TriGen: A genetic algorithm to mine triclusters in temporal gene expression data

D. Gutiérrez-Avilés, C. Rubio-Escudero, F. Martínez-Álvarez, J.C. Riquelme

Keywords:
Tricluster
Genetic algorithms
Microarray data
Time series

A B S T R A C T

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 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 longitudinal experiments in which the genes are evaluated under certain conditions at several time points. We present the *TriGen* algorithm, a genetic algorithm that finds triclusters of gene expression that take into account the experimental conditions and the time points simultaneously. We have used *TriGen* to mine datasets related to synthetic data, yeast (*Saccharomyces cerevisiae*) cell cycle and human inflammation and host response to injury experiments. *TriGen* has proved to be capable of extracting groups of genes with similar patterns in subsets of conditions and times, and these groups have shown to be related in terms of their functional annotations extracted from the Gene Ontology.

1. Introduction

The use of high throughput processing techniques has revolutionized the technological research and has exponentially increased the amount of data available [11]. Particularly, microarrays revolutionized biological research by its ability to monitor changes in RNA concentration in thousands of genes simultaneously [7].

A common practice when analyzing gene expression data is to apply clustering techniques, creating groups of genes that exhibit similar expression patterns [31]. These clusters are interesting because it is considered that genes with similar behavior patterns can be involved in similar regulatory processes [34]. Although in theory there is a big step from correlation to functional similarity of genes, several articles indicate that this relation exists [10]. Traditional clustering algorithms work on the whole space of data dimensions examining each gene in the dataset under all conditions tested.

Applying clustering algorithms on gene expression data usually does not provide the best results. Much of the activity patterns of gene groups are only present under a particular set of experimental conditions. Actually, the available knowledge on cell processes suggests that, while a subset of genes is co-regulated and co-expressed under particular experimental conditions, under different conditions these genes can show independent behavior. Discovering this local behavior patterns can be the key to discover gene *pathways*, which could be hard to discover in other ways. For this reason, the paradigm of clustering techniques must change to methods that allow local pattern discovery in gene expression data [6].

Biclustering addresses this problem [16] 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 in finding gene patterns [21]. However, clustering and biclustering are insufficient when analyzing data from microarray experiments where attention is drawn on how time affects gene's behavior [13]. There is a lot of interest in this type of longitudinal experiments 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 [4]. Therefore, the use of specific tools for data analysis in which genes are evaluated under certain conditions considering the time factor becomes necessary. In this sense, triclustering appears as a valuable tool since it allows for the assessment of genes under a subset of the conditions of the experiment and under a subset of time points.

Biclustering is known to be a NP hard problem [35] and therefore many proposed successful biclustering algorithms are based on heuristics [3,25]. Since the computational complexity of triclustering is greater than the biclustering one, heuristic based algorithms are a good approximation for triclustering.

In this context, triclustering algorithms appear as a way to find genes of similar expression profiles along a segment of time series in a subset of conditions. We define a coherent tricluster as a set of genes which exhibits either similar numeric values for the times and conditions (coherent values [21]) or similar behaviors regardless of the exact numeric values: correlated positive and negative changes in the expression values (coherent behavior [21]). Both types of coherent clusters may contain information useful to identify useful phenotypes, potential genes related to these phenotypes and their regulation relations [36].

We present the *TriGen* (Triclustering-Genetic based) algorithm based on an evolutionary heuristic, genetic algorithms, which finds groups of pattern similarity for genes on a three dimensional space, thus taking into account the gene, conditions and time factor. Although many clustering and biclustering models define similarity based on distance functions [12,37], these functions are not always adequate to capture similarities among genes, since correlations may still exist among a set of genes which are expressed at different levels of magnitude. Therefore we propose two different evaluation functions: the first one finds triclusters of coherent values and is based on a three dimensions adaptation of the Mean Square Residue measure (MSR) which is a classic biclustering distance measure for gene expression analysis [9], the second one is a correlation measure that identifies triclusters of coherent behavior based on the least square approximation (LSL) which calculates the distances among the slopes of the least square lines from a tricluster.

We show that the results obtained from applying the *TriGen* algorithm to a synthetic dataset and two real ones: the yeast cell-cycle regulated genes [33] and the human immune response to inflammation and host response to injury [8] both for coherent values and behaviors of the genes (MSR and LSL evaluation functions respectively). An early version of *TriGen* was presented in [14]. The algorithm has been improved by adding a new correlated based evaluation function f_{LSL}. It has been re-executed on the already used datasets, providing new improved results, and a new dataset related to human inflammation and host response to injury has been used. A validation step has been added to the methodology, based on the Gene Ontology project, providing functional annotations for the genes.

The rest of the paper is structured as follows. A review of the latest related works can be found in Section 2. Section 3 describes the methodology of the *TriGen algorithm*, including a detailed description of the genetic operators used and the two fitness functions. In Section 4, we show the results of applying *TriGen* to the synthetic dataset, the yeast cell cycle and the inflammation and the host response to injury problems. Section 5 shows the conclusions.

2. Related works

This section is to provide a general overview of recent works in the field of gene expression time series data. In particular, those focused on the application of triclustering. Despite the vast variety of (bi)clustering methods existing, there is no one apparently capable of extracting meaningful information from a wide range of gene expression temporal data. On the contrary, the use of triclustering seems to be the most suitable strategy due to its ability to deal with the third dimension: The time.

In 2005, Zhao and Zaki [41] introduced the triCluster algorithm to extract patterns in 3D gene expression data. They also presented a set of metrics to assess its quality and tested the approach on real microarrays. An extended and generalized version (g-triCluster) was published one year later [18]. The generalization claimed by the authors was based on the discovery of more coherent triclusters and on its robustness to noise.

Although this algorithm was able to successfully discover such patterns, the authors in [1] addressed the existing problem associated with its NP-completeness. To overcome this drawback, they proposed a parallelized version using the filter-labeled-stream paradigm, exhibiting great improvement in terms of computational cost.

A new definition for coherent triclusters focused on finding regulatory relationships among genes can be found in [40]. The algorithm was applied on both synthetic and real data. Later on [38], the authors proposed an enhanced model to retrieve time-delayed clusters. The novelty lied on the discovery of gene expression cycle time, essential task to create gene regulatory networks.

Another approach, LagMiner, was introduced in [39] to find time-lagged 3D clusters. The highlighting feature claimed by the authors was, this time, its ability to discover triclusters satisfying the constraints of coherence, regulation, minimum gene number, sample subspace size and time period length. As for previous works, it was evaluated on both synthetic and real-life datasets.

The evolutionary computation has also been used in the search for triclusters. Particularly, a multi-objective algorithm that simultaneously optimizes several conflicting requirements was presented in [19]. Again, the algorithm was tested on real datasets.

A new strategy to mine 3D-clusters in real-valued data was introduced in [32]. In particular, the authors were concerned about discovering subspaces with a significant number of items, one of the main drawbacks typically found in tricluster-based approaches. The authors applied its methodology to a large number of synthetic datasets and, additionally, to a real-world case study. At the same conference, another approach focused on a triclustering method to mine quantitative data was presented [17]. However, the main concern of the authors was, this time, to discover triclusters with low variance. Their approach was successfully tested on a synthetic dataset and on a cross-species genomic dataset, yielding new insights on this topic.

The work in [20] introduced a tricluster-based approach to discover temporal dependency association rules in microarray datasets. The rules obtained are to represent regulated relations among genes.

Finally, a brief survey on triclustering applied to gene expression time series was published in 2011 [22]. It provides a good starting point for those researchers novel in this topic.

3. Methodology

In this section, we describe the implementation of the *TriGen* algorithm, our proposal to mine three dimension gene expression data. The evolutionary process is shown in detail: all the operators involved as well as the fitness functions used on the evaluation step.

A tricluster TC is as a subset from a dataset D which contains information related to the behavior of some genes G_D at a set of times T_D under a set of conditions C_D. The tricluster TC is formally defined as $TC = T \times C \times G$ where $T \subseteq T_D$, $C \subseteq C_D$ and $G \subseteq G_D$. Qualitatively, a tricluster TC will provide information on the pattern of behavior of a subset of genes under certain conditions and at certain time points.

TriGen is based on a genetic algorithm. This evolutionary process has several steps (see Fig. 1): an initialization step in which the initial population will be created taking into account overlapping with previously found triclusters; an evaluation step in which the quality of each individual will be measured; a selection step, which serves to decide which individuals will survive to the next generation;

```
Input:  Dataset D

Output:  Set of tricluster solutions

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 individual
        Add to the solution set
    End Repeat
 End Trigen algorithm
```

Fig. 1. *TriGen* algorithm.

