2.9*10^{-76}$), (103, 11, $9*10^{-82}$), (103, 12, $2.10*10^{-151}$), (62, 14, $1.4*10^{-102}$), (24, 22, $9.2*10^{-39}$), (3, 32, $2.10*10^{-5}$), (2, 34, higher than 10^{-5}).

V. Conclusion

Recently, biclustering methods have been spotlighted as a method of uncovering genetic pathways.

This paper proposed an Evolutionary Computation (EC) algorithm which can efficiently locate biclusters using the Order Preserving Sub-matrix (OPSM) constraint.

In the design of this algorithm, we modulized the bicluster search process and adapted the co-evolution concept to efficiently explore the given search space.

In the set of experiments, our algorithm overwhelmed a representative EC algorithm based on mean square residue, a popular bicluster measure, in the aspect of finding meaningful genetic pathways, and also demonstrated that it found larger and better biclusters than the original OPSM algorithm.

This research can not only be used as a novel method in finding biologically significant biclusters, but can also be adapted as a method of designing EC algorithms that can find biclusters based on the OPSM constraint.

As a future research, we are considering to compare our algorithm with more various EC-based biclustering algorithms or non-EC-based biclustering algorithms.

References

[1] R. R. Sokal and C.D. Michener, "A statistical method for evaluating systematic relationships," *Univ. Kansas Sci. Bull.*, vol. 38, 1958, pp. 1409–1438.

[2] J. A. Hartigan, and M. A. Wong, "A k-means clustering algorithm," *Appl. Stat.*, vol. 28, 1979, pp. 100–108.

[3] J. A. Hartigan, "Direct clustering of a data matrix," *Journal of the American Statistical Association*, vol. 67, no. 337, pp. 123-129, 1972.

[4] Y. Cheng and G. M. Church, "Biclustering of expression data," in *Proc. 8th Int. Conf. Intelligent Systems for Molecular Biology*, 2000, pp. 93–103.

[5] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini, "Discovering local structure in gene expression data: the order-preserving sub-matrix problem," in *Proc. 6th Ann. Int. Conf. Computational Biology*,

[6] J. Liu, J. Yang, and W. Wang, "Biclustering in gene expression data by tendency," in *Computational Systems Bioinformatics Conf. (CSB 2004). IEEE*, 2004.

[7] S. Bleuler, A. Prelic, and E. Zitzler, "An EA framework for biclustering of gene expression data," in *Congress on Evolutionary Computation (CEC-2004)*, 2004, pp. 166-173.

[8] A. Chakraborty, H. Maka, "Biclustering of gene expression data using genetic algorithm," in *Proc. 2005 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, 2005, pp. 1–8.

[9] X. Fei, S. Lu, H. F. Pop, and L.R. Liang, "GFBA: a genetic fuzzy biclustering algorithm for discovering value-coherent biclusters," in *Proc. 2007 Int. Symposium on Bioinformatics Research and Applications*, 2007.

[10] F. Divina and J. S. Aguilar-Ruiz, "Biclustering of expression data with evolutionary computation," *IEEE Trans. Knowledge and Data Engineering*, vol. 18, no. 5, 2006, pp. 49-57.

[11] S. Bleuler and E. Zitzler, "Order preserving clustering over multiple time course experiments," in *Proc. EvoWorkshops 2005*, pp. 33-43, 2005.

[12] R. Giraldez, F. Divina, B. Pontes, and J.S. Aguilar-Ruiz, "Evolutionary search of biclusters by minimal intrafluctuation," in *Fuzzy Systems Conf. (FUZZ-IEEE 2007)*, 2007, pp. 1-6.

[13] A. Prelic, S. Bleuler, P. Zimmermann, A. Wille, P. B¨uhlmann, W. Gruissem, L. Hennig, L. Thiele, and E. Zitzler, "A systematic comparison and evaluation of biclustering methods for gene expression data," *Bioinformatics*, 22, 2006.

[14] A. P. Gasch, P. T. Spellman, C. M. Kao, O. Carmel-Harel, M. B. Eisen, G. Storz, D. Botstein, and P. O. Brown, "Genomic expression programs in the response of yeast cells to environmental changes," *Molecular Biology of the Cell*, vol. 11, 2000, pp. 4241–4257.

[15] J. Orling, "Containment in graph theory: covering graphs with cliques," in *Proc. Koninldijke Nederlandse Akademie van Wetenschappen*, vol. 39, 1977, pp. 211-218.

[16] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Springer-Verlag, 2003.

[17] D. E. Goldberg and L. Robert, "Alleles, loci, and the travelling salesman problem," in *Proc.1st Int. Conf. Genetic Algorithms*, 1985, pp. 154-159.

[18] P. J. Bentley and D.W. Corne, *Creative evolutionary systems*, Morgan Kaufmann Publishers Inc., 2001.

[19] T. Yamada and R. Nakano, "A Genetic algorithm applicable to large-scale job-shop problems," *Parallel Problem Solving from Nature*, vol. 2, Amsterdam: Elsevier Science Publishers, 1992.

[20] D. Corne, P. Ross, and H. L. Fang, "Fast practical evolutionary timetabling," in *Proc. Evolutionary Computing AISB Workshop*, pp. 251-263, URL:http://citeseer.nj.nec.com/corne94fast.html, 1994.

[21] D. K. Gehlhaar, G. M. Verkhivker, P. A. Rejto, C. J. Sherman, D. B. Fogel, L. J. Fogel, and S. T. Freer, "Molecular recognition of the inhibitor ag-1343 by hiv-1 protease: conformationally flexible docking by evolutionary programming," *Chemistry and Biology*, vol. 2, no. 5, 1995, pp. 317-324.

[22] G.F. Spencer, "Automatic generation of programs for crawling and walking," in *Proc. 5th Int. Conf. Genetic Algorithms (ICGA '93)*, 1993, pp. 654.

[23] D.B. Fogel, "Evolving behavious in the iterated prisoner's dilemma," *Evolutionary Computation*, vol. 1, no. 1, 1993, pp. 77-97.

