Revisiting the Yeast Cell Cycle Problem with the Improved TriGen Algorithm

D. Gutiérrez-Avilés
Dpto. Leng. y Sistemas. Informáticos
Universidad de Sevilla
Sevilla, España
davgutavi@alum.us.es

C. Rubio-Escudero
Dpto. Leng. y Sist. Informáticos
Universidad de Sevilla
Sevilla, España
crubioescudero@us.es

J. C. Riquelme
Dpto. Leng. y Sist. Informáticos
Universidad de Sevilla
Sevilla, España
riquelme@us.es

***Abstract*—Analyzing microarray data represents a computational challenge due to the characteristics of these data. Clustering techniques are widely applied to create groups of genes that exhibit a similar behavior under the conditions tested. Biclustering emerges as an improvement of classical clustering since it relaxes the constraints for grouping allowing genes to be evaluated only under a subset of the conditions and not under all of them. However, this technique is not appropriate for the analysis of temporal microarray data in which the genes are evaluated under certain conditions at several time points. On a previous work we presented the TriGen algorithm, a genetic algorithm that finds triclusters of gene expression that take into account the experimental conditions and the time points simultaneously, and was applied to the yeast (Saccharomyces Cerevisiae) cell cycle problem. In this article we present some improvements on the genetic algorithm and we also present the results of applying the improved TriGen algorithm to the yeast cell cycle problem, where the goal is to identify all genes whose expression levels are regulated by the cell cycle.**

***Keywords*-microarrays, temporary data, genetic algorithms, yeast**

I. Introduction

The use of high throughput processing techniques has revolutionized the technological research and has exponentially increased the amount of data available [1]. Particularly, microarrays have revolutionized biological research by its ability to monitor changes in RNA concentration in thousands of genes simultaneously [2]. A common practice when analyzing gene expression data is to apply clustering techniques, creating groups of genes that exhibit similar expression patterns. These clusters are interesting because it is considered that genes with similar behavior patterns can be involved in similar regulatory processes [3]. Although in theory there is a big step from correlation to functional similarity of genes, several articles indicate that this relation exists [4]. Traditional clustering algorithms work on the whole space of data dimensions examining each gene in the dataset under all conditions tested. Biclustering techniques [5] go a step further by relaxing the conditions and by allowing assessment only under a subset of the conditions of the experiment, and it has proved to be successful finding gene patterns [6], [7]. However, clustering and biclustering are insufficient when analyzing data from microarray experiments where attention is payed on how the time affects gene's behavior. There is a lot of interest in this type of time series experiment because they allow an in-depth analysis of molecular processes in which the time evolution is important, for example, cell cycles, development at the molecular level or evolution of diseases [8]. Therefore, the use of specific tools for data analysis in which genes are evaluated under certain conditions considering the time factor becomes necessary. The TriGen algorithm, presented in [9], goes a step further than clustering and biclustering techniques in the creation of groups of pattern similarity for genes. TriGen works on a three-dimensional space, thus taking into account the time factor, and allowing the evaluation of the behavior of genes only under certain conditions and only under certain time points. TriGen applies an evolutionary technique, genetic algorithms, to find solutions that we refer to as triclusters. In this article we present some changes made to the TriGen algorithm, in particular to the initial population creation method and the organization of the input data to avoid overlapping on the solutions. The algorithm is applied to the yeast (Saccharomyces Cerevisiae) cell cycle problem and the results are compared to the ones obtained in [9]. Other works related with this approach are in [10] and [11]. The rest of the paper is structured as follows. Section II describes the TriGen algorithm in detail, Section III shows the results with synthetic data and with the yeast data. Section IV summarizes the conclusions reached and proposals for future work.

II. Methodology

We describe the implementation of the TriGen algorithm. In this section we explain the inputs and outputs of the algorithm and we provide a detailed description of the evolutionary process and all the operators implied.

