Biclustering of Expression Data with Evolutionary Computation

Federico Divina and Jesús S. Aguilar-Ruiz

Abstract—Microarray techniques are leading to the development of sophisticated algorithms capable of extracting novel and useful knowledge from a biomedical point of view. In this work, we address the biclustering of gene expression data with evolutionary computation. Our approach is based on evolutionary algorithms, which have been proven to have excellent performance on complex problems, and searches for biclusters following a sequential covering strategy. The goal is to find biclusters of maximum size with mean squared residue lower than a given δ. In addition, we pay special attention to the fact of looking for high-quality biclusters with large variation, i.e., with a relatively high row variance, and with a low level of overlapping among biclusters. The quality of biclusters found by our evolutionary approach is discussed and the results are compared to those reported by Cheng and Church, and Yang et al. In general, our approach, named SEBI, shows an excellent performance at finding patterns in gene expression data.

Index Terms—Biclustering, gene expression data, evolutionary computation.

1 Introduction

Microarray techniques may provide massive amounts of information, which is leading to the development of sophisticated algorithms capable of extracting novel and useful knowledge from a biomedical point of view. Microarray data are widely used in genomic research due to the enormous potential in gene expression profiling, facilitating the prognosis and the discovering of subtypes of diseases.

The gene expression data are organized in matrices, where rows represent genes and columns represent experimental conditions. Each element in the matrix refers to the expression level of a particular gene under a specific condition. A basic approach to the study of expression data consists of applying traditional statistical techniques. In many problems, these methods have been shown to be unable to extract relevant knowledge from data.

Clustering has been applied to gene expression data [1], which usually refers to conditions or patients, although genes can also be grouped in order to search for functional similarities. However, relevant genes are not necessarily related to every condition or, in other words, there are genes that can be relevant for a subset of conditions [2]. On the contrary, it is also possible to discriminate groups of conditions by using different groups of genes. From this point of view, clustering cannot only be addressed horizontally (conditions) or vertically (genes), but also in the two dimensions simultaneously. This approach, named *biclustering* or *subspace clustering*, identifies groups of genes that show a "similar" expression level or trend under a specific subset of experimental conditions.

- *F. Divina is with the Computational Linguistics and AI Section, Tilburg University, Tilburg, The Netherlands. E-mail: F.Divina@uvt.nl.*
- *J.S. Aguilar-Ruiz is with the Polytechnic, Pablo de Olavide University, Seville, Spain. E-mail: jsagurui@upo.es.*

Manuscript received 10 Aug. 2005; revised 3 Dec. 2005; accepted 15 Dec. 2005; published online 17 Mar. 2006.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0307-0805.

Clustering is an important task within Knowledge Discovery in Databases (KDD), which aims to organize the information in terms of their similarity patterns. The problem of finding a partition of a set of objects into k groups which optimizes a stated criterion of partition adequacy is not, in general, straightforward. Given n examples, the number of ways in which these examples can be partitioned into k nonempty subsets is:

$$P(n,k) = \frac{1}{k!}\sum_{j=0}^{k}\binom{k}{j}(-1)^j(k-j)^n.$$

An approximation to the above equation is:

$$P(n,k) \approx \frac{k^n}{k!} \approx k^{n-k}e^k\sqrt{2\pi k}.$$

Therefore, when we do not know a priori the number of clusters k, the total number of evaluations is:

$$T(n) = \sum_{k=1}^{n} P(n,k).$$

For example, for $n = 8$, $T(8) = 4,140$. This gives an idea of the complexity of clustering.

Biclustering was first introduced by Hartigan [3], as a way to cluster simultaneously rows and columns of a matrix, and it was named "direct clustering." The goal was to find biclusters with minimum variance, that ideally provided biclusters of size 1, since they looked for constant biclusters (constant values within the submatrix). Hartigan tried to avoid this problem by searching for k biclusters at a time. Later, in 2000, Cheng and Church [4] proposed the biclustering of gene expression data, introducing the *residue* of an element in the bicluster and the *mean squared residue* of a submatrix. In addition, they adjusted that measure to reject trivial biclusters by means of the *row variance*. Getz et al. [5] presented the coupled two-way clustering. It uses hierarchical clustering applied separately to each dimension

and then they defined the process to combine both results. Obviously, the quality of biclusters depends on the clusters generated at each dimension, which in turn, allow us to experiment with different types of clustering algorithms. Lazzeroni and Owen [6] used "plaid models" in the same context, where the concept of "layer" (bicluster) is used to compute the values in the data matrix, which is described as a linear function of layers. Basically, each element is seen as a superposition of layers. Yang et al. [7] presented δ-clusters, and a year later, the same authors improved the Cheng and Church's approach in FLOC [8], paying attention to missing values. FLOC follows the same technique as Cheng and Church's algorithm, by adding/removing each row/column to a set of initial biclusters, improving its quality iteratively. Also, in 2002, Tanay et al. [9] identified biclusters by means of a bipartite graph-based model and using a greedy approach to add/remove vertexes in order to find maximum weight subgraphs, which are related to its statistical significance.

Another approach is pattern-based clustering, that captures the similarity of the patterns exhibited by a bicluster. In general, given a set of objects, a subset of these objects form a pattern-based cluster if these objects follow a similar pattern in a subset of dimensions. Wang et al. [10] proposed a depth-first algorithm for detecting pattern-based clusters. In order to speed up the process and to avoid the repetition of computations, the algorithm uses a suffix tree to efficiently enumerate the possible combinations of row and column sets that represent a bicluster. Liu and Wang [11] also proposed an exhaustive bicluster enumeration algorithm, which is based on a model that generalizes the order preserving submatrix model [12]. The objective of finding all biclusters that, after column reordering, represent coherent evolutions of the symbols in the matrix is achieved by using a pattern discovery algorithm inspired in sequential pattern mining algorithms [13]. Another algorithm that uses the pattern-based clustering model is proposed in [14]. This algorithm mines only the maximal pattern-based clusters. It conducts a depth-first, divide and conquer search and prunes unnecessary branches smartly.

The biclustering problem is even more difficult than clustering, as we tried to find clusters using two dimensions, instead of one. In fact, the problem of finding a minimum set of biclusters, either mutually exclusive or overlapped, is a generalization of another problem related to covering bipartite graph, which has been shown to be NP-hard [15]. Therefore, the cost of exploring exhaustively all the possible partitions of the search space into nonoverlapped biclusters would be $T(N) \times T(M)$, where N is the number of genes and M is the number of conditions in the data set.

In this work, we address the biclustering problem with evolutionary computation, which has been proven to have an excellent performance on highly complex optimization problems.

This approach is motivated by two major characteristics of evolutionary algorithms: their excellent exploration power, that gives them the possibility of escaping from local optima and their ability to work well when solutions to a problem contain complex interacting parts, where the impact of each part on the overall solution may be difficult to model [16]. This is true in biclustering, where the genes and conditions that are included in the biclusters interact with each other in order to determine the quality of the bicluster. Moreover, evolutionary computation provides a searching method motivated by an analogy with biological evolution, which is known to be a successful, robust method for adaptation within biological systems.

Our approach, named `SEBI` (for Sequential Evolutionary BIclustering) is based on evolutionary algorithms and searches for biclusters following a sequential covering strategy. As the algorithm partially uses the *squared mean residue*, the results have been compared to those of Cheng and Church. In expression data analysis, the most important goal may not be finding the maximum bicluster or even finding a bicluster covering for the data matrix. It is more interesting to find a set of genes showing strikingly similar up-regulation and down-regulation under a set of conditions. A low mean squared residue score plus a large variation from the constant may be a good criterion for identifying these genes and conditions. Therefore, our goal is to find biclusters of maximum size, with mean squared residue lower than a given δ, with a relatively high row variance, and with a low level of overlapping among biclusters.

Among the biclustering algorithms that perform a greedy search, also FLOC performs some stochastic steps. In fact, FLOC first determines which set of actions will lead to the highest gain. The actions are then applied with a probability proportional to their associated gain. `SEBI` does not use any greedy search phase; the variation operators it uses are completely stochastic. The search in `SEBI` is lead by the fitness function, which is computed only after the application of the variation operators. Moreover, FLOC finds all the biclusters simultaneously, while `SEBI` adopts a sequential covering strategy. FLOC is also included in the experimental results section.