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. We now describe each of these operators.

3.1. *Codification of individuals*

Each individual of the population represents a tricluster TC which is a potential solution. It contains genetic material that will be manipulated by the genetic operators described below. The genetic material is structured as follows. An individual is composed of three sequences of structures: one for the sequence of genes G from the input dataset D, one for the sequence of conditions C, and one sequence of time points T. These sequences are set up based on the input matrix, that is

$$G = \langle g_{i_1}, g_{i_2}, \ldots, g_{i_B} \rangle \tag{1}$$

where B is the number of genes listed in the input matrix, $i_j < i_{j+1}$ for all genes, and $1 < i_j < B$.

Analogously:

$$C = \langle c_{i_1}, c_{i_2}, \ldots, c_{i_L} \rangle \tag{2}$$

where L is the number of conditions listed in the input matrix, $i_j < i_{j+1}$ for all conditions, and $1 < i_j < L$.

Finally, T represents different time stamps or values of pairs gene-condition at different times:

$$T = \langle t_{i_1}, t_{i_2}, \ldots, t_{i_M} \rangle \tag{3}$$

where M is the number of samples measured over time, and $t_{i_1} < t_{i_2} < \cdots < t_{i_M}$.

The algorithm's population is made up of several individuals, as depicted in Fig. 2.

3.2. *Generation of initial population*

The initial population is randomly generated. Part of the individuals are purely random generated, this is, a random subset of genes, conditions and times $TC = T \times C \times G$ are assigned. The rest of the individuals are also randomly generated but attending to some considerations to promote visiting of non-explored areas and visiting the widest area possible of the dataset D. For this purpose, each time a new solution tricluster TC is generated, the

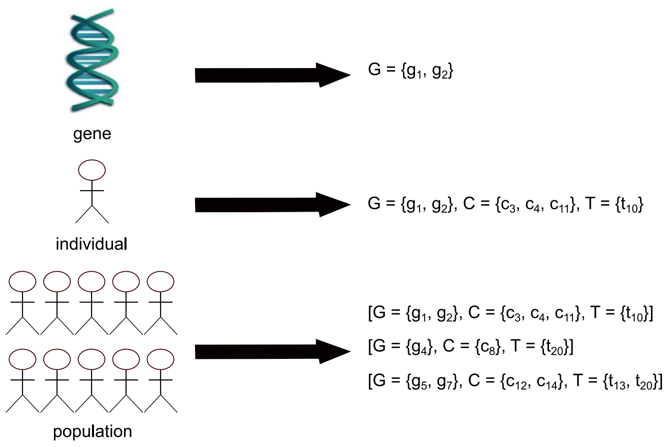
gene
G = {g1, g2}
individual
G = {g1, g2}, C = {c3, c4, c11}, T = {t10}
population
[G = {g1, g2}, C = {c3, c4, c11}, T = {t10}]
[G = {g4}, C = {c8}, T = {t20}]
[G = {g5, g7}, C = {c12, c14}, T = {t13, t20}]

Fig. 2. Genetic algorithm codification.

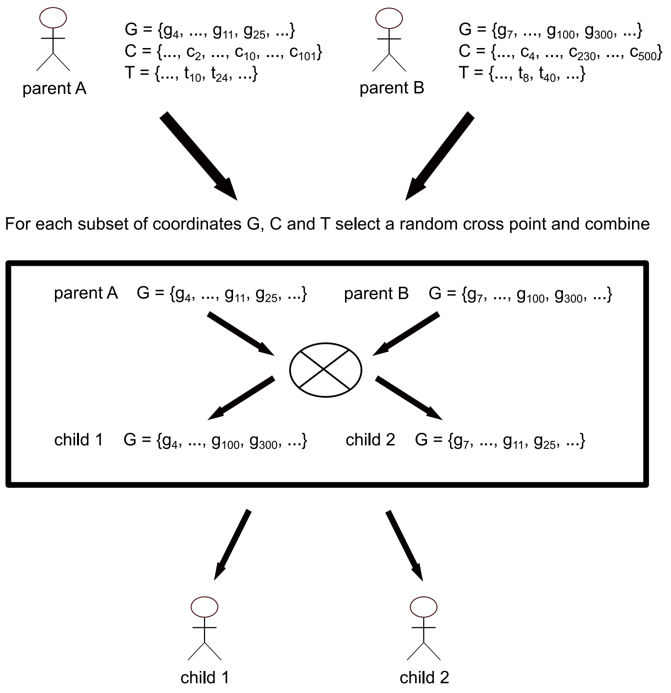
parent A
G = {g4, ..., g11, g25, ...}
C = {..., c2, ..., c10, ..., c101}
T = {..., t10, t24, ...}
parent B
G = {g7, ..., g100, g300, ...}
C = {..., c4, ..., c230, ..., c500}
T = {..., t8, t40, ...}
For each subset of coordinates G, C and T select a random cross point and combine
parent A G = {g4, ..., g11, g25, ...}
parent B G = {g7, ..., g100, g300, ...}
child 1 G = {g4, ..., g100, g300, ...}
child 2 G = {g7, ..., g11, g25, ...}
child 1
child 2

Fig. 3. Representation of the crossover operator.

coordinates of its genes G, conditions C and times T are stored. These new individuals will be created by a random subset $T \times C \times G$ which had not appeared in the previous solutions.

3.3. *Crossover operator*

Two individuals (parents, P_1 and P_2) are combined to create two new individuals (offsprings, O_1 and O_2). They are chosen based on a probability of crossover parameter, denoted by p_c. Their genetic material is combined by a random one-point cross in the genes G, conditions C and times T and mixing the coordinates in both children.

Formally, let be $P_1{}^i$ and $P_2{}^i$ parents #1 and #2 at iteration i. The resulting offspring at this iteration is

$$\{O_1^i, O_2^i\} = \mathfrak{f}_{C,G,T}(P_1^i, P_2^i) \tag{4}$$

where $\mathfrak{f}$ is the function than randomly selects a subset of (C, G, T) from parents $P_1{}^i$ and $P_2{}^i$ at iteration i, given a probability of crossover $p_c{}^i$. The whole procedure is illustrated in Fig. 3.

The procedure can be formalized as follows. Let $S_1 = \langle g_{i_1}, \ldots, g_{i_m} \rangle$ and $S_2 = \langle g'_{j_1}, \ldots, g'_{j_n} \rangle$ be two sequences of genes, where $g_{i_k}, g_{j_l} \in G$, $i_1 < \cdots < i_m$, $j_1 < \cdots < j_n$ and $m, n < B$.

Let p be a number randomly chosen from 1 and $\min(m, n)$. The new two individuals or children are formed as follows:

$$O_1 = \langle g_{i_1}, \ldots, g_{i_p}, g'_{j_{p+1}}, \ldots, g'_{j_n} \rangle \tag{5}$$

$$O_2 = \langle g'_{j_1}, \ldots, g'_{j_p}, g_{i_{p+1}}, \ldots, g_{i_m} \rangle \tag{6}$$

Note that after the creation of both offsprings, the elements must be reordered since positions j_l may not be necessarily greater than positions i_k and viceversa. Additionally, both S_1 and S_2 may share any gene, as they are randomly formed from the input matrix. Should it be the case, repeated elements must be removed from O_1 and O_2. Therefore, the number of genes composing O_1 may eventually be less than n and the number of genes in O_2 could be less than m.

The full process is analogously performed for conditions and time stamps.

3.4. Mutation operator

An individual can be mutated according to a probability of mutation, p_m. The p_m condition is verified for every individual and if it is satisfied, one out of six possible actions is taken. These actions are: add a new random gene coordinate to G, add a new condition coordinate to C or add a new time coordinate to T, or by removing a random coordinate from either G, C or T. The election of these actions is also random. For the case of addition of new coordinates, it is first checked that the new randomly selected coordinate is not included yet in G, C or T. If so, the process is repeated until finding a new coordinate that did not originally belong to the individual. Again, the sequence must be reordered so that the new sequence remains ordered.

3.5. Selection operator

This operator is implemented following the roulette wheel selection method [23]. The fitness level is used to associate a probability of selection with each individual of the population. This emulates the behavior of a roulette wheel in a casino. Usually a proportion of the wheel is assigned to each of the possible selections based on their fitness value. Then a random selection is made similar to how the roulette wheel is rotated. While candidates with a higher fitness will be less likely to be eliminated, there is still a chance that they might be. There is a chance that some weaker solutions may survive the selection process, which is an advantage, as though a solution may be weak, it may include some component which could prove useful following the recombination process.