A. Input data

The input data is obtained from temporal microarray experiments. Each of these microarrays reveals the expression level under specific experimental conditions and at an instant of time. Therefore, the input data consists of T number of microarrays, as many as time points to be analyzed.

```
Input:  Temporary microarray data
Output:  Tricluster Solution Set

Begin TriGen algorithm
  Repeat for each Tricluster solution
    Generate Initial Population
    Evaluate population
    Repeat for Number of Generations
      Select Population
      Cross Population
      Mutate Population
      Evaluate Population
    End Repeat
    Select Best One
    Include Best One in Solution Set
    Data Hierarchy Update
  End Repeat
End TRIGEN algorithm
```

Figure 1: TriGen algorithm

Each value of a microarray for an specific time t represents the level of gene expression of a gene g under a specific experimental condition c.

B. Definition of Tricluster

We define a tricluster as a subset of time instants T, a subset of genes G and a subset of conditions C extracted from the input data. In this particular work, each tricluster contains the expression values of the these three sets and a fitness value that indicates the tricluster's quality. The fitness function will be described in detail in Section II-C4 (Evaluation operator). Qualitatively, a tricluster will provide information on behavior pattern of a subset of genes under certain conditions and at certain time points.

C. TriGen Algorithm Description

TriGen is based on a genetic algorithm. The evolutionary process has an initialization method in which the initial population will be created using a hierarchy data method to avoid overlapping among solutions and several operators: evaluation, which measures the quality of each chromosome or individual of the population, selection, which serves to decide which individuals will survive to the next generation, crossover, which creates the necessary connections between pairs of individuals to share new genetic material and finally mutation, which performs punctual changes to individuals to ensure genetic variability of future generations, i.e., exploring new spaces of solutions (See Figure 1).
We discuss in detail each of these methods and operators.

1) Codification of Individuals: Each member of the population represents a tricluster which is a potential solution. It has genetic material that will be manipulated by the genetic operators described in "Genetic Operators" below. This genetic material is composed by a set of chromosomes, they are a subset of time instants T, a subset of genes G and a subset of conditions C extracted from the input data. Each of them is composed by a number of genes, they are the components of the tricluster (they correspond to the components of the input data).

2) Data Hierarchy: To avoid overlapping solutions (triclusters) and cover the widest space possible in the input data we maintain a hierarchy for the input data. Time points, genes and conditions coordinates are organized by the number of occurrences (levels) in the previous triclusters found. Thus, each coordinate will be assigned an initial level of 0 and, when a tricluster is found, each of its coordinates will be increased by 1. The hierarchy data is a support tool for the initialization method and it helps us to cover the space of the input data efficiently with triclusters.

3) Generation of Initial Population: The population is randomly created following some considerations. The method receives two parameters: the number of individuals and a parameter indicating how many of them will be randomly chosen. For the latter, we randomly choose a subset of times, genes and conditions of the input data. This process is repeated as many times as specified by the second input parameter described above. For the rest of individuals, we use the data hierarchy so that we choose the lowest level coordinates to create the individuals.

4) Genetic Operators:

Selection: Groups of individuals are randomly created sorted from lowest to highest according to the fitness function, and then a random selection from the three groups of the individuals required according to an input parameter is made.

Crossover: This operator completes the population in the next generation P_t generating two new individuals (children) combining the genetic material from two existing ones (parents). For each point of the two parents get two children so the number of crossings is determined by number of individuals who are required to complete the population. This is a one-point cross that determine a random point cross for the times, genes and conditions and mixing each of the parts to obtain two child by crossing.

Mutation: This operator selects, based on a mutation probability input parameter, a number of individuals who suffer a random out of six: add a time component, a gene component or a condition component , or remove a time component, gene component or condition component.

Evaluation: Since triclustering emerges as an improvement of biclustering to analyze microarray data taking into account the temporal dimension, we have adapted the classical biclustering fitness function, Mean Squared Residue