In [17], an evolutionary algorithm (EA) has been used for finding biclusters in gene expression data. The same authors in [18] use the order preserving submatrix method by Ben-Dor et al. [12] inside the EA.

The paper is organized as follows: In Section 2, the definitions related to biclustering are presented. An introduction to Evolutionary Computation is given in Section 3. The description of the algorithm is illustrated in Section 4, together with all the evolutionary features and the evaluation of the quality of a bicluster. Experimental results are discussed in Section 5, comparing the quality to those generated by the algorithms presented by Cheng and Church, and Yang et al. Finally, the most interesting conclusions are summarized in Section 6.

2 The Model of Biclusters

In this section, we present the model of bicluster, and a way for assessing the quality of a bicluster. We follow the biclustering model proposed in [4].

A bicluster is defined on a gene-expression matrix. Let $G = \{g_1, \ldots, g_N\}$ be a set of genes and $C = \{c_1, \ldots, c_M\}$ a set of conditions. The data can be viewed as an $N \times M$ expression matrix EM. EM is a matrix of real numbers, with possible

	c_j							
g_i	9	**(1)**	10	**(1)**	5	7	**(3)**	8
	0	2	3	9	5	3	2	5
	3	**(2)**	6	**(1)**	8	6	**(1)**	2
	3	8	3	4	1	5	3	1
	4	**(1)**	3	**(2)**	2	5	**(2)**	2
	5	7	2	4	2	6	7	2
	11	3	2	7	3	8	4	3
	4	2	12	5	7	0	4	6
	4	2	6	4	2	7	8	2
	3	7	3	7	5	2	4	6

Fig. 1. Expression matrix for Example 1. Each row represents a gene g_i and each column represents a condition c_j. Elements belonging to the bicluster given in Example 1 are between brackets and highlighted.

null values, where each entry e_{ij} corresponds to the logarithm of the relative abundance of the mRNA of a gene g_i under a specific condition c_j. The logarithmic transformation is used to convert doubling or other multiplicative changes of the relative abundance into additive increments.

A bicluster essentially corresponds to a submatrix that exhibits some coherent tendency. Each bicluster can be identified by a unique set of genes and conditions, that determine the submatrix. Thus, a bicluster is a matrix $I \times J$, denoted as (I, J), where I and J are a set of genes (rows) and conditions (columns), respectively, and $|I| \leq |N|$ and $|J| \leq |M|$. We define the volume of a bicluster (I, J) as the number of elements e_{ij} such that $i \in I$ and $j \in J$.

Example 1. Suppose the expression matrix EM consists of 10 genes and eight conditions, like the one represented in Fig. 1, where the genes represent the rows of the matrix and the conditions represent the columns of the matrix. Then, a bicluster defined over the expression matrix EM could be $(\{1, 3, 5\}, \{2, 4, 7\})$, thus consisting of genes g_1, g_3, g_5 and of conditions c_2, c_4, c_7. The volume of this bicluster is 9. In Fig. 1, the elements belonging to the bicluster are highlighted and between brackets.

In the following, we give some definitions related to the measure used here for assessing the quality of a bicluster, most of which are taken from [8].

Definition 1. *Let (I, J) be a bicluster, then we define the base of a gene g_i as $e_{iJ} = \frac{\sum_{j \in J} e_{ij}}{|J|}$. In the same way, we define the base of a condition c_j as $e_{Ij} = \frac{\sum_{i \in I} e_{ij}}{|I|}$. The base of a bicluster is the mean of all the entries contained in (I, J), $e_{IJ} = \frac{\sum_{i \in I, j \in J} e_{ij}}{|I| \cdot |J|}$.*

Note that in the above definition, e_{iJ} and e_{Ij} corresponds to the mean of the ith row and of the jth column of the bicluster (I, J), respectively. In order to quantify the difference between the actual value of an entry and the expected value of an entry predicted from the corresponding gene base, condition base, and the bicluster base, we introduce the concept of *residue*.

Definition 2. *The residue of an entry e_{ij} of a bicluster (I, J) is $r_{ij} = e_{ij} - e_{iJ} - e_{Ij} + e_{IJ}$.*

The residue is an indicator of the degree of coherence of an element with respect to the remaining ones in the bicluster, given the tendency of the relevant gene and the relevant condition. The lower the residue, the stronger the coherence. To assess the quality of a bicluster, the residue of the bicluster can be defined as the squared mean residue of all specified elements, as in [4].

Definition 3. *The mean squared residue of a bicluster (I, J) is $r_{IJ} = \frac{\sum_{i \in I, j \in J} r_{ij}^2}{|I| \cdot |J|}$.*

The mean squared residue is the variance of the set of all elements in the bicluster, plus the mean row variance and the mean column variance. The lower the mean squared residue, the stronger the coherence exhibited by the bicluster, and the better the quality of the bicluster. If a bicluster has a mean squared residue lower than a given value δ, then we call the bicluster a δ-bicluster. The problem of finding the largest square δ-biclusters is NP-hard [4]. In addition to the mean squared residue, we may prefer the row variance to be relatively large to reject trivial bicluster.

Definition 4. *Let (I, J) be a bicluster. The* row variance *of (I, J) is defined as $var_{I,J} = \frac{\sum_{i \in I, j \in J} (e_{ij} - e_{iJ})^2}{|I| \cdot |J|}$.*

By using the row variance as an accompanying score, we want to guarantee that the bicluster captures genes exhibiting fluctuating yet coherent trends under some set of conditions.

Our goal is to find biclusters of maximum size, with mean squared residue lower than a given δ, with a relatively high row variance, and with a low level of overlapping among biclusters.

3 Evolutionary Computation

In this section, we provide the reader with the basic principles of Evolutionary Computation (EC). For a detailed introduction to EC, the reader can refer to [16], [19], [20].

EC is a population-based stochastic iterative optimization technique based on the Darwinian concepts of evolution. Inspired by these principles, like survival of the fittest and selective pressure, EC tackles difficult problems by evolving approximate solutions of an optimization problem inside a computer. An algorithm based on EC is called an Evolutionary Algorithm (EA).

Given an optimization problem, all EAs typically start from a set, called population, of random (candidate) solutions. These solutions are evolved by the repeated selection and variations of more fit solutions, following the principle of the survival of the fittest. We refer to the elements of the population as individuals or as chromosomes, which represent candidate solutions. Solutions can be encoded in many different ways. A typical example is represented by binary string encoding, where each bit of the string has a particular meaning. In general, with the term phenotype we refer to an object forming a possible solution within the original context, while its encoding is called genotype. To each genotype must correspond at most one

```
Sequential Covering(δ)
load Expression Matrix EM
repeat
    EBI(EM, δ)
    if (bicluster returned)
        store bicluster in Results
        adjust weights of EM
    else
        end_cond met
    if (max_iter is reached)
        end_cond met
until end_cond is met
return Results
```

(a)

```
EBI(Expression Matrix EM, δ)
initialize Population
evaluate Population
repeat
    select parents
    recombine pairs of parents
    mutate the resulting offspring
    evaluate new individuals
    select individuals for next generation
until max_iter is reached
best_ind = best individual in Population
if (residue(best_ind) < δ)
    return best individual in Population
```

(b)

Fig. 2. A general scheme of both the (a) sequential covering algorithm and of the (b) procedure EBI. In (b), $residue(X)$ returns the mean squared residue of a bicluster encoded in the individual X.

phenotype, so that the chosen encoding can be inverted, i.e., genotypes can be decoded.

Individuals are typically selected according to the quality of the solution they represent. To measure the quality of a solution, a fitness function is assigned to each individual of the population. Hence, 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.

The selected individuals reproduce by means of crossover and mutation. In simple terms, crossover swaps some genetic material between two or more individuals, while mutation changes a small part of the genetic material of an individual to a new random value. From the reproduction phase, new offspring are generated. Offspring compete with the old individuals for a place in the next generation. If the best individual is always allowed to survive to the next generation, we say that *elitism* is applied.

In this way, EAs can efficiently explore the space of possible solutions of an optimization problem. This space is called search space, and it contains all the possible solutions that can be encoded. EAs have been shown to be efficient in searching in huge spaces. The stochastic operators used allow EAs to search for possible solution in an efficient way. For these reasons, EAs represent a valid alternative to greedy heuristic.

