JOURNAL OF INTEGRATED OMICS
A METHODOLOGICAL JOURNAL
HTTP://WWW.JIOMICS.COM

ORIGINAL ARTICLE | DOI: 10.5584/jiomics.v1i1.28

A simulated annealing-based algorithm for iterative class discovery using fuzzy logic for informative gene selection

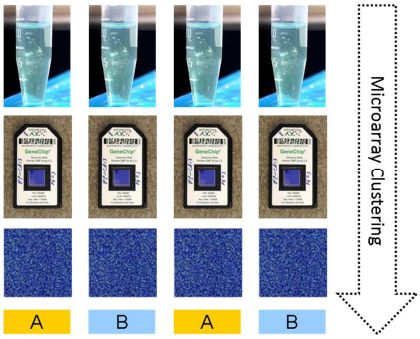
Daniel Glez-Peña[1], Miguel Reboiro-Jato[1], Florentino Fdez-Riverola*[1], Fernando Díaz[2].

[1]Dept. Informática, University of Vigo, Escuela Superior de Ingeniería Informática, Edificio Politécnico, Campus Universitario As Lagoas s/n, 32004, Ourense, Spain. Email Address: dgpena@uvigo.es, mrjato@sing.ei.uvigo.es, riverola@uvigo.es; [2]Dept. Informática, University of Valladolid, Escuela Universitaria de Informática, Plaza Santa Eulalia, 9-11, 40005, Segovia, Spain. Email Address: fdiaz@infor.uva.es

Received: 21 July 2010 **Accepted:** 28 August 2010 **Available Online**: 9 September 2010

ABSTRACT

Within a gene expression matrix, there are usually several particular macroscopic phenotypes of samples related to some diseases or drug effects, such as diseased samples, normal samples or drug treated samples. The goal of sample-based clustering is to find the phenotype structures or sub-structure of these samples. We present a novel method for automatically discovering clusters of samples which are coherent from a genetic point of view. Each possible cluster is characterized by a fuzzy pattern which maintains a fuzzy discretization of relevant gene expression values. Possible clusters are randomly constructed and iteratively refined by following a probabilistic search and an optimization schema. Evaluation of the proposed algorithm on publicly available microarray datasets shows high accuracy in spite of noise and the presence of other clusters. The results obtained support the appropriateness of using fuzzy logic to represent and filter gene expression values following an iterative approach. The proposed method complements our previous GENECBR system and both are freely available under GNU General Public License from http://www.genecbr.org/fpclustering.htm and http://www.genecbr.org/, respectively.



Keywords: Microarray data; Fuzzy discretization; Gene selection; Simulated annealing.

1. Introduction

Following the advent of high-throughput microarray technology it is now possible to simultaneously monitor the expression levels of thousands of genes during important biological processes and across collections of related samples. In this context, sample-based clustering is one of the most common methods for discovering disease subtypes as well as unknown taxonomies. By revealing hidden structures in microarray data, cluster analysis can potentially lead to more tailored therapies for patients as well as better diagnostic procedures.

From a practical point of view, existing sample-based clustering methods can be (*i*) directly applied to cluster samples using all the genes as features (i.e., classical techniques such as K-means, SOM, HC, etc.) or (*ii*) executed after a set of informative genes are identified. The problem with the first approach is the signal-to-noise ratio (smaller than 1:10), which is known to seriously reduce the accuracy of clustering results due to the existence of noise and outliers of the samples [1]. To overcome such difficulties, particular methods can be applied to identify informative genes and reduce gene dimensionality prior to clustering samples in order to detect their phenotypes. In this context, both supervised and unsupervised informative gene selection techniques have been developed.