(MSR), presented by Cheng and Church in [12], to the three dimensional space. MSR compares the similarity of each value in the bicluster to the mean values of all genes under the same condition, the mean of the gene under the other conditions included in the bicluster, and the mean of all values in the bicluster. In the case of triclustering, we will assess the similarity of each value not only related to genes and conditions, but also including the temporary plane, i.e., we asses how a gene g behaves under all conditions C at the time points T, how a condition c affects all genes G in time T, and the time factor t in relation to genes G and conditions C, as well as the mean value of all the tricluster. This is formalized as follows:

$$r_{GCT} = \frac{\sum_{g\epsilon G, c\epsilon C, t\epsilon T} r_{gct}^2}{|G| * |C| * |T|} - Weights - Distinction$$

in the first member of equation, the numerator is:

$$\begin{aligned} r_{gct} &= V_{gct} + M_{GC}(t) + M_{GT}(c) + M_{CT}(g) - \\ &\quad M_G(c,t) - M_C(g,t) - M_T(g,c) - M_{GCT} \end{aligned}$$

where V_{gct} is the tricluster value being evaluated, $M_{GC}(t)$ is the mean of the genes under conditions at a point in time t, $M_{GT}(c)$ is the mean of the genes over time under a condition c, $M_{CT}(g)$ is the mean of a gene g in time under the conditions, $M_G(c,t)$ is the mean of the genes under one condition and a time point, $M_C(g,t)$ is the mean of the values of a gene at a time point under conditions, $M_T(g,c)$ is the mean of a gene under a condition at all time points and M_{GCT} is the mean value of all points of tricluster.
The denominator factor is:

$$|G| * |C| * |T|$$

where $|G|$, $|C|$ and $|T|$ are, respectively, the number of genes, times and conditions in the tricluster under evaluation. The second member of equation, $Weights$, corresponds to:

$$Weights = |G| * w_g + |C| * w_c + |T| * w_t$$

where w_g, w_c and w_t are the weights of the genes, conditions and times for the solution tricluster respectively and $|G|$, $|C|$ and $|T|$ correspond again to the number of genes, times and conditions in the tricluster under evaluation. When increasing the value of one of these weights, we favor the TriGen algorithm finding triclusters with a greater number of components on that term.
The $Distinction$ member can be explained as:

$$Distinction = \frac{CDN_g}{|G|} * wd_g + \frac{CDN_c}{|C|} * wd_c + \frac{CDN_t}{|T|} * wd_t$$

where CDN_g (Coordinate Distinction Number of g), CDN_c (Coordinate Distinction Number of c) and CDN_t (Coordinate Distinction Number of t) are, respectively, the number of genes, conditions and time coordinates in the tricluster solutions that do not contain the tricluster being evaluate, wd_g, wd_c and wd_t are the distinction weights of the genes, conditions and times respectively and $|G|$, $|C|$ and $|T|$ correspond again to the number of genes, times and conditions in the tricluster under evaluation. $Distinction$ measures how different the individual under evaluation is compared to the tricluster previously found. If the the value of any of the weights is increased, we favor the finding of triclusters with non-overlapping gene, condition or time coordinates regarding the triclusters previously found.

III. Results

We show the results obtained applying the TriGen algorithm both to real and synthetic data. Synthetic data are widely used not only for testing the performance of microarray analyzing techniques [13] but also in more general data mining publications [14]. All experiments were done on a Mac Book Pro machine (Intel Core 2 Duo, 2.53 GHz, 4 GB memory) over Mac OS Snow Leopard. TriGen algorithm are developed with Eclipse Helios IDE on Java language. The computational cost of an experiment with 200 generations, 100 individuals in the initial population and 10 solutions (these are the parameters used in the experiments presented in this paper) is 27 seconds.