EC has been applied to find solutions of problems in a variety of domains, e.g., planning [21], design [22], scheduling [23], [24], simulation and identification [25], control [26], and classification [27], [28], [29], [30].

The problem of finding a set of biclusters with some desirable features on an expression matrix can be seen as a search problem in the space consisting of all the possible biclusters than can be obtained from the expression matrix. In the introduction, we showed the complexity of this problem. For EAs adopting a binary encoding, this problem has a huge search space, and it is not easy as it might seem at first sight. This aspect will be analyzed in Section 4.1.

4 Description of the Algorithm

The algorithm adopts a sequential covering strategy: a procedure called EBI (Evolutionary BIclustering) is called several times, until an *end condition* is met. EBI takes as input the expression matrix and the δ value, as a threshold for the mean squared residue. EBI returns either a bicluster with mean squared residue lower than δ or nothing. The returned bicluster is stored in a list called *Results*, and EBI is called again. The end condition is also met when EBI is called a maximum number of times. When the end condition is met, the list *Results* is returned. A general scheme of the sequential covering algorithm is given in Fig. 2a.

In the scheme of Fig. 2a, after that EBI returns a bicluster, weights associated with the expression matrix are adjusted in order to avoid overlapping among biclusters as much as possible. The weight of an element in the expression matrix depends on the number of biclusters in *Results* containing the element. The more biclusters cover an element, the higher the weight of the element will be. The weight w_p associated to an element e_{ij} of the expression matrix is:

$$w_p(e_{ij}) = \begin{cases} 0 & \text{if } |Cov(e_{ij})| = 0; \\ \frac{\sum_{n \in N, m \in M} e^{-|Cov(e_{nm})|}}{e^{-|Cov(e_{ij})|}} & \text{if } |Cov(e_{ij})| > 0, \end{cases} \quad (1)$$

where N and M are the number of rows and the number of columns of the expression matrix, respectively, and $|Cov(e_{ij})|$ is the number of biclusters in *Results* containing e_{ij}.

In EBI, the weights are taken into consideration when the quality of biclusters is assessed, as described in Section 4.2. In this way, we want to bias the search toward biclusters that do not overlap with already found biclusters. $w_p(e_{ij})$ is the inverse of the weight used in the EWUS

selection operator [31], where the weights are assigned to training examples inside a classifier, in order to bias the search toward a subset of training examples.

EBI implements an EA, whose scheme is illustrated in Fig. 2b. The aim of EBI is to find δ-biclusters with maximum volume, with a relatively high row variance, and minimizing the effect of overlapping among biclusters.

The first operations performed by EBI are the initialization and the evaluation of the population.

Initialization of the population is an important aspect. Typically, in GAs, the population is randomly initialized. In our opinion, this is not a good strategy to be used in SEBI. In fact, in this case, initial individuals will have a similar dimension, since each bit has the same probability of being set to one. Moreover, there would be no guarantee to obtain biclusters that have a low mean squared residue. These two aspects would then render the genetic search less efficient, since the initial biclusters will have a high and similar dimensionality and probably low quality. We decided to have initial individuals representing biclusters of dimension one because these biclusters have a mean squared residue equal to zero. In this way, the design of variation operators for growing biclusters was straightforward. However, other initialization strategies may be adopted. For example, the population may be initialized with bicluster found by a greedy search strategy.

In the loop, a population of biclusters is evolved by means of the repeated application of selection, crossover, and mutation. A number of individuals are selected to become parents. Selected pairs of parents are recombined by a crossover operator with a given probability p_c (default value 0.85), and the resulting offspring is mutated with a probability p_m (default value 0.2). The process is repeated with the new generation of offspring, until a maximum number of generations has been reached. Elitism is applied with a probability p_e (default value 0.9). At the end of the evolutionary process, if the best individual encodes a δ-bicluster, then it is returned; otherwise, EBI does not return anything. The best individual is the one having the best fitness in the final population.

In this paper, we use the default values of SEBI for the parameter settings that control the evolutionary process, i.e., population size, number of generation, crossover, and mutation probabilities. The two data sets used have different characteristics, and, nevertheless, SEBI obtained good results with the default parameter setting.

In the following, we address various aspects of the EA implemented in EBI. In particular, we describe how biclusters are encoded into individuals, the fitness function, and the genetic operators adopted for selecting parents and producing offspring. We conclude this section with some considerations about the parameter setting of SEBI (the sequential covering strategy that includes the evolutionary algorithm EBI).

4.1 Encoding of Biclusters

Each individual of the population encodes one bicluster. Biclusters are encoded by means of binary strings of length $N + M$, where N and M are the number of rows (genes) and of columns (conditions) of the expression matrix, respectively. Each of the first N bits of the binary string is related to the rows, in the order in which the bits appear in the string. In the same way, the remaining M bits are related to the columns. If a bit is set to 1, it means that the relative row or column belongs to the encoded bicluster; otherwise, it does not. This encoding presents the advantage of having a fixed size, thus allowing the use of standard variation operators.

TABLE 1
Length of Individuals in the Evolutionary Population and Corresponding Size of the Search Space for the Yeast and Human Data Sets

Dataset	Length of individual	Search Space Size
Yeast	2884+17=2901	$2^{2901} \approx 10^{873}$
Human	4026+96=4122	$2^{4122} \approx 10^{1240}$

Example 2. If we want to encode the bicluster proposed in Example 1, the binary string consists of 18 bits (10 for the genes and eight for the conditions). The genotype of the bicluster will then be:

$$1010100000|01010010,$$

where the symbol | is only used for delimiting the bits relative to the rows from the bits relative to the columns.

Note that the initial population consists of biclusters containing only one element. The genotype of such biclusters is a binary string where only two bits are set to 1, one for the row and one for the column belonging to the bicluster. In the initialization phase, these two bits are randomly chosen. The remaining bits are set to 0.

As it was pointed out earlier, the complexity of the search space is very high, as the size of the search space has an exponential relationship with the length of the individuals. For an individual of length L, the size of the search space is 2^L, that is, $10^{L*lg_{10}2}$. For the two data sets under study in this paper, the size of the search space is amazingly huge (see Table 1), and the search for solutions is a very hard process even for an EA.

4.2 Fitness Function

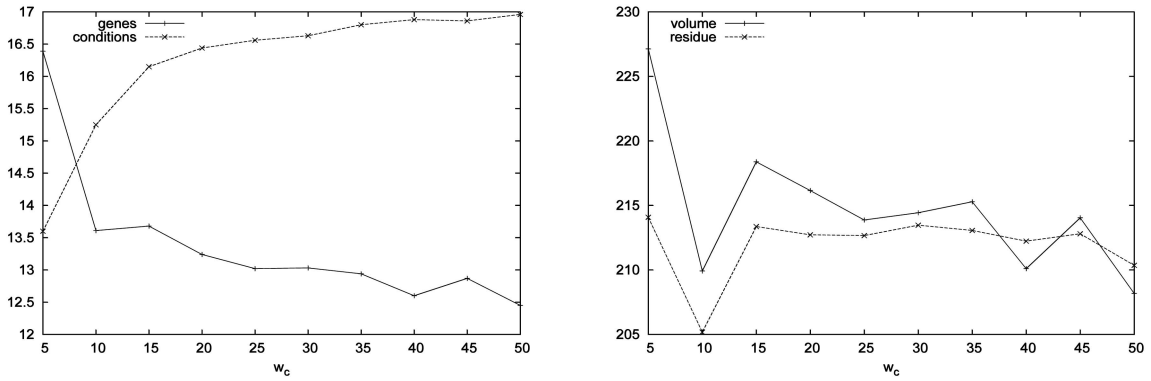
The fitness function rewards individuals encoding biclusters with low mean squared residue, with high volume and row variance, and covering elements of the expression matrix that are not covered by biclusters found by previous executions of EBI. To this aim, the following fitness function is used:

$$f(X) = \frac{residue(X)}{\delta} + \frac{1}{row\_variance(X)} + w\_d + penalty. \tag{2}$$

In the above formula, X is an individual, $residue(X)$ is the mean squared residue of the encoded bicluster, $row\_variance(X)$ is the row variance of X.