3.6. Fitness function

The fitness of each individual allows the algorithm to determine which are the best candidates to remain in subsequent generations. For the *TriGen* algorithm, we have implemented two different fitness functions, the first one based on a three dimensions adaptation of the Mean Square Residue measure (MSR) which is a classic biclustering measure for gene expression analysis [9], from now on referred as f_{MSR}. The second one f_{LSL}. The first function f_{MSR} is defined based on a distance function, while f_{LSL} is more suitable to find correlations among a set of genes even if they are expressed at different levels of magnitude.

Both functions share some common terms which are now defined.

- G: Subset of gene coordinates of the individual.
- C: Subset of condition coordinates of the individual.
- T: Subset of time coordinates of the individual.
- T_l: Number of time coordinates of the individual.
- C_l: Number of condition coordinates of the individual.
- G_l: Number of gene coordinates of the individual.
- $TC_v(t, g, c)$: Expression level of gene g under condition c at time t as seen in the input dataset D.

***Weights* term 1.** The *Weights* term is defined as

$$Weights = G_l * w_g + C_l * w_c + T_l * w_t \tag{7}$$

where w_g, w_c and w_t are weights for the number of genes, conditions and times in a tricluster solution TC, respectively. With high values of the weights, we favor that *TriGen* finds solutions with many components for that term.

***Distinction* term 1.** The *Distinction* term is defined as

$$Distinction = \frac{CDN_g}{G_l} * wd_g + \frac{CDN_c}{C_l} * wd_c + \frac{CDN_t}{T_l} * wd_t \tag{8}$$

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 are not present in the tricluster being evaluated, and wd_g, wd_c and wd_t are the distinction weights of the genes, conditions and times respectively. *Distinction* measures how different the individual under evaluation is compared to the triclusters previously found. If the values of the weights are increased, we favor finding non-overlapping solutions to the previously found.

3.6.1. f_{MSR} fitness function

This function is defined by the following equation:

$$f_{MSR}(TC) = MSR - Weights - Distinction \tag{9}$$

It is a minimizing function (the smaller the value the better) which has three terms, the general *Weigths* and *Distinction* terms and the specific *MSR* term. The *MSR* term can be explained as follows.

***MSR* term 1.** Since triclustering emerges as an improvement of biclustering to analyze microarray data taking into account the temporal dimension, we have adapted a classical biclustering fitness function, Mean Squared Residue (*MSR*), presented by Cheng and Church in [9], 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:

$$MSR = \frac{\sum_{g \in G, c \in \in C, t \in T} r_{gct}^2}{G_l * C_l * T_l} \tag{10}$$

where r_{gct} can be defined as

$$r_{gct} = TC_v(t, g, c) + M_{GC}(t) + M_{GT}(c) + M_{CT}(g) - M_G(c, t) - M_C(g, t) - M_T(g, c) - M_{GCT} \tag{11}$$

where $M_{GC}(t)$ 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

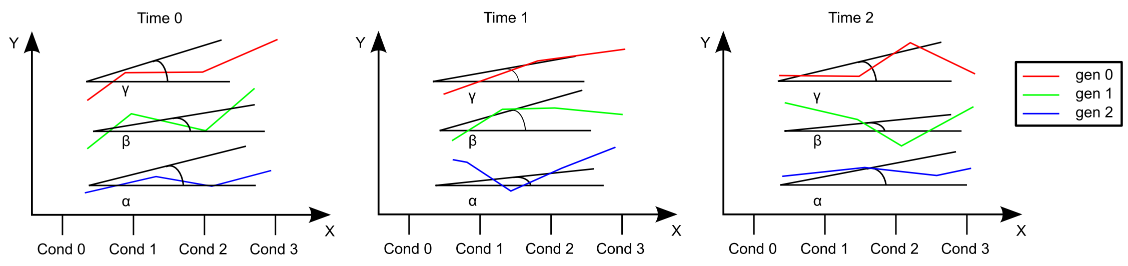

Fig. 4. This solution has $G=3$, $C=4$ and $T=3$. This view represents for each of the time coordinates (0, 1 and 3) conditions in the x-axis, expression levels in the y-axis and the outlines are genes. The slopes of each of the least square approximations α, β and γ are compared in order to detect the correlation among the patterns. The same process is followed with the other views.

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 values in the tricluster.

3.6.2. f_{LSL} Fitness function

This function is defined by the following equation:

$$f_{LSL}(TC) = LSL - Weights - Distinction \tag{12}$$

It is a minimizing function (the smaller the value the better) which has three terms, the general *Weigths* and *Distinction* terms and the specific *LSL* term, explained as follows.

***LSL* term 1.** This term is defined by Eq. (13).

$$LSL = \frac{T_r + C_r + G_r}{3} \tag{13}$$

It measures the similitude between the least squares approximation for the points in each graphic of the three views that represent a tricluster: first, for each time coordinate, conditions in the x-axis, expression levels in the y-axis and the outlines are genes (T_r in (13)); second, for each condition coordinate, times in the x-axis, expression levels in the y-axis and the outlines are genes (C_r in (13)); and third, for each condition coordinate, genes in the x-axis, expression levels in the y-axis and the outlines are times (G_r in (13)). We can see a representation of this in Fig. 4.

All elements in the numerator in Eq. (13) have in common the values indicated in equation group (14).

$$sumX_{tc} = \sum_{g \epsilon G} g \quad sumXX_{tc} = \sum_{g \epsilon G} g^2 \tag{14a}$$

$$sumX_g = \sum_{t \epsilon T} t \quad sumXX_g = \sum_{t \epsilon T} t^2 \tag{14b}$$

where $sumX_{tc}$ is the summation of all genes of the individual under evaluation, $sumXX_{tc}$ is the squared summation of all genes, $sumX_g$ is the summation of all times and $sumXX_g$ is the is the squared summation of al times.

As you can see in Eq. (15a), T_r is the distance between all least square approximations in the first view described above.

$$T_r = \frac{\sum_{t_i,t_j \epsilon T} |TD_{t_i} - TD_{t_j}|}{(T_l - 1) * T_l} \tag{15a}$$

$$\forall t \in T, \quad TD_t = \frac{G_l * sumXY_t - (sumX_{tc} * sumY_t)}{G_l * sumXX_{tc} - sumX_{tc}^2} \tag{15b}$$

$$\forall t \in T, \quad sumXY_t = \sum_{g \epsilon G} \sum_{c \epsilon C} g * IN_v(t,g,c) \tag{15c}$$

$$\forall t \in T, \quad sumY_t = \sum_{g \epsilon G} \sum_{c \epsilon C} IN_v(t,g,c) \tag{15d}$$

where measures are represented by *TD* (15b) and $sumXY_t$ and $sumY_t$ are, respectively, for each individual time, the summation of each expression level value of this time and all combinations of individual genes and conditions multiply by the genes (15c) and for each individual time, the summation of each expression level value of this time and all combinations of individual genes and conditions (15d).

In the same way, Eq. (16a) defines C_r term as the distance between all least square approximations produced in the second view.

$$C_r = \frac{\sum_{c_i,c_j \epsilon C} |CD_{c_i} - CD_{c_j}|}{(C_l - 1) * C_l} \tag{16a}$$

$$\forall c \in C, \quad CD_c = \frac{G_l * sumXY_c - (sumX_{tc} * sumY_c)}{G_l * sumXX_{tc} - sumX_{tc}^2} \tag{16b}$$

$$\forall c \in C, \quad sumXY_c = \sum_{t \epsilon T} \sum_{g \epsilon G} g * IN_v(t,g,c) \tag{16c}$$

$$\forall c \in C, \quad sumY_c = \sum_{t \epsilon T} \sum_{g \epsilon G} IN_v(t,g,c) \tag{16d}$$

where measures are represented by *CD* (16b) and $sumXY_c$ and $sumY_c$ are, respectively, for each individual condition, the summation of each expression level value of this condition and all combinations of individual times and genes multiply by the genes (16c) and for each individual condition, the summation of each expression level value of this condition and all combinations of individual times and genes (16d).