While supervised informative gene selection techniques often yield high clustering accuracy rates, unsupervised

***Corresponding author:** Florentino Fdez-Riverola. Dept. Informática, University of Vigo, Escuela Superior de Ingeniería Informática, Edificio Politécnico, Campus Universitario As Lagoas s/n, 32004, Ourense, Spain. Email Address: riverola@uvigo.es.

informative gene selection methods are more complex because they assume no a priori phenotype information being assigned to any sample [2]. In such a situation, two general strategies have been adopted to address the lack of prior knowledge: (*i*) unsupervised gene selection, this aims to reduce the number of genes before clustering samples by using appropriate statistical models [3-5] and (*ii*) interrelated clustering, that takes advantage of utilizing the relationship between the genes and samples to perform gene selection and sample clustering simultaneously in an iterative paradigm.

Following the second strategy for unsupervised informative gene selection (interrelated clustering), Ben-Dor *et al.* [6] present an approach based on statistically scoring candidate partitions according to the overabundance of genes that separate the different classes. Xing and Karp [1] use a feature filtering procedure for ranking features according to their intrinsic discriminability and irredundancy to other relevant features. Their clustering algorithm is based on the concept of a normalized cut for grouping samples in new reference partition. von Heydebreck *et al.* [7] and Tang *et al.* [8] propose algorithms for selecting sample partitions and corresponding gene sets by defining an indicator of partition quality and a search procedure to maximize this parameter. Varma and Simon [9] describe an algorithm for automatically detecting clusters of samples that are discernable only in a subset of genes. They use iteration between Minimal Spanning Tree based clustering and feature selection to remove noise genes in a step-wise manner while simultaneously sharpening the clustering.

In this article we improve a previous initial work [10] by providing *i*) a complete mathematical formulation of the proposed method, *ii*) an evaluation of our method using two real datasets, herein referred as HC-Salamanca dataset [11] and Armstrong dataset [12] (see Sections 3.1 and 3.2 for a detailed description of these datasets), and *iii*) a comparison of the results obtained by the proposed method against the ones obtained by the standard hierarchical clustering algorithm for the same datasets. As introduced in [10], our clustering technique is based on the notion of *genetic coherence* of the each cluster, and this "coherence" is computed by taking into consideration the genes which share the same expression value through all the samples belonging to the cluster (which we term a *fuzzy pattern* or FP in short), but discarding those genes present due to pure chance (herein referred to *noisy genes* of a fuzzy pattern). The proposed clustering technique combines both (*i*) the simplicity and good performance of a heuristic search method able to find good partitions in the space of all possible partitions of the set of samples with (*ii*) the robustness of fuzzy logic, able to cope with several levels of uncertainty and imprecision by using partial truth values.

2. Material and methods

2.1 Overview of the proposed method.

As mentioned above we propose a simulated annealing-based algorithm for iterative class discovery. It uses a novel fuzzy logic method for informative gene selection. The interrelated clustering process carried out is based on an iterative approach in which possible clusters are randomly constructed and evaluated by following a probabilistic search and an optimization schema. Our clustering technique is not based on the distance between the microarrays belonging to each given cluster, but on the notion of genetic coherence of the own clusters. The genetic coherence of a given partition is calculated by taking into consideration the genes which share the same expression value through all the samples belonging to the cluster (we term this a fuzzy pattern), but discarding those genes present purely by chance (or noisy genes of a fuzzy pattern). The global view of the proposed method is sketched in Figure 2 and following sections give details about the mathematical background and proposed algorithm.

2.2 Fuzzy discretization and fuzzy pattern construction.

A fuzzy pattern is based on the fuzzy discretization given by three membership functions which are associated with each probe set in the microarray. Basically, for each probe set we consider three linguistic labels (Low, Medium and High levels of gene expression), each one associated with a polynomic function. Given a fixed value for the θ parameter (θ defines the threshold for the membership function from which a linguistic label is activated), the different labels are only activated in specific intervals within the whole range of