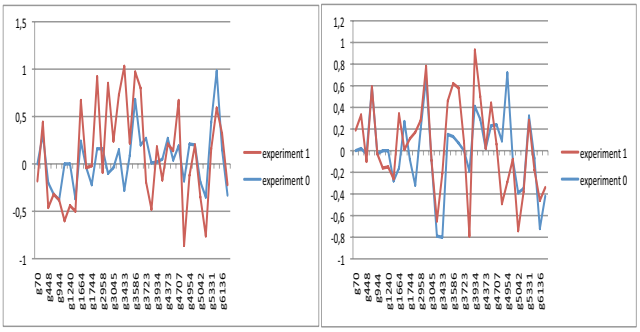
A. Results using Synthetic Data

The set of synthetic data has been generated using a software application developed for such purpose. For this particular work, we have simulated data from 5 different time points and 10 conditions using microarrays containing 1000 genes. Each gene is assigned a random value which is contained in the rank, respectively for each condition, $[1, 15]$, $[7, 35]$, $[60, 75]$, $[0, 25)$, $[30, 100]$, $[71, 135]$, $[160, 375]$, $[5, 30]$, $[25, 40]$ y $[10, 30]$. In such data set, we have allocated a tricluster with all its values fixed to 1. The size of the tricluster is $time = 5$, $genes = 8$ and $conditions = 8$. TriGen was able to successfully find a solution containing the aforementioned tricluster. The execution was made with the following parameters: 100 generations and 500 members in the population. The selection parameter is 70% and the mutation probability is 5%. The weight values have been adjusted to $w_g = 0.01$, $w_c = 0.55$ y $w_t = 0.35$, in order to favor the number of conditions and time points, since the genes show high dimensionality in relation to conditions and time.

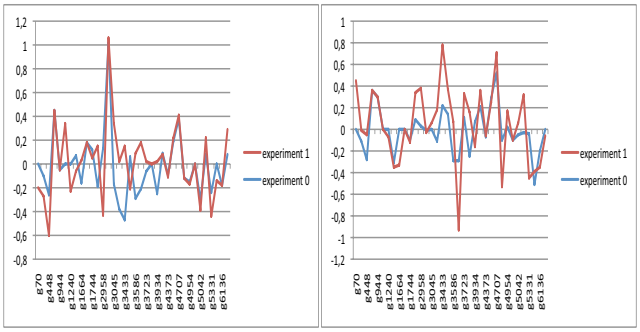
B. Results using Real Data

We have applied the TriGen algorithm to the yeast (Saccharomyces Cerevisiae) cell cycle problem [15] as we did in [9]. The yeast cell cycle analysis project's goal is to identify all genes whose mRNA levels are regulated by the cell cycle. By applying TriGen to this dataset, we aim to find a pattern on this cell cycle.

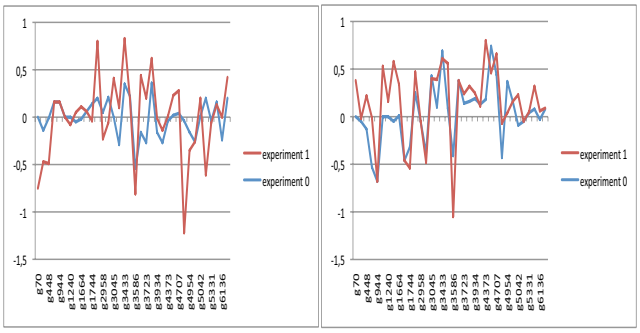
The data is available in http://genome-www.stanford.edu/cellcycle/. In this experiment, 6179



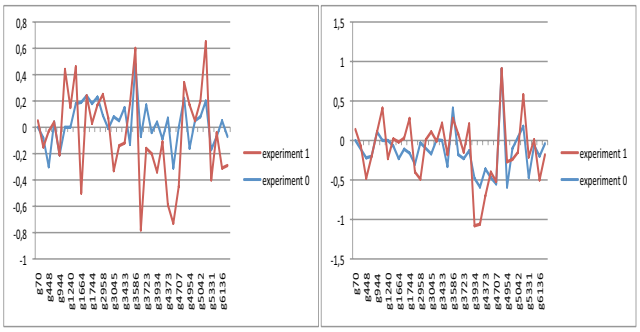
(a) time point 0 (b) time point 2



(c) time point 3 (d) time point 4

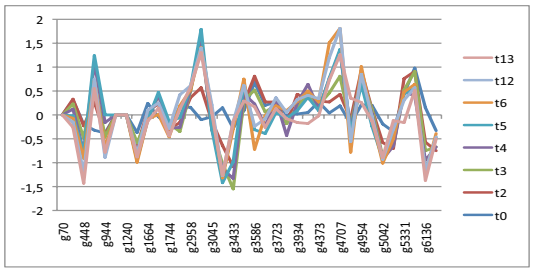


(e) time point 5 (f) time point 6

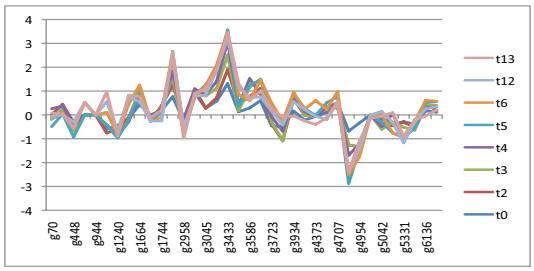


(g) time point 12 (h) time point 13

Figure 2: Gene expression values under two experiments at time point 0 (a), 2 (b), 3 (c), 4 (d), 5 (e), 6 (f), 12 (g) and 13 (h).