[24] F. Divina and E. Marchiori, "Evolutionary concept learning," in *Proc. Genetic and Evolutionary Computation Conf.*, 2002, pp. 343-350.

[25] J. S. Aguilar-Ruiz, J. Riquelme, and M. Toro, "An evolutionary approach to estimating software development projects," *Information and Software Technology*, vol. 14, no. 43, 2001, pp. 875-882.

[26] J. S. Aguilar-Ruiz, J. Riquelme, and C. D. Valle, "Evolutionary learning of hierarchical decision rules," *IEEE Trans. Systems, Man, and Cybernetics*, Part B, vol. 33, no. 2, 2003, pp. 324-331.

[27] J. E. Baker, "Adaptive selection methods for genetic algorithms," in *Proc. 1st International Conf. on Genetic Algorithms and Their Applications*, 1985

[28] 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, 1998, pp. 65-73.


```
EC_OPSM()
load GEM
convert GEM to SCM
generate Initial Population
set init_pop to generated initial population
set is_end and len to false and 2, respectively
repeat
   best_inds = GA_Eachlen(init_pop, len)
   best_ind = best_inds[0]
   set best_inds to best_inds_array[len]
   if best_ind's fitness = 0 or len = MaxLen
      set is_end to true
   else
      increment len
      init_pop = Gen_Initpop(best_inds, len)
until is_end = true
return best_inds_array
```

```
EC_Eachlen(init_pop, len)
cur_inds = init_pop
repeat
   evaluate each individual fitness in cur_inds
   repeat
      select two parents from cur_inds
      reproduce offspring by crossover
      mutate the offspring
      add the offspring to the new_inds
   until new_inds' size is PopSize
   cur_inds = new_inds
   increment gen_num
until gen_num = MaxGen
sort cur_inds in decreasing order of fitness
return cur_inds

Gen_Initpop(best_inds, len)
for each indiviual in best_inds
   randomly choose a position
   insert a cond_no
      into the position of the individual
end for
return best_inds
```

Gene Expression Matrix (GEM)

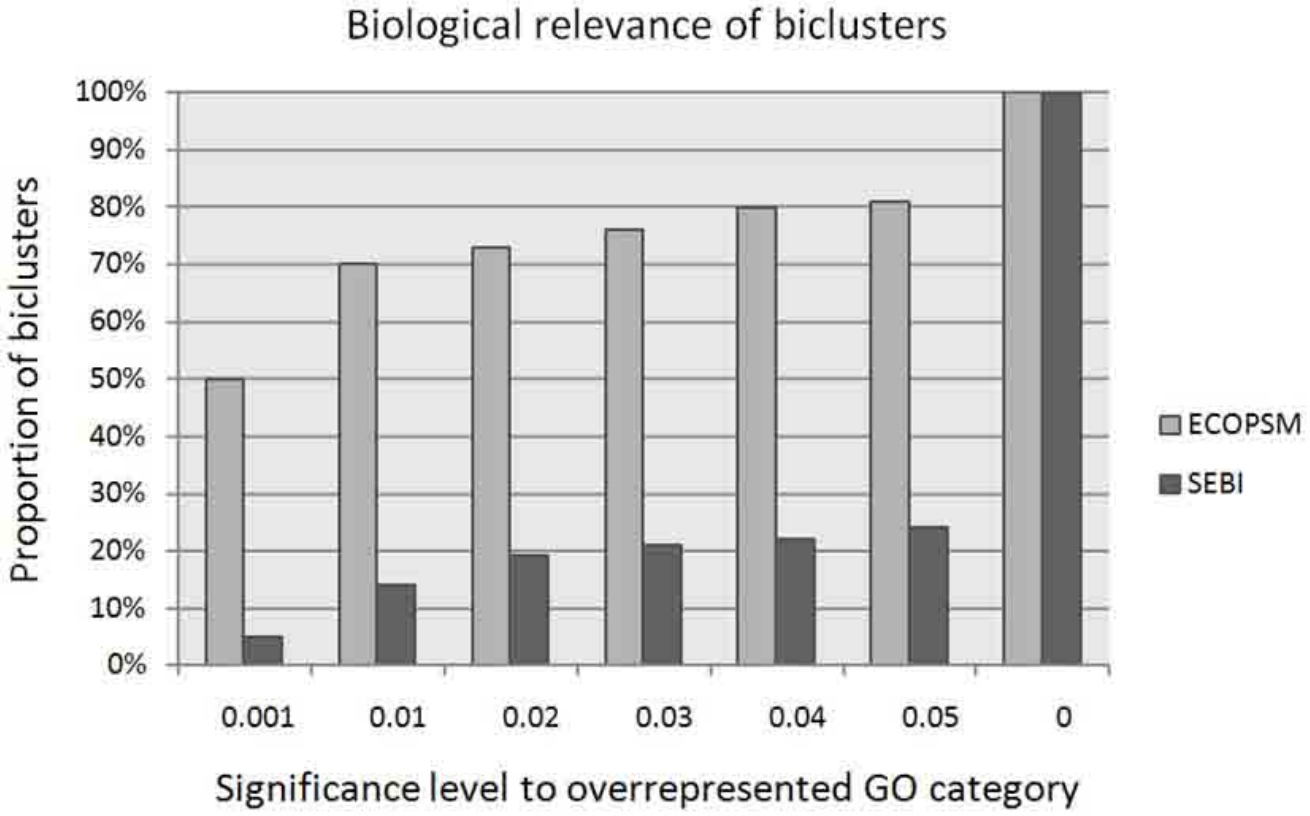
Condition / Gene	Cond1	Cond2	Cond3	Cond4
Gene1	0.34	0.01	0.23	0.21
Gene2	0.43	0.69	0.49	0.58
Gene3	0.10	0.12	0.31	0.30
Gene4	0.13	0.22	0.34	0.52
Gene5	0.71	0.19	0.77	0.34