Finally, Eq. (17a) defines the G_r term as the distance between all least square approximations produced in the third view described above.

$$G_r = \frac{\sum_{c_i,c_j \epsilon C} |GD_{c_i} - GD_{c_j}|}{(C_l - 1) * C_l} \tag{17a}$$

$$\forall c \in C, \quad GD_c = \frac{T_l * sumXY_g - (sumX_g * sumY_g)}{T_l * sumXX_g - sumX_g^2} \tag{17b}$$

$$\forall c \in C, \quad sumXY_c = \sum_{g \epsilon G} \sum_{t \epsilon T} t * IN_v(t,g,c) \tag{17c}$$

$$\forall c \in C, \quad sumY_c = \sum_{g \epsilon G} \sum_{t \epsilon T} IN_v(t,g,c) \tag{17d}$$

where measures are represented by *GD* (17b) and $sumXY_c$ and $sumY_c$ are, respectively, for each individual condition, the summation of each expression level value of this condition and all combinations of individual genes and times multiply by the times (17c) and for each individual condition, the summation of each expression level value of this condition and all combinations of individual genes and times (17d).

4. Results

We show the results obtained applying the *TriGen* algorithm both to real and synthetic data. Synthetic data has the advantage that the process that generates the data is well known and so one is able to judge the success or failure of the algorithm [24]. Synthetic datasets generation has been widely applied both in microarray related publications [5,15], and in other general data mining applications [26].

TriGen takes several parameters: probability of crossover p_c, probability of mutation p_m, weights w_g for the number of genes, w_c for the conditions and w_t for the times, used to calculate the term *Weights* (see Section 3.6), and weights wd_g, wd_c and wd_t are the distinction weights of the genes, conditions and times respectively, used to calculate the term *Distinction* (see Section 3.6).

All experiments were executed on a multiprocessor machine with 64 processors Intel Xeon E7-4820 2.00 GHz with 8 GB RAM memory. We now describe the results obtained.

4.1. Results on the synthetic dataset

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 value which represents its level of expression, i.e., quantification of the mRNA present in a time point under certain conditions. This value has been randomly chosen from the rank, respectively for each condition, $[1, 15]$, $[7, 35]$, $[60, 75]$, $[0, 25)$, $[30, 100]$, $[71, 135]$, $[160, 375]$, $[5, 30]$, $[25, 40]$ and $[10, 30]$. In such data set, we have allocated two areas (a and b) of prefixed values:

Area a: It is an area of size $G_l = 20$, $C_l = 5$ and $T_l = 3$ with all its values fixed to 1.

Area b: It is an area with $G_l = 30$, $C_l = 4$ and $T_l = 4$ with an ascending pattern at times t=0,1 and descending at times t=2, 3 at different levels of magnitude $[1, 15]$, $[60, 75]$, $[5, 30]$ and $[160, 375]$.

TriGen was executed with each of the fitness functions available, f_{MSR} and f_{LSL} with the parameters in Table 1. The weights w_c and w_t are high in relation to w_g since the genes show high dimensionality in relation to conditions and times. The distinction is not taken into account in these executions. The *TriGen* algorithm did successfully find areas *a* and *b* in both executions. The execution with fitness function f_{MSR} found the whole area *a* and missed some genes (seven out of thirty) and missed one time point (three out of four). The execution with f_{LSL} found tricluster solutions for both areas *a* and *b*. This might be due to the fact that area *a*, with fixed values can be easily detected both by distance and correlation measures. In contrast, area *b* exhibits correlation patterns, and therefore is easily found by f_{LSL} and f_{MSR} was capable of only finding part of it.

Table 1
Parameters for the execution of the *TriGen* algorithm on the synthetic dataset.

Parameter	Value
Number of solutions	4
Generations	50
Members in the population	100
p_c	0.7
p_m	0.3
w_g	0.01
w_c	0.55
w_t	0.35
wd_g	0.0
wd_c	0.0
wd_t	0.0

4.2. Results on the yeast dataset

We have applied the *TriGen* algorithm to the yeast (*Saccharomyces cerevisiae*) cell cycle problem [33]. 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 at finding meaningful patterns on this cell cycle. In this experiment, 6179 genes are analyzed under 6 conditions, termed cln3, clb2, pheromone, cdc15, cdc28 and elutriation [33]. 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. To apply the *TriGen* algorithm we did not take into account the conditions with only 2 time points and we used the first 14 time points of the pheromone, cdc15, cdc28 and elutriation experiment, in order to have a compact dataset. Therefore our dataset contains 6179 genes, 4 conditions and 14 time points.

For this experiment, after some preliminary executions, we chose the set of parameters in Table 2 and executed both with the f_{MSR} fitness function and with the f_{LSL}. The value of w_g was set to 0.8 favor the presence of several genes in the solutions, since the algorithm had a tendency to provide solutions with a low number of genes. This might be due to the fact that the genes do not exhibit exactly equal patterns, neither for the distance nor the correlation measure. For the parameters related to the *Distinction* term we provide a high value for the genes in order to cover as much space in this dimension as possible. We also provide two measures to better understand the results obtained: overlapping of the 20 triclusters obtained, with a value of 0.1, this value due to the overlapping of conditions and times, which appear repeated in the solutions due to their low dimensionality, and coverage of the solutions, which represents the proportion of cells from the input dataset D present in the solutions.

We were interested in the relation among the 20 triclusters generated by f_{MSR} and the 20 generated by f_{LSL} (see Table 2). We calculated the average overlapping of the two groups, comparing each tricluster generated by f_{MSR} to each tricluster generated by f_{LSL}. On average, the overlapping of the genes is as low as 0.07%, while the overlapping of conditions is 75% and the overlapping of time points is 53%. The percentages agree with the number of potential genes, conditions and time points for each tricluster, 6179, 6 and 14 respectively.

For legibility reasons, we focus on one of the solutions. It was obtained with the f_{MSR} measure. It groups 20 genes under 2 conditions, pheromone (condition 0) and cdc28 (condition 3) and five

Table 2
Parameters for the execution of the *TriGen* algorithm on the yeast cell cycle dataset.

Parameter	Value
Number of solutions	20
Generations	400
Members in the population	200
p_c	0.8
p_m	0.5
w_g	0.8
w_c	0.1
w_t	0.1
wd_g	1.0
wd_c	0.0
wd_t	0.0
Overlapping	0.1
Coverage	0.08

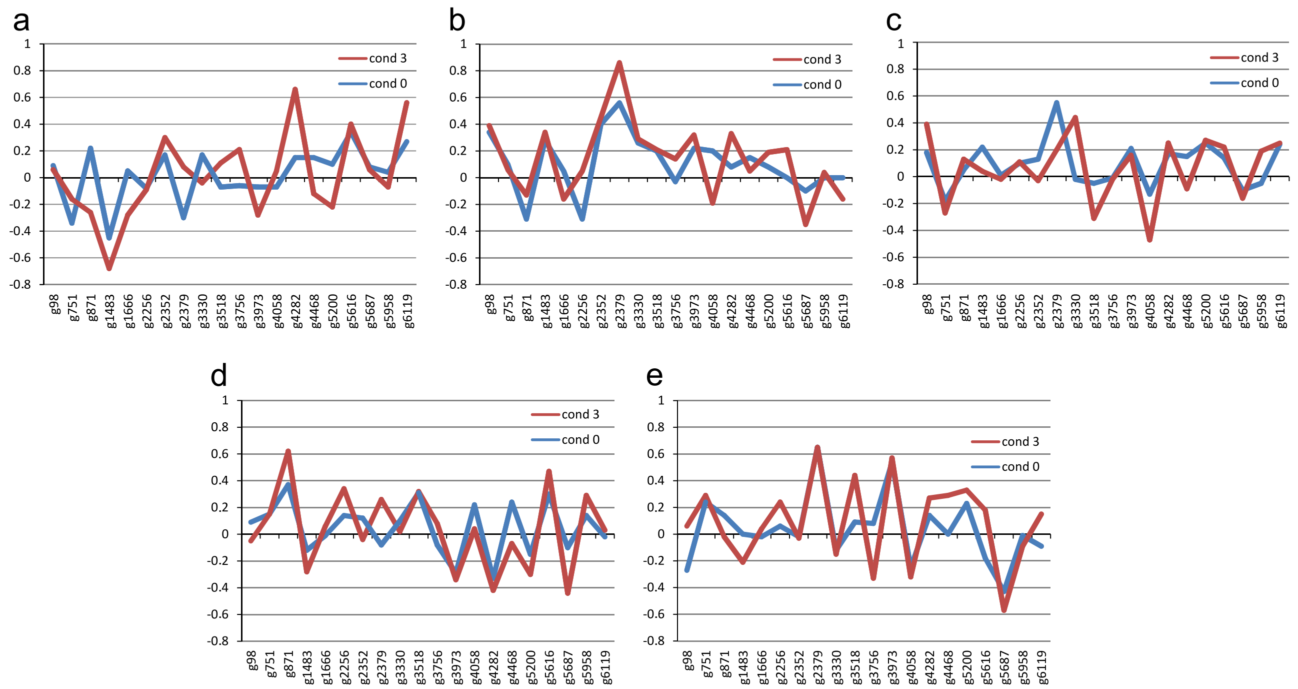
Fig. 5. Gene expression values under two conditions at time point 6 (a), 7 (b), 8 (c), 9 (d) and 10 (e). (a) time point 6 (b) time point 7 (c) time point 8 (d) time point 9 and (e) time point 10.