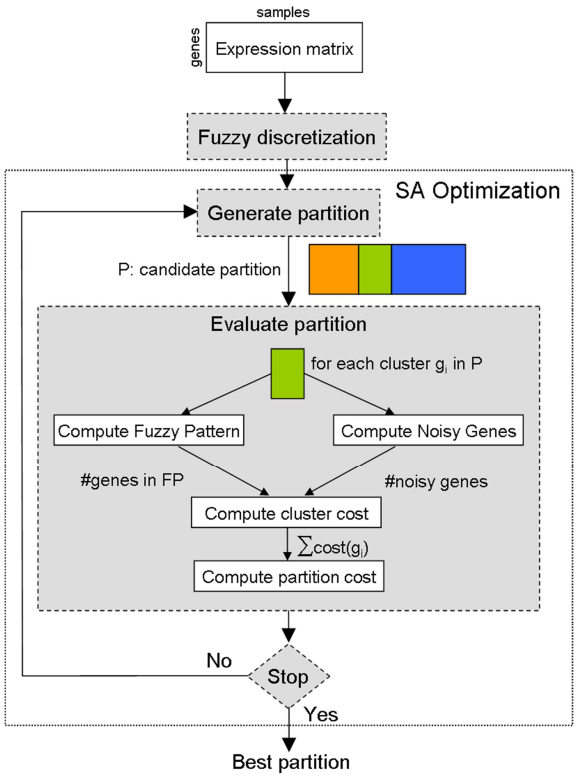


Figure 2. Overview of the iterative class discovery method. This figure shows how, from the fuzzy discretization of the microarrays from raw dataset, the method performs a stochastic search, looking for a "good partition" of microarrays in order to maximize the genetic coherence of each one cluster within the tentative partition. The simulated annealing technique is used to implement this search.

variation of a gene's expression level as shown in Figure 3.

For a specific probe set and a given value of the θ parameter, the label assigned to the gene's expression level in a probe will be one of the following alternatives:

- one of the three basic labels (LOW, MEDIUM and HIGH), if the numeric value is in only one of the associated intervals,
- one of the two combined labels (LOW-MEDIUM or MEDIUM-HIGH), if the numeric value is at the intersection of two intersections,
- the empty label, if the numeric value is in an interval where none of the labels are activated for the selected θ parameter.

Therefore, for each gene probe set we are considering a universe of six possible symbols: 'Low' (L), 'Low-Medium' (L-M), 'Medium' (M), 'Medium-High' (M-H), 'High' (H) and '*' (empty), herein denoted by $\boldsymbol{S} = \{s1, s2, s3, s4, s5, s6\}$ where sj maps the j-th symbol in the list given above.

Once the discretization of the whole microarray dataset $\boldsymbol{D}$ has been completed, given a subset of m microarrays, $D_m = \{x1, x2, \ldots, x_m\} \subseteq \mathbf{D}$, which represents any target concept (a class within a classification or a group in a clustering), its associated fuzzy pattern is constructed by selecting those linguistic labels which are different to the empty label and have a relative appearance frequency in set $\mathbf{D}_m$ equal to or greater than the predefined ratio given by the π parameter (with $0 < \pi \leq 1$). Formally, for a specific gene g_j ($1 \leq j \leq N$, where N is the number of probes in the microarray), the appearance frequency of any symbol $s \in \boldsymbol{S}$ in the set $\boldsymbol{D}_m$, $freq_j(s)$, can be computed according to the following expression:

$$freq_j(s) = \frac{\sum_{1 \leq i \leq m} \delta_j(\boldsymbol{x}_i, s)}{m}, \text{ where } \delta_j(\boldsymbol{x}_i, s) = \begin{cases} 1 & \text{if DiscretizedLabel}(\boldsymbol{x}_{ij}) = s \\ 0 & \text{otherwise} \end{cases}$$

Then, the gene g_j (with the label associated with the most frequent symbol sm_f) is selected as a candidate for the fuzzy pattern only if the most frequent symbol is different from the empty label and its appearance frequency is at least equal to the π parameter, namely, $freq_j(sm_f) \geq \pi$. Therefore, the parameter π controls the degree of exigency for selecting a gene in the fuzzy pattern, since the higher the value of the π parameter the fewer the number of genes which form the fuzzy pattern associated to the target concept $\boldsymbol{D}_m$.