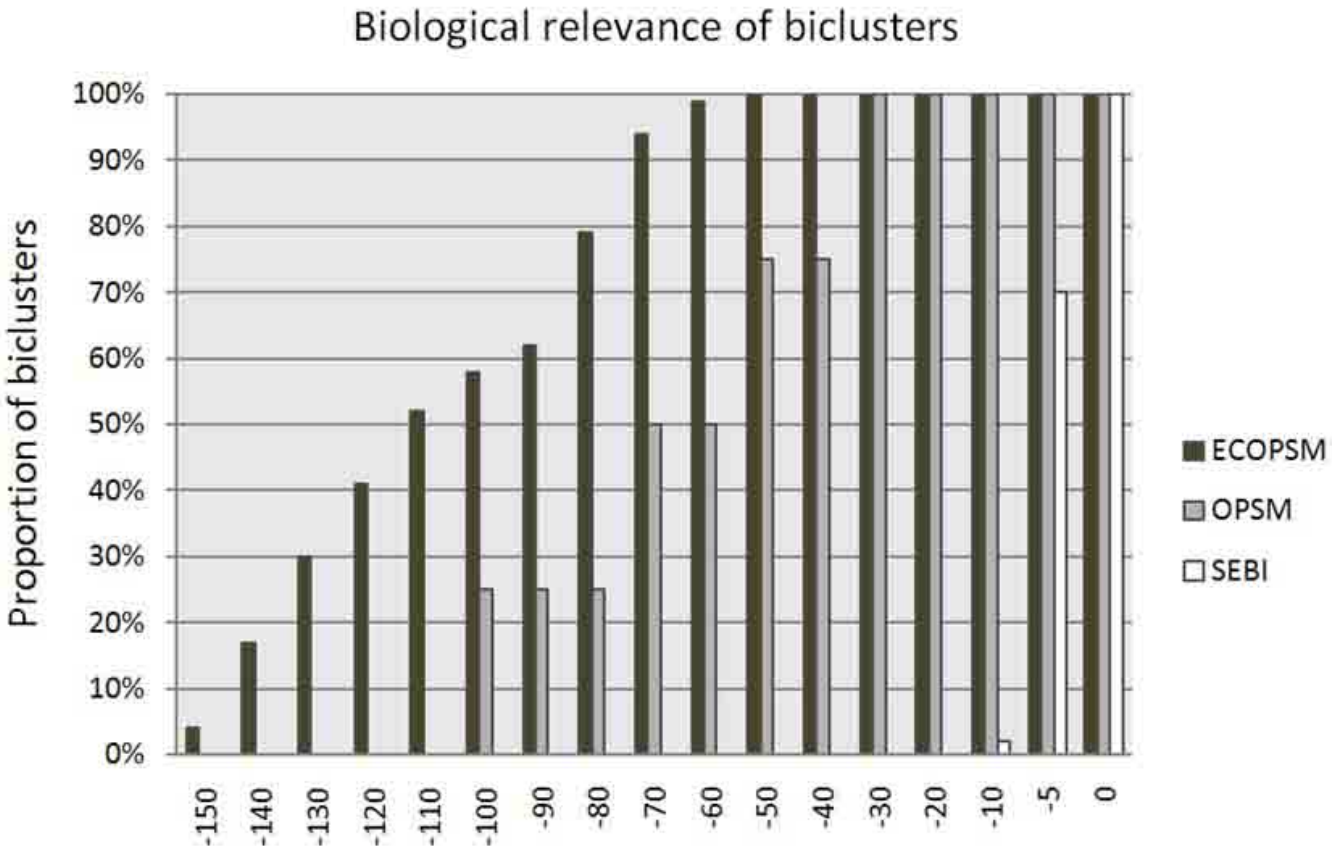
Convert →

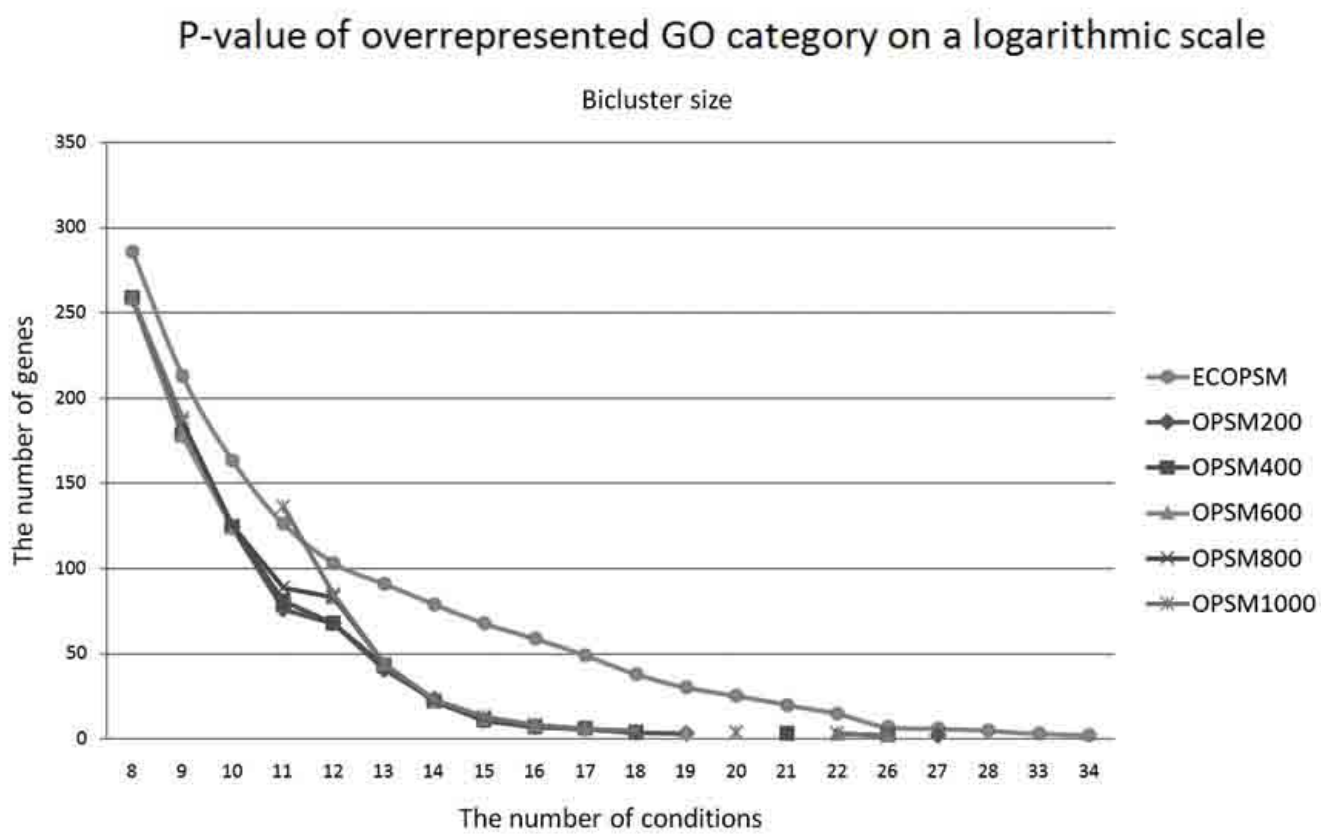
Ranked Condition Matrix (RCM)