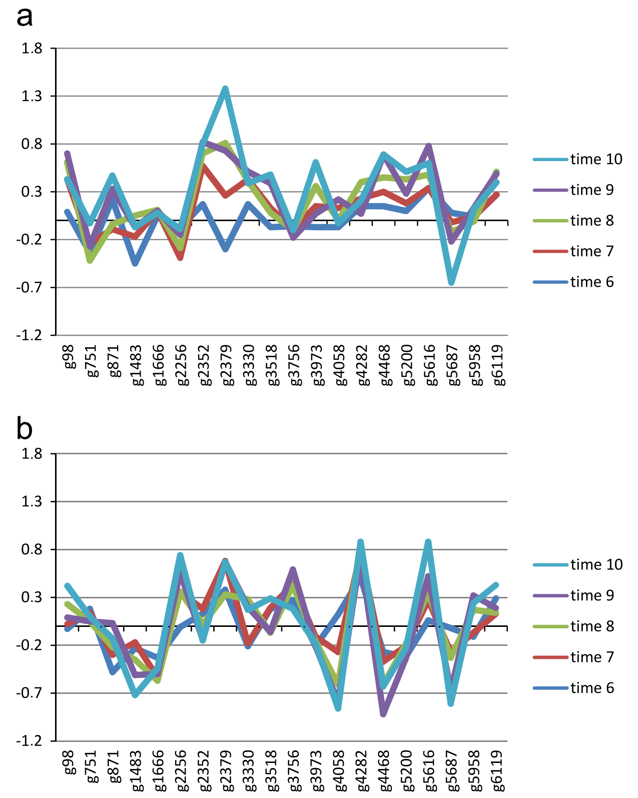
Fig. 6. Gene expression values under five time points at pheromone (a) and cdc28 (b) experiments. (a) condition 0 and (b) condition 3.

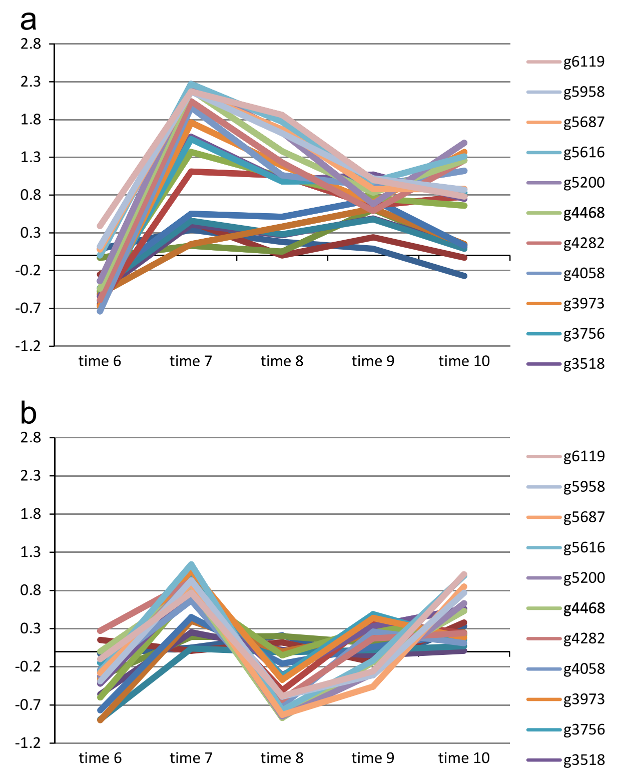
Fig. 7. Gene expression for five time points under gene solution set at pheromone (a) and cdc28 (b) experiments. (a) condition 0 and (b) condition 3.

time points (6, 7, 8, 9 and 10). The graphics are organized in three groups related to this solution: in Fig. 5 we present the outline of the gene expression values (*y*-axis) for each gene in the solution (*x-axis*) comparing the pheromone (condition 0) and cdc28 (condition 3). A graphic is provided for each time point: 6(a), 7(b), 8(c), 9 (d) and 10(e). In Fig. 6, we present the outline of gene expression values (*y*-axis) for each gene in the solution (*x*-axis) comparing time points 6, 7, 8, 9 and 10 and providing a graphic for each condition in the solutions: pheromone (a) and cdc28 (b). Finally in Fig. 7 we present the outline of gene expression values (*y*-axis) for each time point (*x*-axis) providing a graphic for each condition: pheromone (a) and cdc28 (b). As we see, in the three dimensions depicted the

Table 3
GO for yeast cell cycle results.

Cluster ID	Biological process	Molecular function	Cellular component	PI
19		Pyruvate dehydrogenase (lipoamide) phosphatase activity	Actin cap	8.54E – 05
2		Endopeptidase activity	Vacuolar lumen (sensu Fungi)	1.18E – 03
14	Protein metabolism, cytokinesis			1.79E – 03
25	Protein metabolism		Nucleus	2.47E – 03
9	Cytokinesis, regulation of biological process			5.64E – 03
23		Transcription regulator activity, binding, catalytic activity	Intracellular	1.54E – 02
12	Oligosaccharide-lipid intermediate assembly		Cytoplasm	1.71E – 02
21	Organelle organization and biogenesis, cell proliferation		Intracellular membrane-bound organelle	1.72E – 02
11		Signal transducer activity	Intracellular, Organelle	1.96E – 02
31	Cytokinesis, cellular morphogenesis	Structural constituent of cytoskeleton		1.96E – 02
22	Protein complex assembly, cell proliferation		Nucleus, protein complex	2.60E – 02
30	Ribosome biogenesis and assembly, metabolism	Transporter activity	Cytoplasm, nucleus	2.68E – 02
6		Copper ion binding		2.93E – 02
16	Protein amino acid dephosphorylation		Cytoplasm	3.52E – 02
13	Regulation of biological process		Cell wall (sensu Fungi), intracellular	3.5E – 02
34	Cellular process, protein biosynthesis		Cytoplasm	3.76E – 02
10	Vacuole fusion, non-autophagic, regulation of phosphate metabolism, signal transduction, pseudohyphal growth		Nucleus	3.99E – 02

behavior of the genes is very similar throughout the conditions and the times.

We provide a validation of the solutions obtained based on the Gene Ontology project (GO) [2]. GO is a major bioinformatics initiative with the aim of standardizing the representation of gene and gene product attributes across species and databases. The project provides an ontology of terms for describing gene product characteristics and gene product annotation data. The ontology covers three domains: cellular component, the parts of a cell or its extracellular environment; molecular function, the elemental activities of a gene product at the molecular level, such as binding or catalysis; and biological process, operations or sets of molecular events with a defined beginning and end, pertinent to the functioning of integrated living units: cells, tissues, organs, and organisms. We have queried the terms associated to our genes in GO using the *Onto-CC* tool [29,28], an automatic method specially suited for independent validation of gene grouping hypotheses (e.g. co-expressed genes) based on GO clusters (i.e. expression versus GO). *Onto-CC* reduces the uncertainty of the queries by identifying optimal conceptual clusters that combine terms from different ontologies simultaneously, as well as terms defined at different levels of specificity in the GO hierarchy. A probability of intersection p-value is given for each group of genes returned by *Onto-CC*. This value lies in the [0–1] interval and groups of genes are considered as relevant with values below 0.05. In Table 3, we show the validation results using the Onto-CC program for this tricluster solution. The p-values obtained are quite low (up to 8.54E – 05) and the terms provided are very specific and many of them directly related to the cell cycle such as those in cluster IDs 2, 9, 10 and 21.

For further validation of the solutions provided by *TriGen*, we have executed the *Onto-CC* tool with ten groups of genes randomly chosen from the 6179 genes in the yeast dataset, each group containing 20 genes as in the tested solution. The best (minimum) p-value obtained has been 0.025, and the deepest level in the ontologies has been 3, very generic terms. Therefore, we can conclude that the genes selected by *TriGen* have a closer relation in terms of GO annotations than expected if they were selected at random.

Table 4
Parameters for the execution of the *TriGen* algorithm on the inflammation and host response to injury dataset.