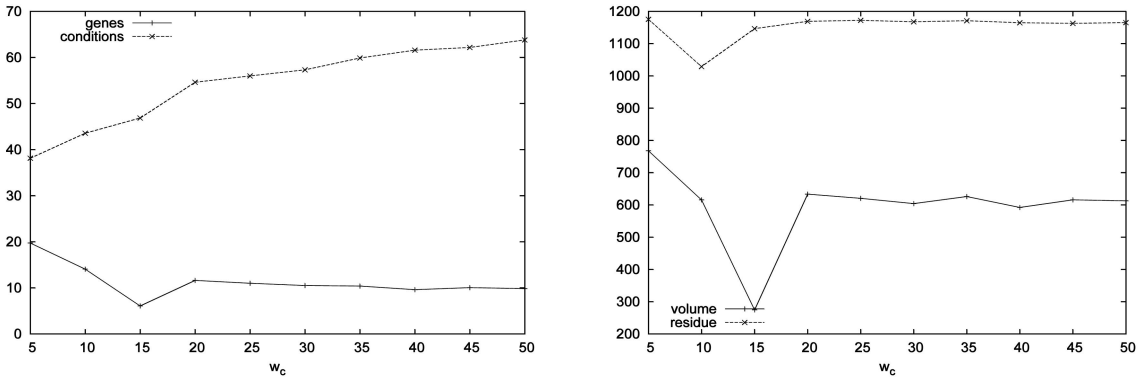
$$penalty = \sum_{i \in I, j \in J} w_p(e_{ij}),$$

where I, J are the rows and columns belonging to bicluster, respectively, and w_p is defined in (1). The use of $penalty$



(a) (b)

Fig. 3. Influence of w_c on the residue, volume, and the number of genes and conditions for the 100 biclusters obtained from the yeast data set ($w_r = 1.$). (a) Influence on the number of genes and conditions. (b) Influence on the residue and volume.



(a) (b)

Fig. 4. Influence of w_c on the residue, volume, and the number of genes and conditions for the 100 biclusters obtained from the human data set ($w_r = 1.$). (a) Influence on the number of genes and conditions. (b) Influence on the residue and volume.

allows us to avoid overlapping among biclusters. w_d is equal to $w_V \cdot (w_r \cdot \frac{\delta}{row_X} + w_c \cdot \frac{\delta}{col_X})$, where w_V is a weight used for giving more or less importance to the volume of the bicluster (default value is 1), row_X and col_X are the number of rows and columns of the encoded bicluster, respectively, and w_r, w_c are weights assigned to the number of rows and to the number of columns, respectively. This expression was empirically obtained as a way to balance the influence of the large number of genes in comparison to the small number of conditions in the fitness function. By varying the values of w_V, w_r, and w_c, we can bias the genetic search in such a way to prefer biclusters having great volume, involving more or less genes or conditions. The influence of the factor w_d (and particularly of w_r and w_c) on the results is analyzed in Section 4.4, and particularly in Figs. 3 and 4.

If $residue(X) > \delta$, then the value of $\frac{residue(X)}{\delta}$ is greater than 1; otherwise, it is less than 1. If the $row\_variance(X)$ is much greater than 1, then the factor $\frac{1}{1+row\_variance(X)}$ becomes very small. In this way, the smaller the residue and the larger the row variance are, the smaller the fitness value, i.e., the better the quality of that bicluster is.

The final objective of the algorithm EBI is to minimize the fitness.

4.3 Genetic Operators

Individuals are selected for reproduction with a tournament selection operator. A number of individuals are randomly selected, and the one with lower fitness is chosen as *elite*. The number of individuals selected determines the size of the tournament. The size of the tournament can be supplied by the user, and its default value is 2. Elitism is applied with a given probability (default value 0.9).

Three crossover operators are used: one-point, two-point, and uniform crossover. In short, one-point crossover selects a point inside the two strings and produces the offspring by exchanging the substrings of the parents. Two-point crossover works in the same way, but selects two points inside the two strings. Uniform crossover combines bits sampled uniformly from the two parents. If crossover has to be applied, one of the three operators is applied with equal probability.

The crossover operators produce biclusters containing rows and columns that were present in the parents, which represented good biclusters at a given point of the evolutionary process.

The basic principle behind crossover is simple: By mating two individual encoding biclusters with different but desirable features, we can produce a bicluster that combines both of those features. In other words, if crossover

is applied to two biclusters containing rows and columns identifying two good biclusters, columns and rows may be combined in such a way that a better bicluster is obtained. This new bicluster will contain parts of both parents. It is accepted that some offspring created by crossover may have undesirable combinations of traits. In this case, they will have a low probability of being selected for reproduction, and to survive to next generations. With the use of crossover, diversity is maintained in the population, so that more biclusters can be searched.

Three mutation operators are employed: standard mutation operator, a mutation operator that adds a row, and a mutation that adds a column to the bicluster. The last two operators are employed for giving more possibility of growing to biclusters when mutated.

Mutation is a highly random operation, and for this reason it is applied with a low probability. In particular, with the last two mutation operators, we want to give more possibilities to biclusters to expand. It is not guaranteed that the addition of a row or of a column will be beneficial for the bicluster, e.g., its mean square residue may increase. However, there is the possibility that a random mutation could lead to the discovery of a good bicluster. If this is not the case, then the bicluster has few possibilities of being selected for reproduction, and transmitting in this way part of its genetic material to next generations. Mutation operators guarantee that the space of the biclusters is connected, i.e., that all the biclusters can be reached during the search. Thus, the use of mutation is important because it allows to explore more extensively the space of all the biclusters that can be obtained from a given expression matrix.

4.4 Parameters Setting

As most EAs, SEBI uses several parameters that control the genetic search, e.g., crossover and mutation probabilities, population size, etc. In general, for an EA to perform well, the issue of setting the parameter values used to guide the genetic search is critical.

Typically, in order to determine a good parameter setting for an EA on a given problem, a number of preliminary runs are performed with different parameter settings. This could be a time consuming operation, and is not guaranteed to lead to the optimal parameter setting, since the number of possible combination of parameters values is very high, and many runs with different random seeds should be performed.

In this paper, we have used the standard values for the EA, which are shown in Table 2. However, it is necessary to balance the effect of the great difference between the number of genes and conditions. As this ratio is not the same for different data sets, in principle the EA might not have good performance for inappropriate choices of w_r and w_c. As we show in Figs. 3 and 4, the influence of an incorrect setting for w_r and w_c is not very relevant. In Fig. 3b, the best value for w_c is 10. However, as it is shown in Fig. 4b, w_c could be in $[10, 15]$. We preferred biclusters with small number of genes (generally, small volume) and low mean squared residue, so the value $w_c = 10$ fulfills this requirement.

TABLE 2
Parameter Values of SEBI

Parameter	Value
Population size	200
Number of generations	100
Crossover probability	0.85
Mutation probability	0.20
Elitism probability	0.90
Weight for conditions	10
Weight for genes	1

5 Experimental Results

In order to assess the quality of the proposed method for finding biclusters in expression data, we conducted experiments on two well-known data sets. The first data set is the yeast *Saccharomyces cerevisiae* cell cycle expression data set originated from [32]. The expression matrix contained in this data set consists of 2,884 genes and 17 experimental conditions. The second data set is the human B-cells expression data originated from [33]. The human data set consists of an expression matrix of 4,026 genes and 96 conditions. The two data sets are taken from [4], where the original data is preprocessed, replacing missing values with random values. However, it is known the existing risk that these random numbers can affect the discovery of biclusters [7].

The values of δ for the two data sets are taken from [4], which are calculated from the clustering experiments done in [34]. For the yeast data set δ was set to 300, and for the human data set to $1,200$.

All the experiments were performed using the same parameter setting of the EA (Table 2). When the search space size is huge, the choice of a correct set of parameter values is very important, and might need some trials. However, what is truly critical is the definition of a precise fitness function, as the parameter values will only help to reach the goal defined by that function, i.e., the evolutionary algorithm might need much more time to converge, and the results might not be so good. In SEBI, the size of the population is 200 and the number of generations is 100. These are very tiny values for the parameters taking into account the size of the search space illustrated in Table 1. It is important to note that every time EBI is called, it runs 20,000 evaluations of potential biclusters, and the size of the search space is about 10^{1000}. Thus, the number of candidate biclusters evaluated by EBI is ridiculous in comparison to the number of potential biclusters. Therefore, the role of genetic operators is important to correctly guide the search towards good solutions.