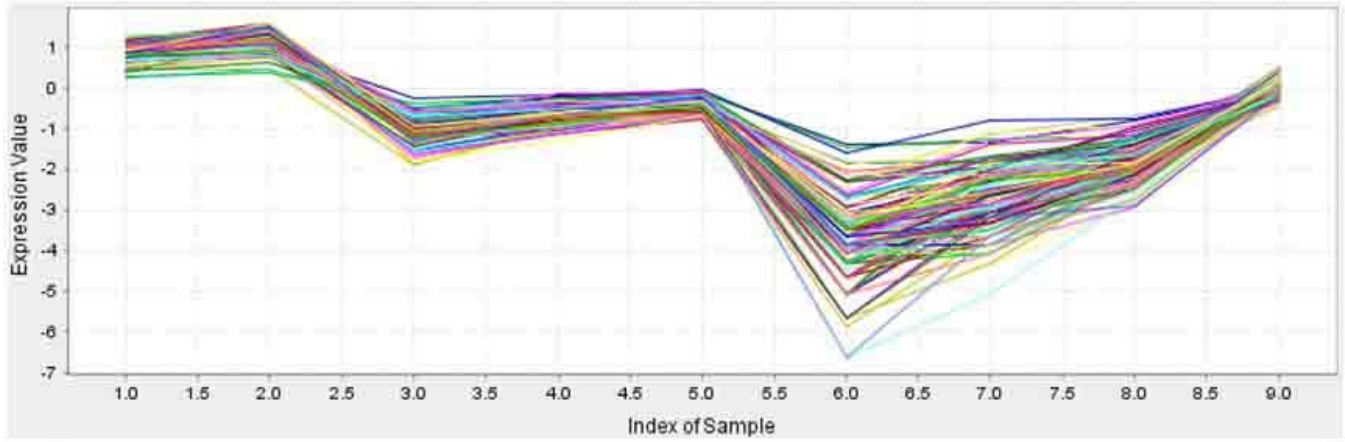
Rank / Gene	Rank1	Rank2	Rank3	Rank4
Gene1	cond2	cond4	cond3	cond1
Gene2	cond1	cond3	cond4	cond2
Gene3	cond1	cond2	cond4	cond3
Gene4	cond1	cond2	cond3	cond4
Gene5	cond2	cond4	cond1	cond3

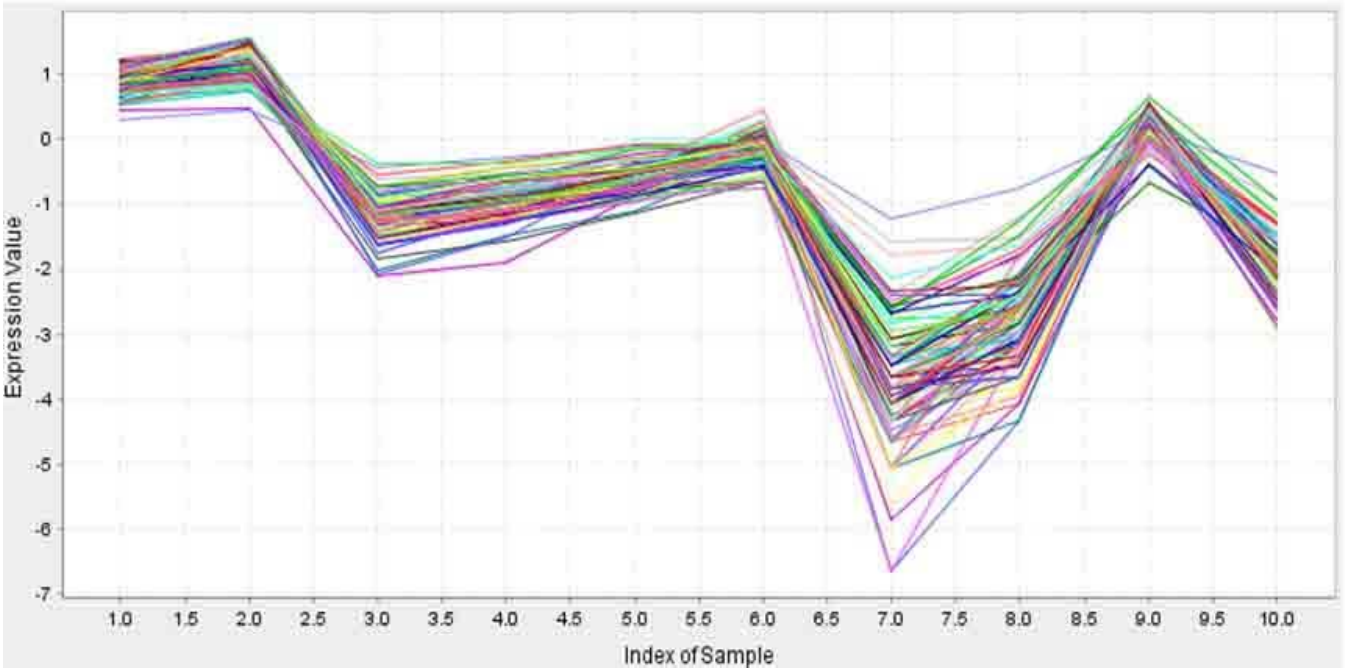
Parameter	DESCRIPTION	QUANTITY
MaxLen	Maximum condition length	n
PopSize	Number of individuals in a population	300
MaxGen	Maximum generation number	300
Pc	Crossover rate	0.6
Pm	Mutation rate	0.01
Min	Parameter for Rank based selection	0 to 0.9
Max	Parameter for Rank based selection	1.1 to 2.0
MaxIter	Maximum pair-iteration count	15 to 20