Parameter	Value
Number of solutions	20
Generations	100
Members in the population	200
p_c	0.7
p_m	0.4
w_g	0.9
w_c	0.0
w_t	0.1
wd_g	0.9
wd_c	0.0
wd_t	0.1
Overlapping	0.08
Coverage	0.2

4.3. Results on the inflammation and host response to injury dataset

The problem under study deals with human inflammation and the host response to injury. Understanding the inflammation process is critical because the body uses inflammation to protect itself from infection or injury (e.g., crushes, massive bleeding, or a serious burn). The host response to trauma and burns is a collection of biological and pathological processes that depends critically upon the regulation of the human immuno-inflammatory response [8].

The data has been acquired from an experiment about inflammation and host response to injury carried out with microarrays. In this experiment, blood samples from 8 volunteers are analyzed, 4 treated with a toxin that simulates an inflammatory process and 4 with a placebo. Samples were taken at 6 time points throughout 24 h, obtaining a total of 48 microarrays. We work with a set of 2155 genes selected as relevant for the problem [30] considering 2 conditions: endotoxin and placebo.

The set of parameters chosen can be seen in Table 4 and TriGen was executed both with the f_{MSR} fitness function and with the f_{LSL}

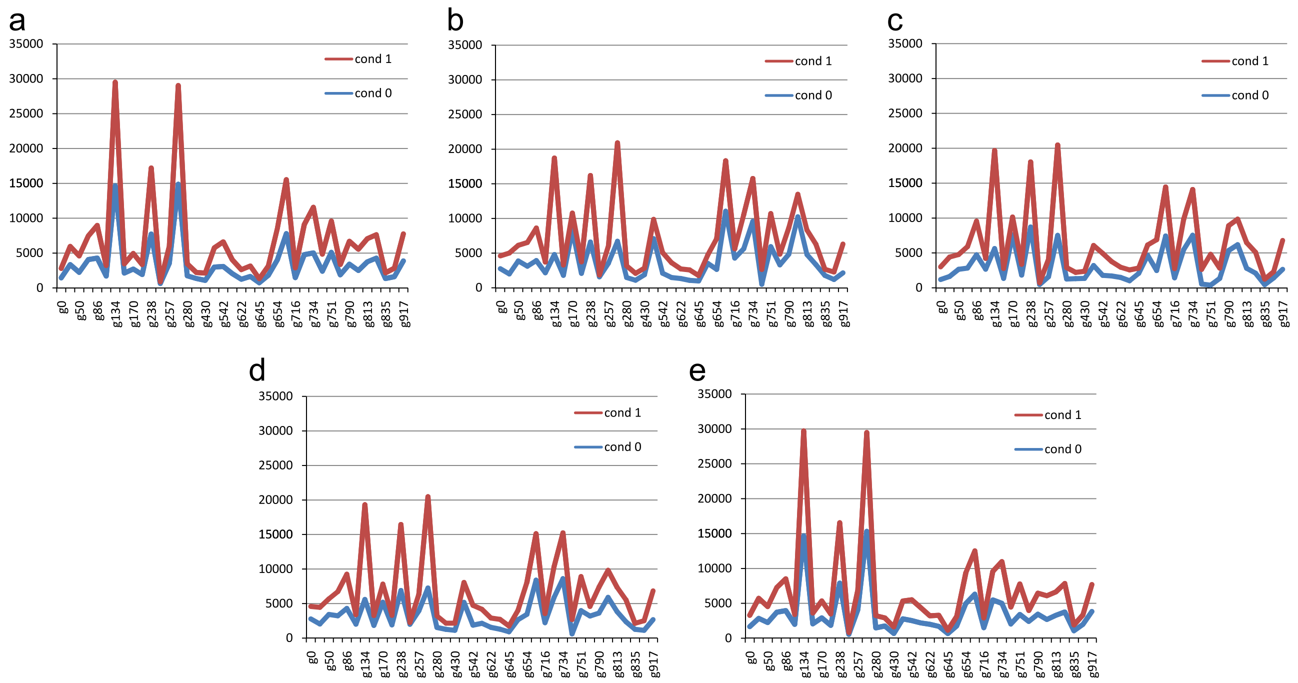

Fig. 8. Gene expression values under two conditions at time point 0 (a), 1 (b), 3 (c), 4 (d) and 5 (e).

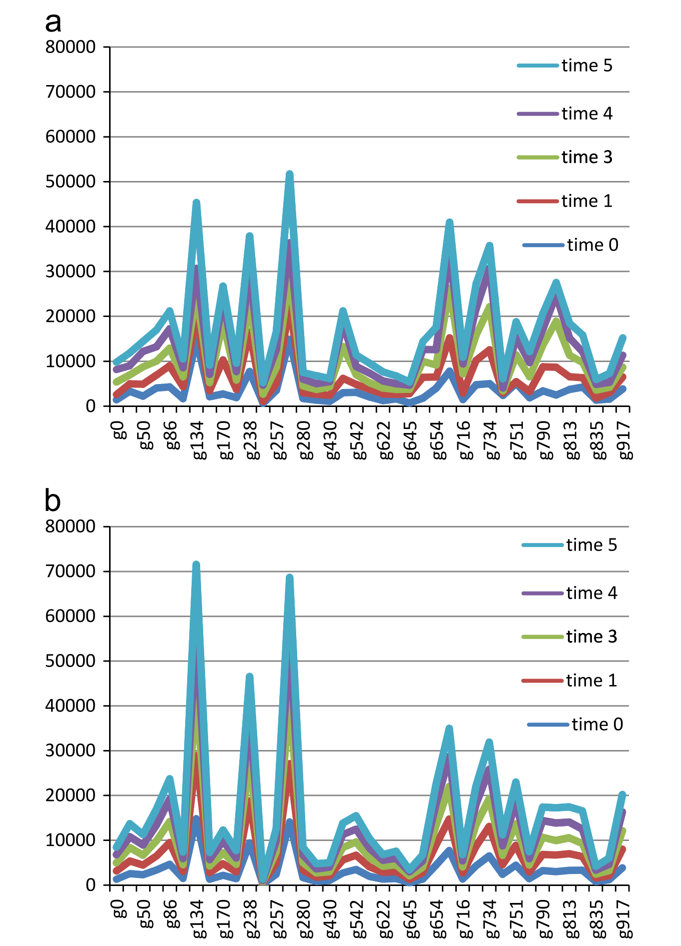

Fig. 9. Gene expression values under five time points at condition 0 (a) and condition 1 (b) experiments.

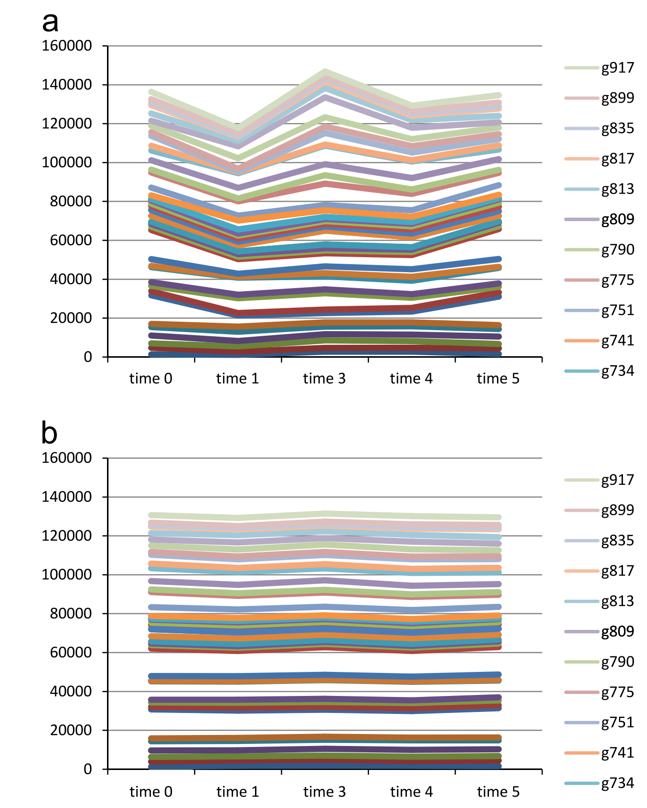

Fig. 10. Gene expression for five time points under gene solution set at condition 0 (a) and condition 1 (b) experiments.

one. We can see that the number of iterations has been decreased compared to the yeast cell cycle experiment. This is due to the dimensionality of the experiments. In the yeast cell cycle problem with $G=6179$, $C=4$ and $T=14$ the total number of cells to visit is 346,024. For the inflammation and host response to injury problem, $G=2155$, $C=2$ and $T=5$, a total number of 25,860 cells to visit. Therefore, *TriGen* does not need to iterate so many times. In this case, we also favor the diversity of the genes found by setting wd_g to 0.9.