5.1 Yeast Data Set

In Fig. 5, 12 out of 100 biclusters found by EBI on the yeast data set are shown. The first bicluster shown, labeled 1_1 is the bicluster found with the first call of EBI. As in [4], biclusters like this one need to be discovered, then more "interesting" biclusters may emerge. From a visual inspection of the other

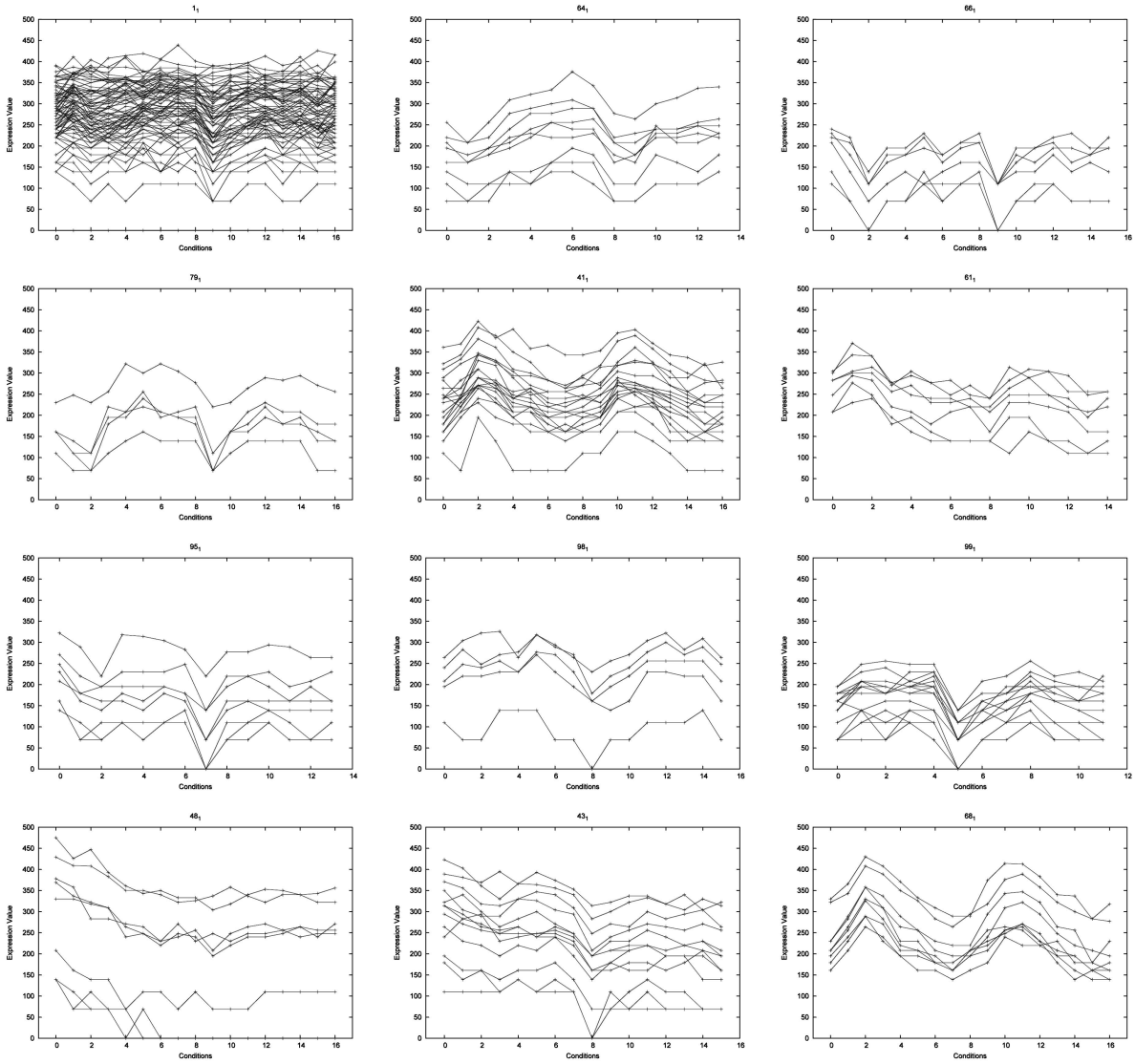


Fig. 5. Twelve biclusters found for the yeast data set. All the mean squared residues of the biclusters are lower than 220.

biclusters proposed in Fig. 5, one can notice that the genes present a similar behavior under a set of conditions. This is especially evident in the bicluster labeled 68_1, where the variance is very high. Many biclusters found on the yeast data set contains all 17 conditions, indicating that these conditions form a good cluster, with respect to the genes included in the biclusters. Of the 12 biclusters shown in Fig. 5, five contain all 17 conditions. A similar result was also obtained in [4]. In general, the evolutionary technique performs very well, finding several groups of genes with the same behavior although shifted. For instance, the bicluster 48_1 contains three groups of genes, two the bicluster 64_1 and other two the bicluster 98_1. Generally, all the biclusters but the first one describe two or more groups of genes along the conditions, which vary from 13 to 17. Biclusters 41_1 and 98_1 are interesting because they differentiate one gene from the others, although all the genes seem to have the same trend. SEBI shows an excellent performance at finding shifting patterns (see biclusters 48_1 or 68_1) and scaling patterns (see biclusters 41_1 and 43_1).

Information about these biclusters is given in Table 3. All the biclusters have residues less than 300. In fact, they are less than 220, so the evolutionary approach is very precise at adjusting this value. In addition, the row variance is considerably high for all the biclusters except the first one, which includes 82 genes. Bicluster 68_1 is the biclusters characterized by the highest row variance. It is interesting to notice that this is also the bicluster with the lowest residue.

In order to illustrate the interesting performance of our approach, the 100 biclusters obtained by the EA are represented in Fig. 6. Biclusters are enumerated on the X-axis, while in the Y-axis the value of the mean squared residue and the volume are reported. The EA is very stable at obtaining biclusters with a prefixed value for the mean squared residue, under the value of 300, during all the runs. However, it is very conservative at producing biclusters with great volume, since the averaged size of the biclusters is decreasing along the process, becoming almost stabilized at the end.

In Fig. 7, three graphs relative to a typical run of EBI on the yeast data set are shown. We show the run relative to a

TABLE 3
Information about Biclusters of Fig. 5

Bicluster	Rows	Columns	Residue	Row Variance
1_1	82	17	215.29	317.23
64_1	9	14	200.047	1049.88
66_1	7	16	213.98	1240.12
79_1	5	17	215.46	1476.49
41_1	20	17	205.27	1074.67
61_1	7	15	217.74	1343.32
95_1	8	14	219.97	1037.63
98_1	5	16	214.03	1062.62
99_1	18	12	219.54	1101.94
48_1	9	17	213.73	1629.94
43_1	13	16	207.99	1360.68
68_1	9	17	192.26	1995.21

In the first column, the identifier of each bicluster is reported. The second and third columns report the number of rows (genes) and of columns (conditions) of the bicluster, respectively, the fourth column reports the mean squared residues, and the last column reports the row variance of the biclusters.

first call of EBI because in this run the weights associated to the elements of the expression matrix are all equal to zero. In this way, the fitness is of easier interpretation. In fact, in the first call of EBI, only the residue, the row variance, and the volume of the bicluster are considered in the fitness function. In Fig. 7a, the average fitness and the best fitness present in the population at each generation are shown. It can be noticed that the fitness decreases rapidly in the first generations, until about the 23rd generation, and then it keeps decreasing, although more slowly. This means that the evolutionary algorithm converges quickly to a local optimum. In Fig. 7b, the average volume and the best volume of the biclusters encoded in the population are given at each generation. We can notice that the average volume of the biclusters increases constantly, meaning that the fitness function is successful in promoting biclusters with greater volume. The best volume also increases constantly on average. In Fig. 7c, the average mean squared residue and the lowest residue are given at each generation. In the first generations, the average residue increases. This is due to the fact that in the first generations, the biclusters are very small, and when growing, their residues increase. After some generations, the residue becomes almost stable, with a value close to 200. The residue does not decrease because the fitness function gives more importance to the volume of a bicluster when its mean squared residue is lower than δ. Until about the 20th generation, the best mean squared residue is 0. This is due to the fact that until that generation some bicluster containing only one element are still present in the population.