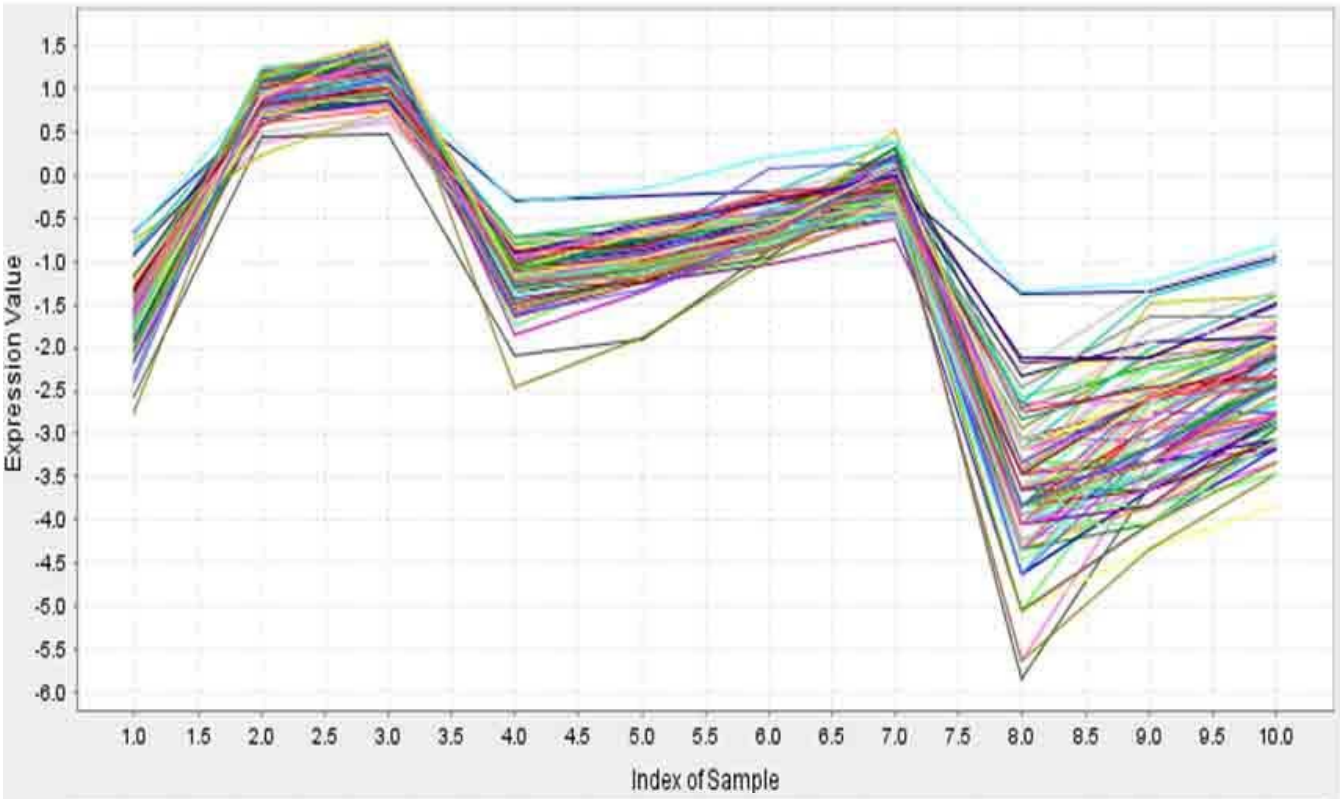


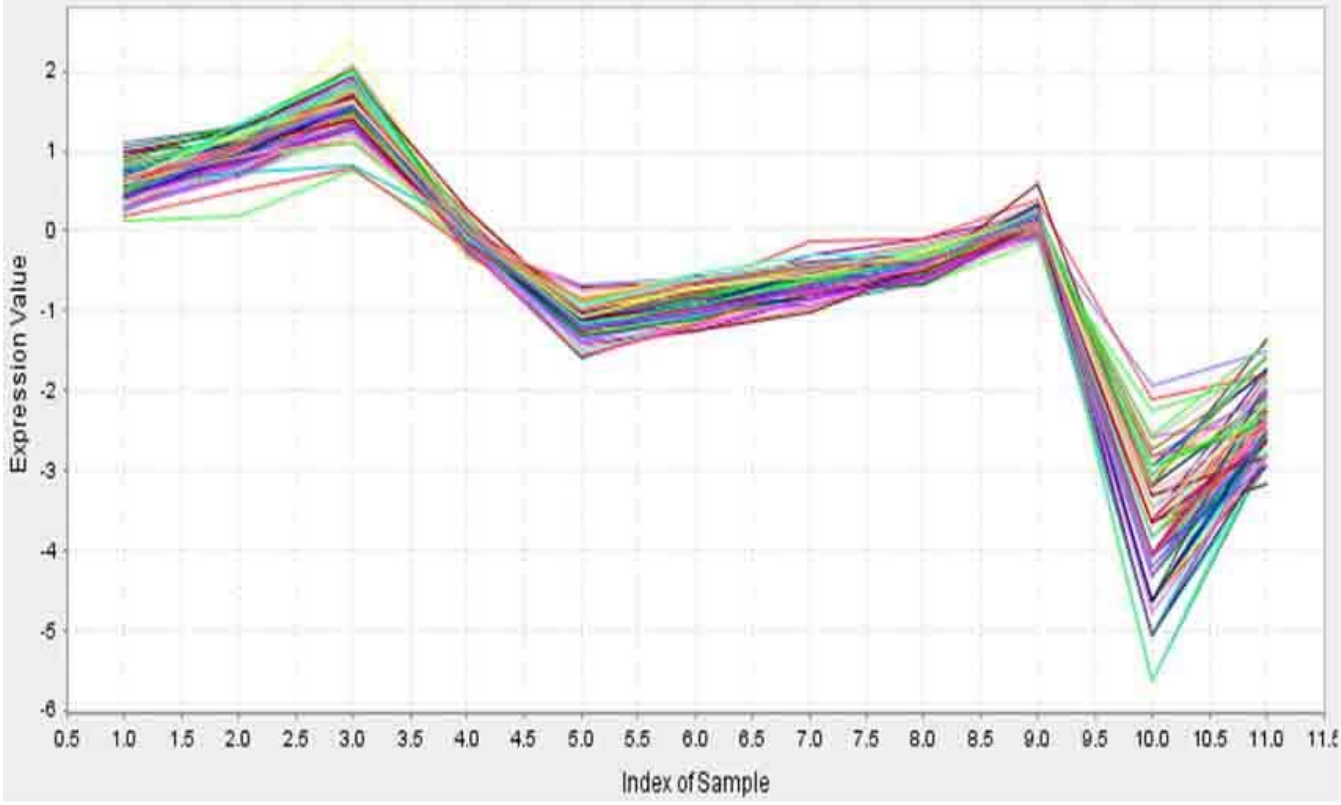

Biological relevance of biclusters
Proportion of biclusters
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
-150
-140
-130
-120
-110
-100
-90
-80
-70
-60
-50
-40
-30
-20
-10
-5
0
ECOPSM
OPSM
SEBI