We also provide two measures of the solutions, overlapping of the 20 triclusters obtained, with a low value because genes, which are the ones with more influence in this measure due to the dimensionality, are very diverse. Coverage of the solutions is 0.2 from the input dataset.

We also provide information about the relation among the 20 triclusters generated by f_{MSR} and the 20 generated by f_{LSL} (see Table 4). We calculated the average of the overlapping of the two groups, comparing each tricluster generated by f_{MSR} to each tricluster generated by f_{LSL}. On average, the overlapping of the genes is 1.5%, while the overlapping of conditions is 100% and the overlapping of time points is 78%. The percentages agree with the number of potential genes, conditions and time points for each tricluster, 2155, 2 and 6 respectively. In particular, both conditions are present in all the clusters.

Again we focus only on one of the solutions for legibility reasons. It was obtained with the f_{LSL} measure. It groups 11 genes under the 2 conditions, endotoxin (condition 0) and placebo (condition 1) and five time points (0, 1, 2, 3, 4, 5).

We show three groups of graphics related to this solution: In Fig. 8, we present the outline of the gene expression values (y-axis) for each gene in the solution (x-axis) comparing the endotoxin (condition 0) and placebo (condition 1). A graphic is provided for each time point: 0 (a), 1 (b), 3 (c), 4 (d) and 5 (c). In Fig. 9, we present the outline of gene expression values (y-axis) for each sgene in the solution (x-axis) comparing time points 0, 1, 3, 4 and 5 and providing a graphic for each condition: endotoxin (a) and placebo (b). Finally, in Fig. 10 we present the outline of gene expression values (y-axis) for each time point (x-axis) comparing each gene in the solution providing a graphic for each condition: endotoxin (a) and placebo (b).

In this case, we see in each of the three dimensions represented the similarity of the behavior of the genes. However, we can see that the patterns exhibited are extremely correlated, much more than in the yeast cell cycle experiment. In fact, the genes are expressed at different levels of magnitude. We can see in Fig. 9 how the genes in time 0 are expressed in the 0–15,000 interval, while in time 5 the rank in the 9000–70,000 interval. However, since those patterns are correlated they are retrieved by *TriGen*. This is due to the fact that this solution was found using the f_{LSL} based fitness function.

We also provide the validation results using the Onto-CC program for this tricluster solution in Table 5. We see that the *PI* results obtained are low (up to 6.07E − 09) meaning that the groups of genes are significative related to the GO terms associated to them. Furthermore, the terms are very specific as in cluster IDs 6, 33, 51, 66 and in particular the term *immune response* is present in 31.

As for the yeast cell cycle problem, we provide further validation of the solutions provided by *TriGen* executing the *Onto-CC* tool with 10 groups of genes randomly chosen from the 2155 genes in the inflammation and host response to injury problem, each group

Table 5
GO for inflammation and host response to injury results.

Cluster ID	Biological process	Molecular function	Cellular component	PI
71		Cation binding, catalytic activity, transcription regulator activity		6.07E − 09
51	Protein amino acid phosphorylation		Intracellular, extracellular region	6.28E − 09
46	Ion transport	ion transporter activity, protein binding	Integral to membrane, intracellular	2.81E − 07
69	Regulation of transcription, DNA-dependent	Zinc ion binding, protein binding		7.66E − 06
37	Regulation of transcription, DNA-dependent		Cytoplasm	1.12E − 05
61		Binding, ubiquitin-like-protein ligase activity	Cytoplasm	1.88E − 05
4	Regulation of biological process	Binding	Intracellular	5.12E − 05
52	Protein metabolism, regulation of biological process	Catalytic activity, nucleotide binding, nucleic acid binding		5.15E − 05
33	Regulation of Rho protein signal transduction		Intracellular	5.54E − 05
68	Regulation of transcription, DNA-dependent	Transcription regulator activity, catalytic activity	Intracellular	7.75E − 05
29		Anion transporter activity, nucleic acid binding		2.82E − 04
31	Immune response	Obsolete molecular function	Extracellular region	3.21E − 04
7			Intracellular, cell fraction	1.01E − 03
57		Transcription regulator activity, nucleotide binding	Extracellular region	1.08E − 03
6	Regulation of biological process	ARF guanyl-nucleotide exchange factor activity		1.13E − 03
66	Bone resorption, signal transduction, response to stimulus			1.47E − 03
47			Peripheral to membrane of membrane fraction	2.34E − 03
2	Response to stimulus, protein targeting, regulation of biological process	Signal transducer activity	Cell	2.84E − 03
8	Protein targeting	Transcription regulator activity	Intracellular	3.26E − 03
12		Catalytic activity, transporter activity	Extracellular region	4.11E − 03
15	Protein metabolism, regulation of biological process	Catalytic activity, transporter activity	Membrane	5.39E − 03
17		GTPase activator activity, binding	Intracellular	1.07E − 02
16		Protein binding, nucleotide binding, ATP-dependent helicase activity, nucleic acid binding		1.08E − 02
76		Nucleotide binding	Cytoplasm, integral to membrane	1.09E − 02
70			Extracellular region, membrane fraction	1.33E − 02
53	Regulation of transcription, DNA-dependent	ATP binding		1.71E − 02
67		Protein phosphatase type 2C activity		1.84E − 02
55	Regulation of transcription, DNA-dependent	Nucleotide binding, nucleic acid binding, ATP-dependent helicase activity		1.89E − 02

containing 11 genes as in the tested solution. The best (minimum) p-value obtained has been 0.008, and the deepest level in the ontologies has been 2, very generic terms. Therefore, we can conclude that the genes selected by *TriGen* have a closer relation in terms of GO annotations than expected if they were selected at random.

5. Conclusions

We have presented *TriGen*, a triclustering algorithm based on an evolutionary heuristic, genetic algorithms, which finds groups of pattern similarity for genes on a three dimensional space, thus taking into account the gene, conditions and time factor. The genetic operators used have been described in detail, along with the two fitness functions available to use with *TriGen*: f_{MSR} based on a three-dimensional adaptation of the Mean Square Residue measure, a classic biclustering distance measure and f_{LSL}, a correlation measure based on the distance among the slopes of the least square lines from the three views of a tricluster.

The algorithm has been applied to three different datasets: a synthetic dataset, a dataset from a yeast cell cycle experiment and a dataset from the inflammation and host response to injury. The results show that the algorithm finds genes with high similarity in a subset of conditions and times. Moreover, the genes grouped together by the *TriGen* algorithm showed to be biologically related in terms of the functional annotations associated to them in the Gene Ontology project. We have also proved that genes selected by *TriGen* have a closer relation in terms of GO annotations than expected if they were selected at random. We have also seen that the two evaluation functions provide complementary information, since the overlapping of genes between triclusters found by f_{MSR} and by f_{LSL} is very low (0.07% for the yeast cell cycle problem and 1.5% for the inflammation and host response to injury problem).

TriGen is an algorithm created to mine longitudinal experiments with microarray data, but it can be used in other biologically related fields, for instance combining expression data with gene regulation information by means of substituting the time dimension by ChIP-chip data representing transcription factor–gene interactions what can provide us with regulatory network information. This proposal can also be applied to mine RNA-seq data repositories. Also, the seismic regionalization of the Iberian Peninsula is currently being addressed through *TriGen,* turning the 2D-dimensional problem proposed in [27] into a 3D one. In this case, the third component does not identify time stamps but features associated to every pair of geographical coordinates that represent the Iberian Peninsula.

Acknowledgments

The authors want to thank the financial support given by the Spanish Ministry of Science and Technology, projects TIN2011-28956-C00 and TIN2009-13950. This work has also been partially funded by Junta de Andalucía's project TIC-7528 and by JPI 2012 Banco Santander's grant.

References

[1] R.B. Araújo, G.H.T. Ferreira, G.H. Orair, W. Meira, R.A.C. Ferreira, D.O.G. Neto, M.J. Zaki, The ParTriCluster algorithm for gene expression analysis, Int. J. Parallel Progr. 36 (2) (2008) 226–249.