The 100 biclusters obtained by SEBI cover 38.14 percent of the elements of the expression matrix, 43.55 percent of the genes, and 100 percent of the conditions. In [4], the 100 biclusters covered 2,801, or 97.12 percent of the genes, and 100 percent of the conditions, and 81.47 percent of the cells in the matrix. These results confirm the effectiveness of the adopted method for avoiding overlapping. Each call to EBI on the yeast data set requires about 70 seconds on a Pentium IV 3GHz.

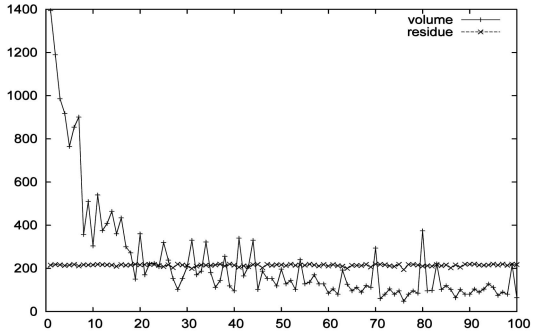


Fig. 6. Evolution of the mean squared residue and the volume for the 100 biclusters obtained by SEBI on the yeast data set.

5.2 Human Data Set

Fig. 8 shows 12 out of 100 biclusters found for the human data set (Table 4 collects information for these biclusters).

The first bicluster contains 37 genes and 56 conditions. The second and third biclusters are still very general, with 18×59 and 14×64 genes and conditions, respectively. From bicluster 50_1, we find a more specific and interesting group of genes, the trend of which is more defined and the row variance is high. Bicluster 89_1 presents a high row variance for six genes and 49 conditions, the same as bicluster 97_1, although in this case with 62 conditions and the highest row variance. The most interesting bicluster is the one labeled 89_1. This bicluster contains six genes showing strikingly similar behavior under 49 conditions.

The 100 biclusters obtained by SEBI are represented in Fig. 9. As in Fig. 6, the EA is very stable at obtaining biclusters with a prefixed value of the mean squared residue, under the value of 1,200, during all runs. However, the averaged size of the biclusters is decreasing along the process, stabilizing around 400. The value of w_c slightly varies this size.

In Fig. 10, three graphs relative to a typical run of the algorithm on the human data set are shown. For the same reasons explained for the yeast data set, we show the run relative to a first call of EBI. In Fig. 10a, at each generation the average and best fitness are shown. It can be seen that the fitness of the individuals decreases rapidly until about generation 40, and then it keeps on decreasing but more slowly. The situation is different as far as the volume of the bicluster is concerned, as can be seen in Fig. 10b. The average volume of the biclusters increases slowly until about the 25th generation, and after this point, it increases very rapidly. The best volume does not increase constantly, and sometimes it decreases. This is due to the fact that elitism is applied with a given probability, so in some generations, the best individual can be lost, and this can explain the decrements in the graph. The population evolves constantly even if after about 50 generations the individuals evolve more slowly. In Fig. 10c, the average mean squared residue and the lowest mean squared residue at each generation are

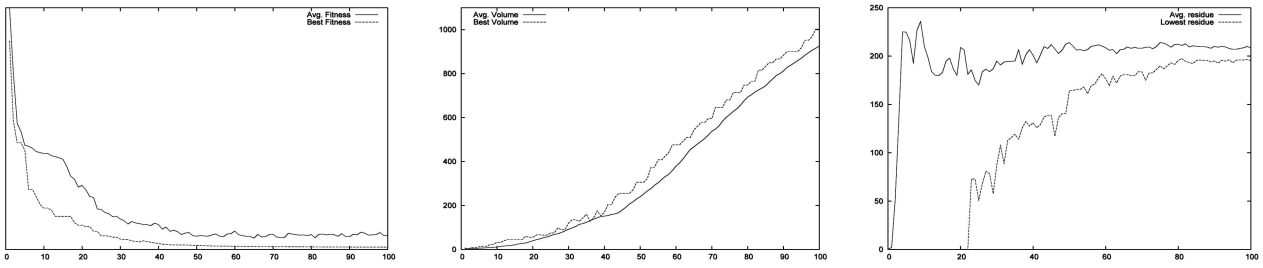


Fig. 7. Graphs relative to a typical run of EBI on the yeast data set. In (a), the average and best fitness at each generation are shown. In (b), the average and best volumes are shown for each generation. In (c), the average residue and the lowest residue at each generation are plotted.

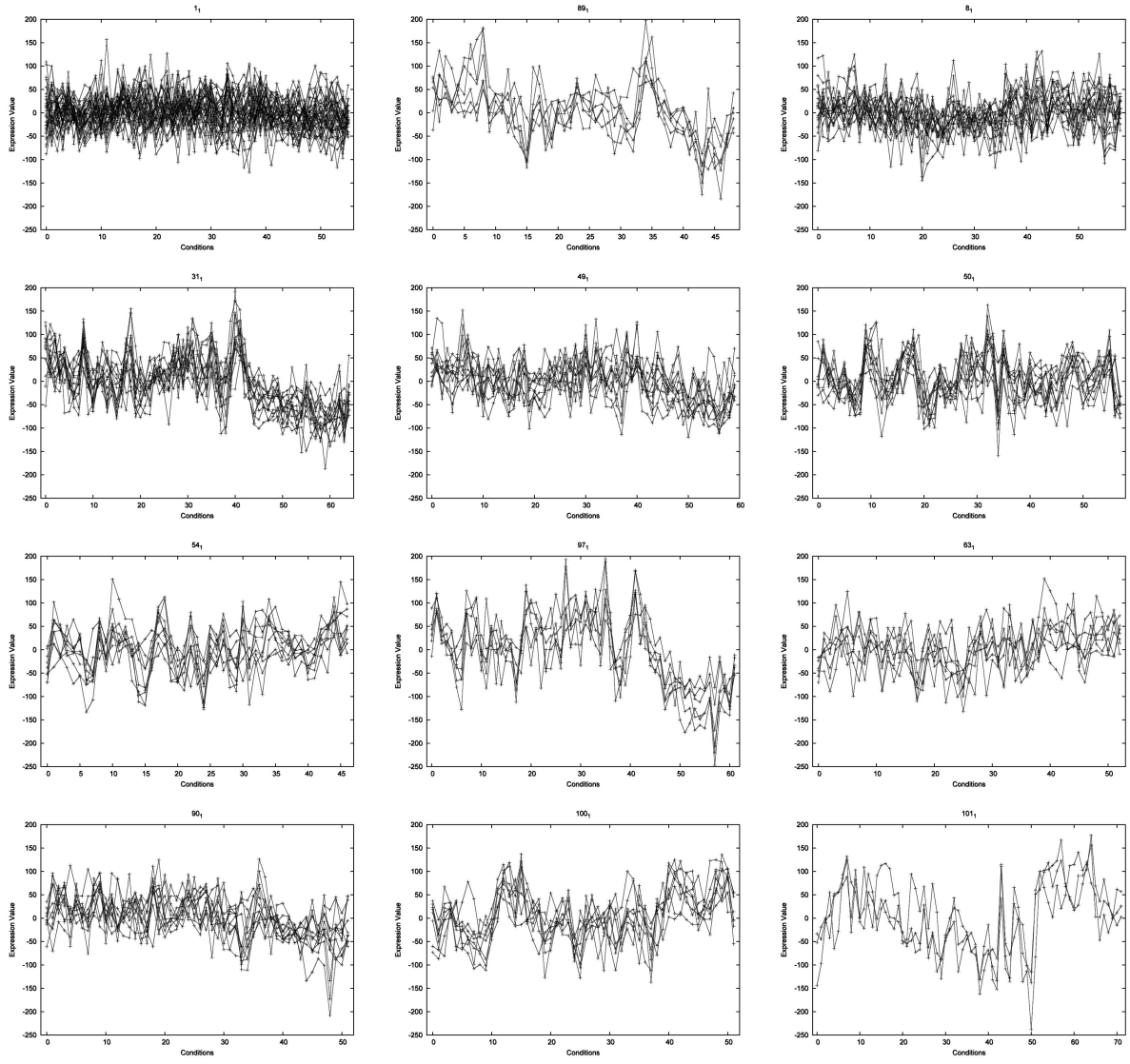


Fig. 8. Twelve biclusters found for the human data set. All the mean squared residues of the biclusters are lower than 1,200.

shown. Until about the 24th generation the lowest residue is 0. However, the average residue increases rapidly in the first eight generations, then decreasing quickly until about the 40th generation. After this point, it becomes almost stable, with a value close to δ. The lowest residue increases until it reaches a value close to 1,200, i.e., δ for this data set. This is due to the fact that in the fitness function, if the residue is lower than δ, then the volume of the bicluster has more importance. The evolutionary algorithm shows an excellent performance in this data set, as it controls the residue through generations, and produces biclusters with high row variance.