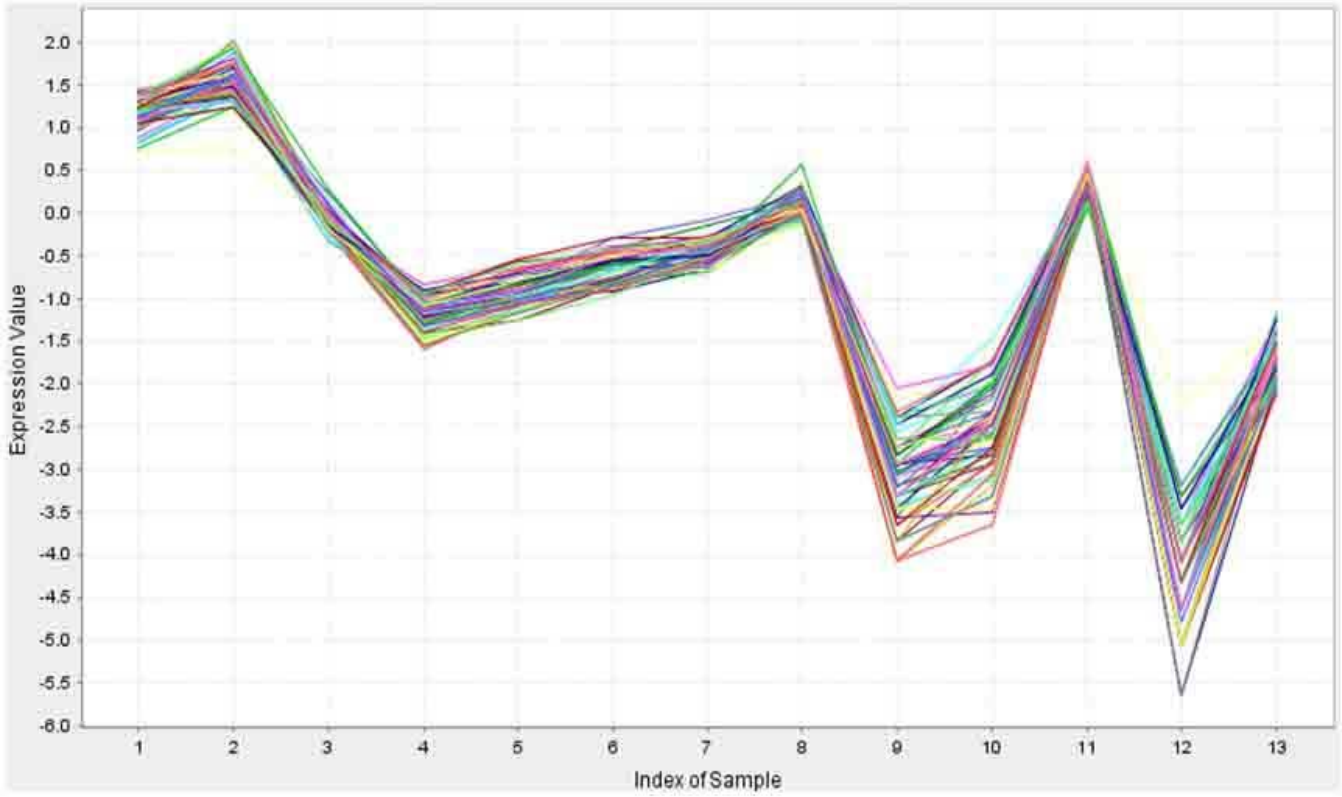

P-value of overrepresented GO category on a logarithmic scale
Bicluster size
The number of genes
350
300
250
200
150
100
50
0
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
26
27
28
33
34
The number of conditions
ECOPSM
OPSM200
OPSM400
OPSM600
OPSM800
OPSM1000