Basically, our assertion is that a fuzzy pattern is able to capture the meaningful genes of any group of microarrays which are coherent from a genetic point of view. The underlying hypothesis is that any subtype of a given disease must necessarily have an internal genetic coherence, namely, those microarrays belonging to patients which suffer from one specific subtype, should share a large number of genes (i.e., present a similar expression level in a large number of genes, at least, more than if the microarrays come from patients with different subtypes of the same or a different disease). This fact has been empirically observed in previous experiments carried out with our GENECBR platform, a translational tool for multiple-microarray analysis and integrative information retrieval for aiding diagnosis in cancer research, when we studied the differences between the number of genes in the FP belonging to well and not well-defined pathologies [13-15]. This circumstance supports the development of new approaches able to take advantage of this situation. The underlying idea is that if this behaviour is observed in well-known and well-characterized classes of a disease, it must also be true in unknown groups representing new subtypes of the disease. Therefore, these newly discovered classes must be characterized by having a large number of genes in their associated fuzzy patterns. As a consequence, this situation can be used to consider the problem of clustering microarrays in terms of maximizing the number of genes in the fuzzy pattern associated with each cluster.

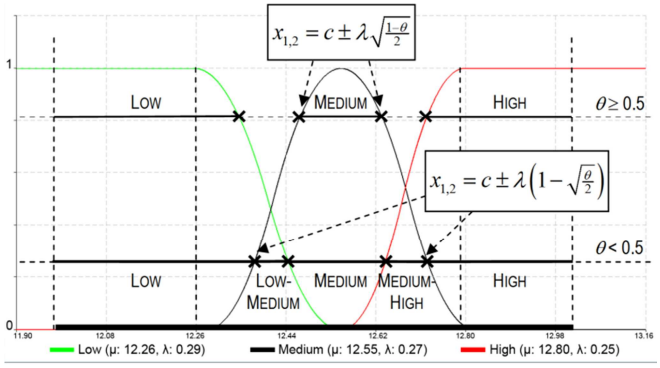


Figure 3. Membership functions and activation regions. This figure illustrates the three membership functions for a specific probe and shows the cut points which determine the length of the segment of each fuzzy expression level within the probe domain.

A key issue is the setting-up process of the parameters θ and π, since the computation of fuzzy patterns is high sensitive to these values. Although for different learning tasks (prediction or supervised classification), in previous works [13-14], a cross-valitation strategy was used to set up these parameters for the same two datasets tested in this work and then, those values has been also used in the experimental section of this work.

2.3 Noisy genes identification.

Now, working with a set of m microarrays, we are interested in the estimation of the probability that a specific gene, g_i, of the n available in a microarray, may appear in a fuzzy pattern merely by chance. First of all, we need an estimation of the occurrence probability of each symbol in the fuzzy discretization of numeric data, namely, $p(\boldsymbol{S}) = (p(s_1), p(s_2), p(s_3), p(s_4), p(s_5), p(s_6))$. Given a fixed θ, these probabilities can be estimated by the ratio of the length of each interval (associated to the labels) and the length of the whole variation range, $\Delta = c_H - c_L + \lambda_H + \lambda_L$ (see Figure 3).

Since the membership functions are polynomic, the length of each interval can be computed in a closed form. For example, the cut points of the membership function for the 'Medium' label with the line $\mu_M(x) = \theta$, is given by:

$$x_{1,2} = c_M \pm \delta_M(\theta)$$

Where:

$$\delta_M(\theta) = \begin{cases} \lambda_M \sqrt{\frac{1-\theta}{2}} & \text{if } 0.5 \le \theta \le 1 \\ \lambda_M \left(1 - \sqrt{\frac{\theta}{2}}\right) & \text{if } 0 \le \theta < 0.5 \end{cases}$$

In this way, the probabilities of each symbol in $\boldsymbol{S}$ can be computed, and obviously, their sum is equal to 1.