[2] M. Ashburner, C.A. Ball, J.A. Blake, D. Botstein, H. Butler, J.M. Cherry, A.P. Davis, K. Dolinski, S.S. Dwight, J.T. Eppig, M.A. Harris, D.P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J.C. Matese, J.E. Richardson, M. Ringwald, G.M. Rubin, G. Sherlock, Gene ontology: tool for the unification of biology. The Gene ontology consortium, Nat. Genet. 25 (1) (2000) 9.

[3] H. Banka, S. Mitra, Evolutionary biclustering of gene expressions, Ubiquity 5 (2006) 2006.

[4] Z. Bar-Joseph, Analyzing time series gene expression data, Bioinformatics 20 (16) (2004) 2493.

[5] M. Barenco, J. Stark, D. Brewer, D. Tomescu, R. Callard, M. Hubank, Correction of scaling mismatches in oligonucleotide microarray data, BMC Bioinforma. 7 (2006) 251.

[6] A. Ben-Dor, B. Chor, R. Karp, Z. Yakhini, Discovering local structure in gene expresión data: the order-preserving submatrix problem, in: Proceedings of the 6th International Conference on Computational Biology, 2002, pp. 49–57.

[7] P. Brown, D. Botstein, Exploring the new world of the genome with DNA microarrays, Nat. Genet. 21 (Suppl.) (1999) 33–37.

[8] S.E. Calvano, W. Xiao, D.R. Richards, R.M. Felciano, H.V. Baker, R.J. Cho, R.O. Chen, B.H. Brownstein, J.P. Cobb, S.K. Tschoeke, C. Miller-Graziano, L. L. Moldawer, M.N. Mindrinos, R.W. Davis, R.G. Tompkins, S.F. Lowry, I.A. Large Scale Collab Res Program, A network-based analysis of systemic inflammation in humans, Nature 437 (2005) 1032–1037.

[9] Y. Cheng, G.M. Church, Biclustering of expression data, in: International Conference on Intelligent Systems for Molecular Biology, 2000, pp. 93–103.

[10] P. D'haeseleer, S. Liang, R. Somogyi, Genetic network inference: from co-expression clustering to reverse engineering, Bioinformatics 16 (8) (2000) 707–726.

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

[12] G. Getz, E. Levine, E. Domany, Coupled two-way clustering analysis of gene microarray data, Proc. Natl. Acad. Sci. 97 (22) (2000) 12079–12084.

[13] F. Gómez-Vela, F. Martínez-Álvarez, C.D. Barranco, N. Díaz-Díaz, D.S. Rodríguez-Baena, J.S. Aguilar-Ruiz, Pattern recognition in biological time series, in: Lecture Notes in Artificial Intelligence, vol. 7023, 2011, pp. 164–172.

[14] D. Gutiérrez-Avilés, C. Rubio-Escudero, J.C. Riquelme, Revisiting the yeast cell cycle problem with the improved trigen algorithm, in: IEEE World Congress on Nature and Biologically Inspired Computing, 2011, pp. 515–520.

[15] K. Hakamada, M. Okamoto, T. Hanai, Novel technique for preprocessing high dimensional time-course data from DNA microarray: mathematical model-based clustering, Bioinformatics 22 (7) (2006) 843.

[16] J.A. Hartigan, Direct clustering of a data matrix, J. Am. Stat. Assoc. 67 (337) (1972) 123–129.

[17] Z. Hu, R. Bhatnagar, Algorithm for discovering low-variance 3-clusters from real-valued datasets, in: IEEE International Conference on Data Mining, 2010, pp. 236–245.

[18] H. Jiang, S. Zhou, J. Guan, Y. Zheng, gTRICLUSTERL: a more general and effective 3D clustering algorithm for gene-sample-time microarray data, in: Lecture Notes in Computer Science, vol. 3916, 2006, pp. 48–59.

[19] Z. Li, X. Hu, Y. Chen, Multi-objective evolutionary algorithm for mining 3D clusters in gene-sample-time microarray data, in: IEEE International Conference on Granular Computing, 2008, pp. 442–447.

[20] Y.C. Liu, C.H. Lee, W.C. Chen, J.W. Shin, H.H. Hsu, V.S. Tseng, A novel method for mining temporally dependent association rules in three-dimensional microarray datasets, in: IEEE International Computer Symposium, 2010, pp. 759–764.

[21] S.C. Madeira, A.L. Oliveira, Biclustering algorithms for biological data analysis: a survey, IEEE/ACM Trans. Comput. Biol. Bioinform. 1 (1) (2004) 24–45.

[22] P. Mahanta, H.A. Ahmed, D.K. Bhattacharyya, J.K. Kalita, Triclustering in gene expression data analysis: a selected survey, IEEE International Conference on Emerging Trends and Applications in Computer Science, 2011, pp. 1–6.

[23] M. Martínez-Ballesteros, F. Martínez-Álvarez, A. Troncoso, J.C. Riquelme, An evolutionary algorithm to discover quantitative association rules in multi-dimensional time series, Soft Comput. 15 (2011) 2065–2084.

[24] P. Mendes, GEPASI: a software package for modelling the dynamics, steady states and control of biochemical and other systems, Comput. Appl. Biosci. 9 (2003) 563–571.

[25] S. Mitra, H. Banka, Multi-objective evolutionary biclustering of gene expression data, Pattern Recognit. 39 (12) (2006) 2464–2477.

[26] R.P. Pargas, M.J. Harrold, R.R. Peck, Test-data generation using genetic algorithms, Softw. Test. Verif. Reliab. 9 (4) (1999) 263–282.

[27] J. Reyes, V. Cárdenas, A Chilean seismic regionalization through a Kohonen neural network, Neural Comput. Appl. 19 (2010) 1081–1087.

[28] R. Romero-Záliz, C. Del Val, J.P. Cobb, I. Zwir, Onto-cc: a web server for identifying gene ontology conceptual clusters, Nucl. Acids Res. 36 (Suppl. 2) (2008) W352–W357.

[29] R. Romero-Záliz, C. Rubio-Escudero, J.P. Cobb, F. Herrera, O. Cordón, I. Zwir, A multiobjective evolutionary conceptual clustering methodology for gene annotation within structural databases: a case of study on the Gene Ontology database, IEEE Trans. Evol. Comput. 12 (6) (2008) 679–701.

[30] C. Rubio-Escudero, Fusion of knowledge towards the identification of genetic profiles, AI Commun. 25 (1) (2012) 65–67.

[31] C. Rubio-Escudero, F. Martínez-Álvarez, R. Romero-Záliz, I. Zwir, Classification of gene expression profiles: comparison of K-means and expectation maximization algorithms, in: Proceedings of the IEEE International Conference on Hybrid Intelligent Systems, 2008, pp. 831–836.

[32] K. Sim, Z. Aung, Discovering correlated subspace clusters in 3D continuous-valued data, in: IEEE International Conference on Data Mining, 2010, pp. 471–480.

[33] P.T. Spellman, G. Sherlock, M.Q. Zhang, V.R. Iyer, K. Anders, M.B. Eisen, P.O. Brown, D. Botstein, B. Futcher, Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization, Mol. Biol. Cell 9 (3) (1998) 3273–3297.

[34] M.P. Tan, E.N. Smith, J.R. Broach, C.A. Floudas, Microarray data mining: a novel optimization-based approach to uncover biologically coherent structures, BMC Bioinforma. 9 (2008) 268.

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

[36] A.B. Tchagang, S. Phan, F. Famili, H. Shearer, P. Fobert, Y. Huang, J. Zou, D. Huang, A. Cutler, Z. Liu, Mining biological information from 3D short time-series gene expression data: the OPTricluster algorithm, BMC Bioinforma. 13 (1) (2012) 54.

[37] R. Tibshirani, T. Hastie, M. Eisen, D. Ross, D. Botstein, P. Brown, et al., Clustering Methods for the Analysis of DNA Microarray Data, Technical Report, Department of Statistics, Stanford University, 1999.

[38] G. Wang, L. Yin, Y. Zhao, K. Mao, Efficiently mining time-delayed gene expression pattern, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 40 (2) (2010) 400–411.

[39] X. Xu, Y. Lu, K.L. Tan, A. Tung, Finding time-lagged 3D clusters, in: IEEE International Conference on Data Engineering, 2009, pp. 445–456.

[40] Y. Yin, Y. Zhao, B. Zhang, G. Wang, Mining time-shifting co-regulation patterns from gene expression data, in: Lecture Notes in Computer Science, vol. 4505, 2007, pp. 62–73.

[41] L. Zhao, M.J. Zaki, TRICLUSTER: an effective algorithm for mining coherent clusters in 3D microarray data, in: ACM SIGMOD International Conference on Management of Data, 2005, pp. 694–705.