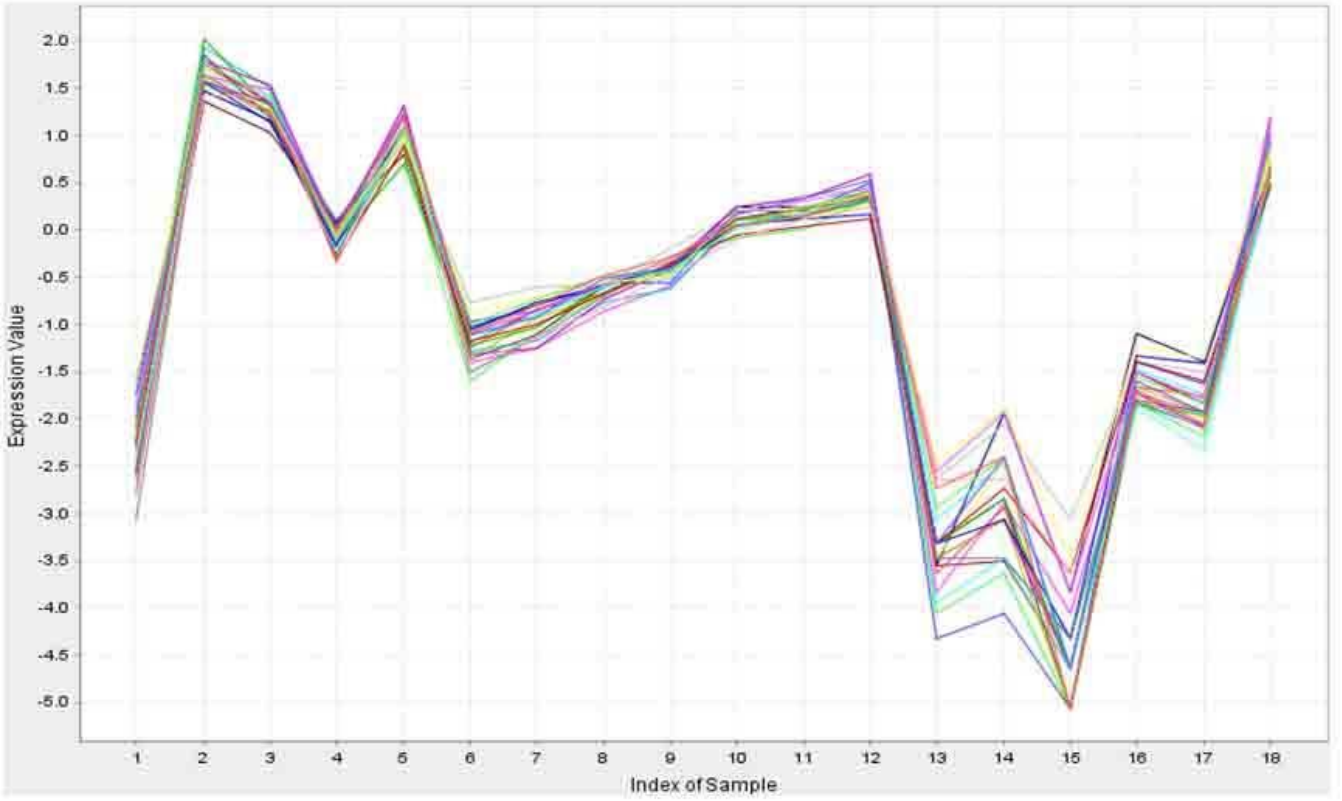

Expression Value
1
0
-1
-2
-3
-4
-5
-6
-7
1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0
Index of Sample