All the biclusters found on the human data set cover 34.07 percent of the elements of the expression matrix,

TABLE 4
Information about Biclusters of Fig. 8

Bicluster	Rows	Columns	Residue	Row Variance
1_1	37	56	1167.58	1272.55
89_1	6	49	1183.54	3529.10
8_1	18	59	1175.16	1538.76
31_1	14	65	1161.00	3228.62
49_1	11	60	1155.89	2002.09
50_1	11	58	1104.18	2159.42
54_1	9	47	1144.98	2274.42
55_1	12	49	1155.16	40637.10
63_1	7	53	1181.10	2082.99
90_1	11	52	1198.46	2015.91
100_1	8	52	1188.86	2843.53
101_1	3	72	1164.50	5691.07

In the first column, the identifier of each bicluster is reported. The second and third columns report the number of rows (genes) and of columns (conditions) of the bicluster, respectively, the fourth column shows the mean squared residue, and the last column reports the row variance of the biclusters.

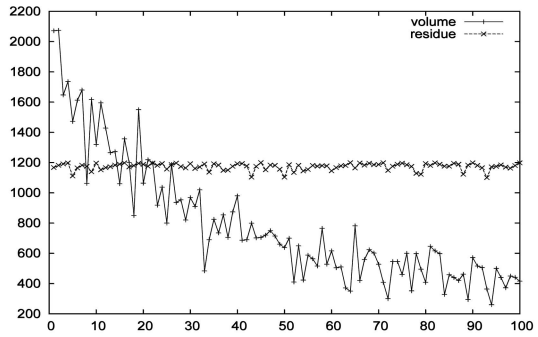


Fig. 9. Evolution of the mean squared residue and the volume for the 100 biclusters obtained sequentially by SEBI on the human data set.

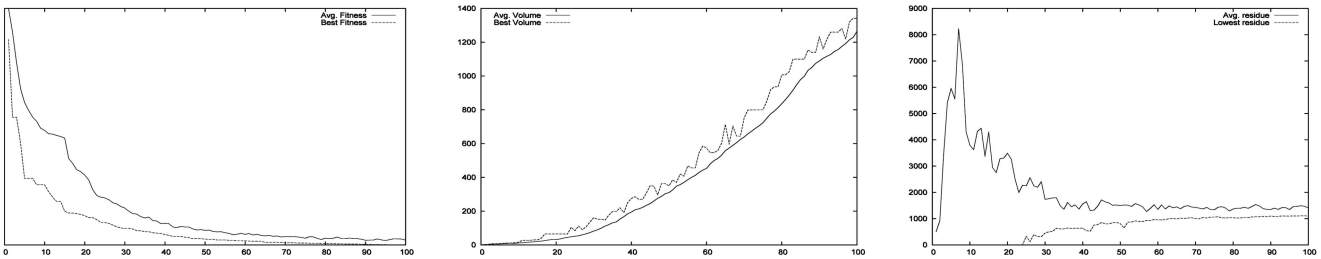
covering 38.23 percent of the genes and 100 percent of the conditions. In [4], the first 100 biclusters from the human data covered 3,687, or 91.59 percent of the genes, 100 percent of the conditions and 36.81 percent of the cells in the data matrix. As for the yeast data set, this result confirms the effectiveness of assigning weights to the elements of the expression matrix in order to avoid overlapping. Each call to EBI on the human data set requires about 200 seconds on a Pentium IV 3GHz.

5.3 Comparison

In Table 5, we compare the performance of SEBI, with that of Cheng and Church's algorithm (henceforth CC) and the algorithm FLOC by Yang et al. [35] for what concerns the average residue and the average dimension of the biclusters found. We can see that CC and FLOC are capable of finding biclusters characterized by a higher volume than the ones found by SEBI. This is probably due to the overlapping policy adopted by SEBI. In fact, the first biclusters found by SEBI have volumes comparable with those of the biclusters found by CC or FLOC. After some iterations, when the most trivial biclusters have been found, SEBI focuses on elements of the expression matrix that are not contained in already found biclusters. However, after CC has found a bicluster, the covered elements of the expression matrix are substituted by randomly generated values, in the range of the original data. This may cause the biclusters to overlap much more than in SEBI, where overlapping is avoided as much as possible. As far as the residue is concerned, the results obtained by the two systems are comparable on the yeast data set, while for the human data set CC, on average, is able to find biclusters with a lower residue (Yang et al. did not report results for human data set.). However, the standard deviation of CC is much higher than that of SEBI, meaning that SEBI has a more stable behavior. FLOC improves CC on the yeast data set, but it is more general at finding biclusters than SEBI, which is much more specific and the biclusters therefore involve less number of genes. In short, the biclusters found by CC and FLOC are characterized by a lower row variance than those found by SEBI, although they have higher volumes, so many of them are not very interesting.

6 Conclusions

In this paper, we have introduced an algorithm based on EC, called SEBI, for finding biclusters on expression data. The proposed algorithm adopts a sequential covering strategy, and an EA in order to find biclusters. To avoid overlapping among biclusters, a weight is assigned to each element of the expression matrix. Weights are adjusted each time a bicluster is found. This is different from other



(a) (b) (c)

Fig. 10. Graphs relative to a typical run of EBI on the human data set. In (a), the average and best fitness at each generation are shown. In (b), the average and best volumes are shown for each generation. In (c), the average residue and the lowest residue at each generation are plotted.

TABLE 5
Performance Comparison between SEBI and CC

Algorithm	Dataset	Avg. residue	Avg. Volume	Avg. gene num.	Avg. cond. num
SEBI	Yeast	205.18 (4.49)	209.92 (171.39)	13.61 (10.38)	15.25 (1.37)
	Human	1028.84 (29.19)	615.84 (278.35)	14.07 (5.39)	43.57 (6.20)
CC	Yeast	204.29 (42.78)	1576.98 (2178.46)	166.71 (226.37)	12.09 (4.39)
	Human	850.04 (153.91)	4595.98 (3353.72)	269.22 (204.71)	24.5 (20.92)
FLOC	Yeast	187.54 (NA)	1825.78 (NA)	195.00 (NA)	12.80 (NA)
	Human	–	–	–	–

We compare the average residue, the average volume, and the average number of genes and conditions of the found biclusters. Standard deviation is given between brackets.

methods, which, for instance, substitute covered elements with random values. Experimental results confirm the quality of the proposed method for avoiding overlapping among biclusters.

It is interesting to notice that the default parameter setting of SEBI was adopted on both data sets used in the empirical experimentations. Other runs with different parameter settings were performed, but the results were not significantly different from those presented in this paper.

We can conclude that SEBI is successful in finding sets of genes that show strikingly similar up-regulations and down-regulations under a set of experimental conditions. This is confirmed, for instance, by the results on the human data set, where SEBI could find biclusters consisting of genes with a very similar behavior under a set of conditions. The quality of biclusters found by our evolutionary approach is discussed and the results are compared to those reported by Cheng and Church, and Yang et al. In general, SEBI shows an excellent performance at finding shifting and scaling patterns in gene expression data.

In short, Evolutionary Computation represents a useful framework for addressing the challenges of gene expression data analysis.

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for their valuable suggestions. The research was supported by the Spanish Research Agency CICYT under grant TIN2004-00159 and Junta de Andalucia (III Research Program).

REFERENCES

[1] A. Ben-Dor, R. Shamir, and Z. Yakhini, "Clustering Gene Expression Patterns," *J. Computational Biology,* vol. 6, nos. 3-4, pp. 281-297, 1999.

[2] H. Wang, W. Wang, J. Yang, and P.S. Yu, "Clustering by Pattern Similarity in Large Data Sets," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 394-405, 2002.