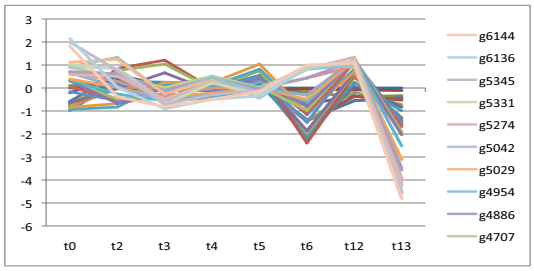


(a) pheromone

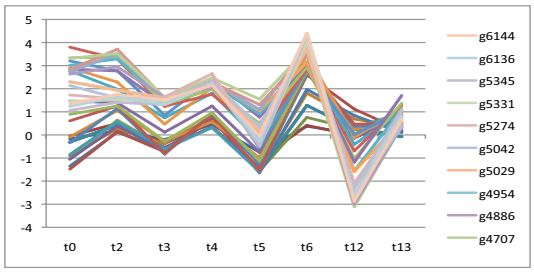


(b) cdc15

Figure 3: Gene expression values under eight time points at pheromone (a) and cdc15 (b) experiments.



(a) pheromone



(b) cdc15

Figure 4: Gene expression for eight time points under gene solution set at pheromone (a) and cdc15 (b) experiments.

genes are analyzed under 6 conditions, termed cln3, clb2, pheromone, cdc15, cdc28 and elutriation [15]. Samples were taken at 2 time points for cln3, 2 for clb2, 18 for pheromone, 24 for cdc15, 17 for cdc28 and 14 for elutriation. As we did in [9], to apply the TriGen algorithm we did not take into account the conditions with only 2 time points, since

they are not relevant for a time course experiment, so we used the first 14th time points of the pheromone, cdc15, cdc28 and elutriation experiment. Therefore our dataset contains 14 time points, 6179 genes and 4 conditions. The algorithm has been executed to extract 20 solutions, i.e. 20 triclusters with the following parameters: 100 generations, 200 members in the population, 80% for selection probability and 50% for mutation. The weights applied have been $w_g = 0.0$, $w_c = 0.2$ y $w_t = 0.8$, thus we favored the condition dimension and penalized gene dimension to get a reduced subset of genes on solution triclusters. The distinction weights have been $wd_g = 1.0$, $wd_c = 0.0$ y $wd_t = 0.0$, thus we favor the distinction ratio of the gene dimension, therefore solution triclusters cover as much as possible the input data space on gene dimension since it is the largest one.

For legibility reasons we focus in one of the solutions, a tricluster gathering 36 genes under 2 experiments, pheromone (experiment 0) and cdc15 (experiment 1) and 8 time points, instants 0, 2, 3, 4, 5, 6, 12 and 13.

We show three groups of graphics related to this solution: In Figure 2 we present the outline of the gene expression values (Y axis) for each solution gene point (X axis) comparing the pheromone (experiment 0) and cdc15 (experiment 1), experiments setting time points to instants 0 (a), 2 (b), 3 (c), 4 (d), 5 (e), 6 (f), 12 (g) and 13 (h). In Figure 3 we present the outline of gene expression values (Y axis) for each solution gene point (X axis) comparing 0, 2, 3, 4, 5, 6, 12 and 13 time points setting the experiments to pheromone (a) and cdc15 (b). Finally in Figure 4 we present the outline of gene expression values (Y axis) for each time point (X axis) comparing each solution gene setting the experiments to pheromone (a) and cdc15 (b).