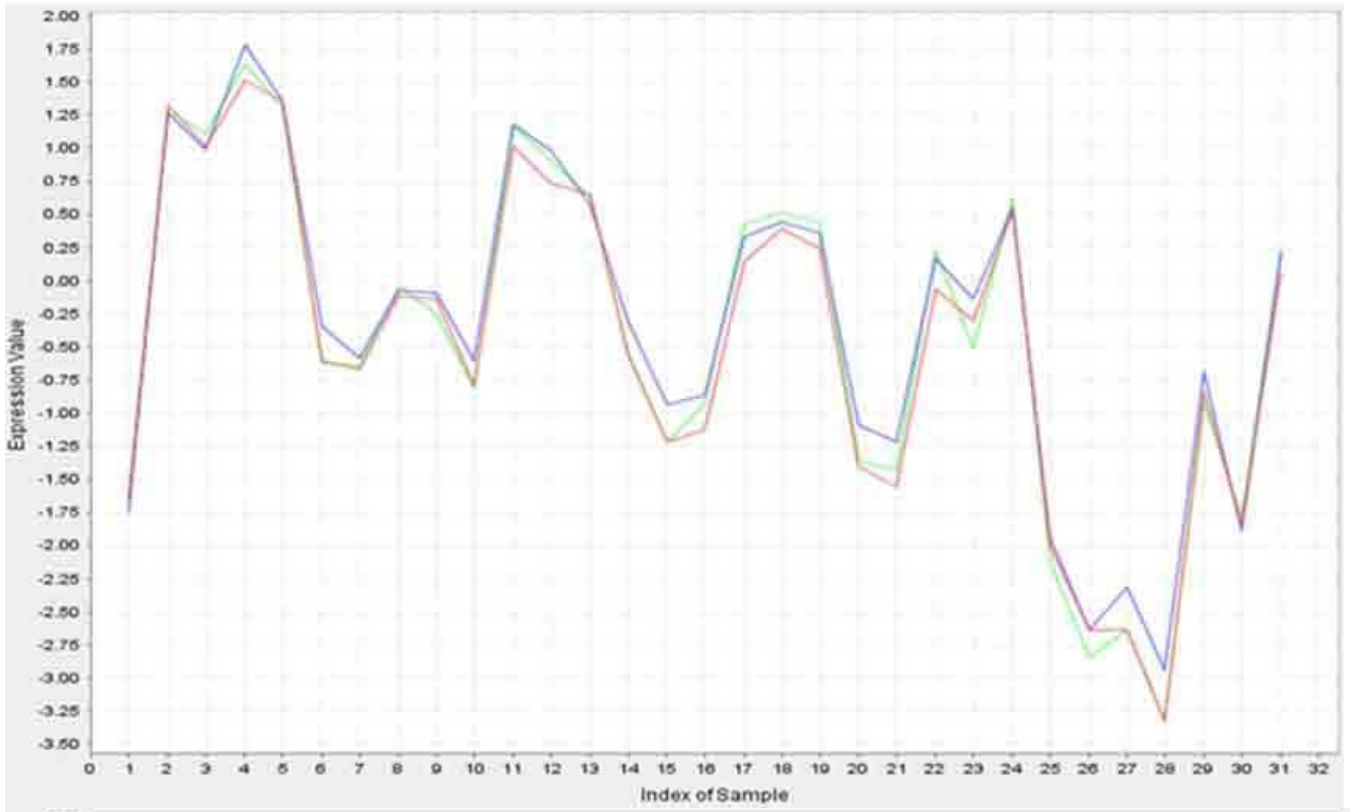

Expression Value
1
0
-1
-2
-3
-4
-5
-6
-7
1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0 9.5 10.0
Index of Sample

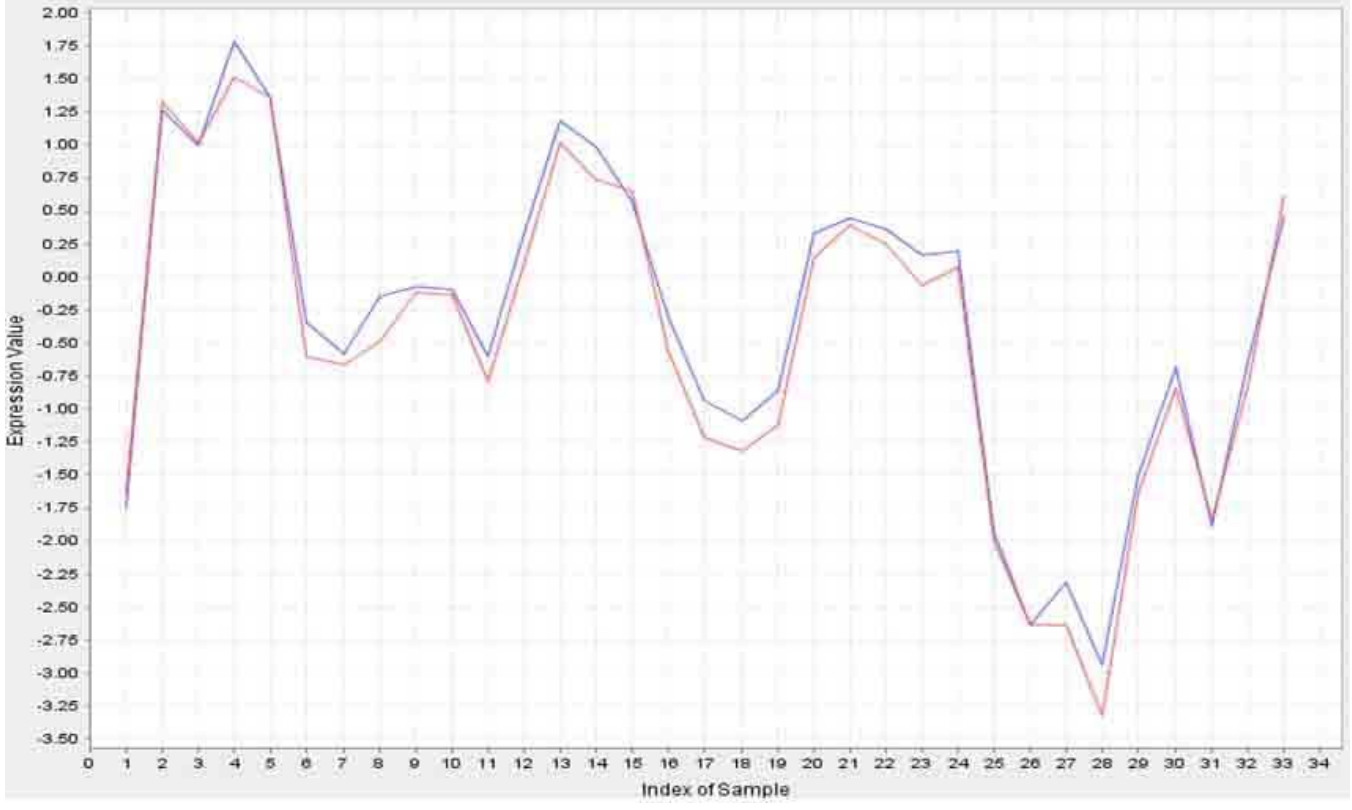

Expression Value
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
-2.0
-2.5
-3.0
-3.5
-4.0
-4.5
-5.0
-5.5
-6.0
1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0 9.5 10.0
Index of Sample


Expression Value
2
1
0
-1
-2
-3
-4
-5
-6
0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0 9.5 10.0 10.5 11.0
Index of Sample


Expression Value
2.0
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
-2.0
-2.5
-3.0
-3.5
-4.0
-4.5
-5.0
-5.5
-6.0
1
2
3
4
5
6
7
8
9
10
11
12
13
Index of Sample


Expression Value
2.0
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
-2.0
-2.5
-3.0
-3.5
-4.0
-4.5
-5.0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
Index of Sample


Expression Value
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
-0.25
-0.50
-0.75
-1.00
-1.25
-1.50
-1.75
-2.00
-2.25
-2.50
-2.75
-3.00
-3.25
-3.50
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
Index of Sample


Expression Value
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
-0.25
-0.50
-0.75
-1.00
-1.25
-1.50
-1.75
-2.00
-2.25
-2.50
-2.75
-3.00
-3.25
-3.50
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
Index of Sample