[3] J.A. Hartigan, "Direct Clustering of a Data Matrix," *J. Am. Statistical Assoc.,* vol. 67, no. 337, pp. 123-129, 1972.

[4] Y. Cheng and G.M. Church, "Biclustering of Expression Data," *Proc. Eighth Int'l Conf. Intelligent Systems for Molecular Biology,* pp. 93-103, 2000.

[5] G. Getz, E. Levine, and E. Domany, "Coupled Two-Way Clustering Analysis of Gene Microarray Data," *Proc. Natural Academy of Sciences USA,* pp. 12,079-12,084, 2000.

[6] L. Lazzeroni and A. Owen, "Plaid Models for Gene Expression Data," technical report, Stanford Univ., 2000.

[7] J. Yang, W. Wang, H. Wang, and P.S. Yu, "δ-Clusters: Capturing Subspace Correlation in a Large Data Set," *Proc. 18th IEEE Conf. Data Eng.,* pp. 517-528, 2002.

[8] J. Yang, W. Wang, H. Wang, and P.S. Yu, "Enhanced Biclustering on Expression Data," *Proc. Third IEEE Conf. Bioinformatics and Bioeng.,* pp. 321-327, 2003.

[9] A. Tanay, R. Sharan, and R. Shamir, " Discovering Statistically Significant Biclusters in Gene Expression Data," *Bioinformatics,* vol. 19, (Sup. 2), pp. 196-205, 2002.

[10] H. Wang, W. Wang, J. Yang, and P.S. Yu, "Clustering by Pattern Similarity in Large Data Sets," *Proc. ACM SIGMOD Conf.,* 2002, citeseer.ist.psu.edu/553109.html.

[11] J. Liu and W. Wang, "Op-Cluster: Clustering by Tendency in High Dimensional Space," *Proc. Third IEEE Int'l Conf. Data Mining,* p. 187-194, 2003.

[12] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini, "Discovering Local Structure in Gene Expression Data: The Order-Preserving Sub-Matrix Problem," *Proc. Sixth Ann. Int'l Conf. Computational Biology,* pp. 49-57, 2002.

[13] J. Hipp, U. Güntzer, and G. Nakhaeizadeh, "Algorithms for Association Rule Mining—A General Survey and Comparison," *SIGKDD Explorations Newsletter,* vol. 2, no. 1, pp. 58-64, 2000.

[14] J. Pei, X. Zhang, M. Cho, H. Wang, and P.S. Yu, "Maple: A Fast Algorithm for Maximal Pattern-Based Clustering," *Proc. Third IEEE Int'l Conf. Data Mining,* p. 259-266, 2003.

[15] J. Orling, "Containment in Graph Theory: Covering Graphs with Cliques," *Nederl. Akad. Wetensch. Indag. Math.,* vol. 39, pp. 211-218, 1977.

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

[17] S. Bleuler, A. Prelić, and E. Zitzler, "An EA Framework for Biclustering of Gene Expression Data," *Congress on Evolutionary Computation (CEC-2004),* pp. 166-173, 2004.

[18] S. Bleuler and E. Zitzler, "Order Preserving Clustering over Multiple Time Course Experiments," *Proc. EvoWorkshops 2005,* pp. 33-43, 2005.

[19] T. Bäck, D.B. Fogel, and Z. Michalewicz, *Evolutionary Computation 1: Basic Algorithms and Operators.* Inst. of Physics Publishing, 2000.

[20] X. Yao, *Evolutionary Computation: A Gentle Introduction,* chapter 2. Kluwer Academic Publishers, pp. 27-53, 2002.

[21] D.E. Goldberg and L. Robert, "Alleles, Loci, and the Travelling Salesman Problem," *Proc. First Int'l Conf. Genetic Algorithms,* pp. 154-159, 1985.

[22] P.J. Bentley and D.W. Corne, *Creative Evolutionary Systems.* Morgan Kaufmann Publishers Inc., 2001.

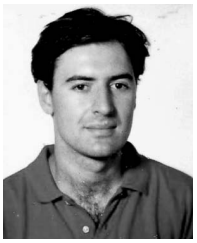
[23] T. Yamada and R. Nakano, "A Genetic Algorithm Applicable to Large-Scale Job-Shop Problems," *Parallel Problem Solving from Nature,* R. Männer and B. Manderick, eds. vol. 2, Amsterdam: Elsevier Science Publishers, B.V., 1992.

[24] D. Corne, P. Ross, and H.-L. Fang, "Fast Practical Evolutionary Timetabling," *Proc. Evolutionary Computing AISB Workshop,* pp. 251-263, 1994, citeseer.nj.nec.com/corne94fast.html.

[25] 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, pp. 317-324, 1995.

[26] G.F. Spencer, "Automatic Generation of Programs for Crawling and Walking," *Proc. Fifth Int'l Conf. Genetic Algorithms (ICGA '93)*, p. 654, 1993.
[27] D.B. Fogel, "Evolving Behavious in the Iterated Prisoner's Dilemma," *Evolutionary Computation*, vol. 1, no. 1, pp. 77-97, 1993.
[28] F. Divina and E. Marchiori, "Evolutionary Concept Learning," *Proc. Genetic and Evolutionary Computation Conf.*, pp. 343-350, July 2002.
[29] 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, pp. 875-882, 2001.
[30] 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, pp. 324-331, 2003.
[31] F. Divina and E. Marchiori, "Knowledge-Based Evolutionary Search forInductive Concept Learning," *Knowledge Incorporation in Evolutionary Computation*, Y. Jin, ed. ch. Part 3, Springer-Verlag, pp. 237-254, 2004.
[32] R. Cho, M. Campbell, E. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T. Wolfsberg, A. Gabrielian, D. Landsman, D. Lockhart, and R. Davis, "A Genome-Wide Transcriptional Analysis of the Mitotic Cell Cycle," *Molecular Cell*, vol. 2, pp. 65-73, 1998.
[33] A.A. Alizadeh, M.B. Eisen, R.E. Davis, C. Ma, I.S. Lossos, A. Rosenwald, J.C. Boldrick, H. Sabet, T. Tran, X. Yu, J.I. Powell, L. Yang, G.E. Marti, T. Moore, J. Hudson, L. Lu, D.B. Lewis, R. Tibshirani, G. Sherlock, W.C. Chan, T.C. Greiner, D.D. Weisenburger, J.O. Armitage, R. Warnke, R. Levy, W. Wilson, M.R. Grever, J.C. Byrd, D. Botstein, P.O. Brown, and L.M. Staudt, "Distinct Types of Diffuse Large B-Cell Lymphoma Identified by Gene Expression Profiling," *Nature*, vol. 403, pp. 503-511, 2000.
[34] S. Tavazoie, J. Hughes, M. Campbell, R. Cho, and G. Hurch, "Systematic Determination of Genetic Network Architecture," *Bioinformatics*, vol. 19, (Sup. 2), pp. 281-285, 1999.
[35] J. Yang, H. Wang, W. Wang, and P. Yu, "Enhanced Biclustering on Expression Data," *Proc. Third IEEE Symp. BioInformatics and BioEng. (BIBE '03)*, pp. 321-327, 2003, citeseer.ist.psu.edu/568558.html.

Federico Divina received a degree in computer science from the Ca' Foscari University of Venice, Italy, and the PhD degree with a dissertation on the use of hybrid Evolutionary Computation for Inductive Logic Programming from the Department of Computer Science at the Free University of Amsterdam, The Netherlands. He was a visiting researcher at the Machine Learning Group of the University of Seville, Spain. Since September 2004, he has been a postdoc at the Tilburg University, the Netherlands, in the European project NEW TIES. His research interests include evolutionary computation, machine learning, bioinformatics and language evolution. His homepage is http://www.cs.vu.nl/~divina.



Jesús S. Aguilar-Ruiz received the BSc degree in 1992, the MSc degree in 1997, and the PhD degree in 2001, all in computer science, from the University of Seville, Spain. He is an associate professor of computer science at Pablo de Olavide University, Seville, Spain. He has been a member of the programm committee of several international conferences, and a reviewer for relevant journals. His areas of research interest include evolutionary computation, data mining, and bioinformatics. His homepage is http://www.lsi.us.es/~aguilar.