In view of the results, we can say that in the levels of expression of the genes for the pheromone and cdc15 experiments share a common behaviour at the indicated time points. Regarding the results obtained in [9], we can see that the changes on the TriGen algorithm have improved the quality of the solutions in terms of pattern finding and solution distribution over the input data. Generally, we see that the algorithm has been capable to group together genes with very similar gene expression values for the three dimensions visited. Therefore, TriGen has shown its ability to mine groups of co-expressed genes taking into account the time dimension.

IV. Conclusions and Future Work

We have presented the results obtained by applying the tricluster algorithm TriGen to the yeast cell cycle problem. TriGen represents an step further than clustering and biclustering in the analysis of temporal microarray data since it groups genes which exhibit a similar behavior under a subset of conditions and under a subset of time points. It is genetic based algorithm, with an evaluation function developed as the natural 3D extension from the classic function evaluation for biclustering proposed by Cheng y Church in [12]. The results show that the algorithm is capable to mine triclusters of genes based on their expression levels. We have improved, as we have seen, the pattern finding capability of the algorithm but it is still in an early development stage, so there is still a lot of work to do, not only for the algorithm, such as a deeper study of the evaluation function or parallelization of the algorithm to make it faster, but also for the validation phase or the application of this algorithm for other types of data, such as image analysis. Analyze the biological value of these patterns and its possible applications is the next step in our work and also included in the future work.

References

[1] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[2] P. Brown and D. Botstein, "Exploring the new world of the genome with dna microarrays," *Nature Genet.*, vol. 21, no. Suppl., pp. 33–37, 1999.

[3] M. Tan, E. Smith, J. Broach, and C. Floudas, "Microarray data mining: A novel optimization-based approach to uncover biologically coherent structures," *BMC Bioinformatics*, vol. in press.

[4] P. D'haeseleer, S. Liang, and R. Somogyi, "Genetic network inference: from co-expression clustering to reverse engineering," *Bioinformatics*, vol. 16, no. 8, pp. 707–726, 2000.

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

[6] M. Eisen, P. Spellman, P. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," *Proceedings of the National Academy of Sciences*, vol. 95, no. 25, p. 14863, 1998.

[7] B. Pontes, F. Divina, R. Giráldez, and J. Aguilar-Ruiz, "Improved biclustering on expression data through overlapping control," *International Journal of Intelligent Computing and Cybernetics*, vol. 3, no. 2, pp. 293–309, 2010.

[8] Z. Bar-Joseph, "Analyzing time series gene expression data," *Bioinformatics*, vol. 20, no. 16, p. 2493, 2004.

[9] D. Gutiérrez-Avilés, C. Rubio-Escudero, and J. C. Riquelme, "Unravelling the yeast cell cycle using the trigen algorithm," *LNAI (accepted)*, 2011.

[10] L. Zhao and M. J. Zaki, "tricluster: An effective algorithm for mining coherent clusters in 3d microarray data," *In Proc. of the 2005 ACM SIGMOD international conference on Management of data*, pp. 694–705, 2005.

[11] P. Mahanta, H. Ahmed, D. Bhattacharyya, and J. Kalita, "Triclustering in gene expression data analysis: A selected survey," *Emerging Trends and Applications in Computer Science (NCETACS)*, pp. 1–6, 2011.

[12] Y. Cheng and G. Church, "Biclustering of expression data." in *Proceedings/... International Conference on Intelligent Systems for Molecular Biology; ISMB. International Conference on Intelligent Systems for Molecular Biology*, vol. 8, 2000, p. 93.

[13] K. Hakamada, M. Okamoto, and T. Hanai, "Novel technique for preprocessing high dimensional time-course data from dna microarray: mathematical model-based clustering," *Bioinformatics*, vol. 22, no. 7, p. 843, 2006.

[14] R. Pargas, M. Harrold, and R. Peck, "Test-data generation using genetic algorithms," *Software Testing Verification and Reliability*, vol. 9, no. 4, pp. 263–282, 1999.

[15] P. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher, "Comprehensive identification of cell cycleregulated genes of the yeast saccharomyces cerevisiae by microarray hybridization," *Molecular Biology of the Cell*, vol. 9, no. 3, pp. 3273–3297, 1998.