In a second step, having an estimate of the probability of each possible symbol, we need to assess the probability that the gene g_i appears in the fuzzy pattern associated with a sample of m microarrays with a minimum frequency ratio equal to π. Under these conditions, in order to include the gene in the fuzzy pattern it is necessary that in the set of m microarrays, it must have at least $\lceil \pi \cdot m \rceil$ repetitions of the same symbol in their associated fuzzy representations. The empty label (with symbol '*') must be excluded since it reflects that none of the labels are activated and therefore, it never can be part of a fuzzy pattern. By $p(k)$ we denote the probability that a valid label (all except the '*' symbol), appears exactly in k discretized values (of the m available). It can be shown that this probability is given by:

$$\begin{aligned} p(k) &= \sum_{j=1}^{5} p(k \mid s_j) p(s_j) \\ &= \sum_{j=1}^{5} p(s_j) \cdot C_m^k \cdot \left(p(s_j)^k (1 - p(s_j))^{m-k} \right) \end{aligned}$$

Where $C_m{}^k$ is the number of ways of picking k unordered outcomes from m possibilities, and $p(s_j)^k(1 - p(s_j))^{m-k}$ stands for the probability that symbol s_j appears k times in a sequence of m symbols.

Therefore, the probability that a gene g_i appears in the fuzzy pattern which is associated with a sample of m microarrays, is given by the sum of individual probabilities that any symbol appears, at least, $\lceil \pi \cdot m \rceil$ times. This probability can be calculated by means of the following expression:

$$\begin{aligned} p(g_i) &= \sum_{k=\lceil \pi \cdot m \rceil}^{m} p(k) \\ &= \sum_{k=\lceil \pi \cdot m \rceil}^{m} \sum_{j=1}^{5} p(s_j) \cdot C_m^k \cdot \left(p(s_j)^k (1 - p(s_j))^{m-k} \right) \end{aligned}$$

Where $p(g_i)$ is upper bound by 1, being closer to 1 depending on the distribution probability of each gene and without taking account of the degree of exigency imposed by π. For example, assuming that the probability of the empty label is null, the probability is closer to one as there is a predominant symbol (a symbol with occurrence probability close to 1), whereas the worst case is represented by the situation where all the valid symbols have the same probability of occurrence.

Finally, assuming that, in the random case, the selection of a gene g_i is independent of the selection of another one g_j ($j \neq i$), the number of noisy genes due to the chance for a group of m microarrays with N probes at the levels imposed by θ and π parameters, is given by:

$$\begin{aligned} ng(m, \theta, \pi) &= \sum_{i=1}^{N} p(g_i) = \sum_{i=1}^{N} \sum_{k=\lceil \pi \cdot m \rceil}^{m} p(k) \\ &= \sum_{i=1}^{N} \sum_{k=\lceil \pi \cdot m \rceil}^{m} \sum_{j=1}^{5} p(s_j) \cdot C_m^k \cdot \left(p(s_j)^k (1 - p(s_j))^{m-k} \right) \end{aligned}$$

Where $ng(m, \theta, \pi)$ is upper bound by N, although this value is only reachable in the ideal case where only one symbol has probability 1 for all of the genes.

As the uncertainty decreases (there is a predominance of one expression level over the other ones for all the genes in the available set of microarrays) the number of $ng(m, \theta, \pi)$ decreases (the amount of information encoded by the data also decreases and then, there are more irrelevant genes). When uncertainty increases, the amount of information also grows and more genes are necessary to distinguish samples in absence of other information. Figure 4 illustrates the variation of noisy genes depending on the θ, and π parameters.

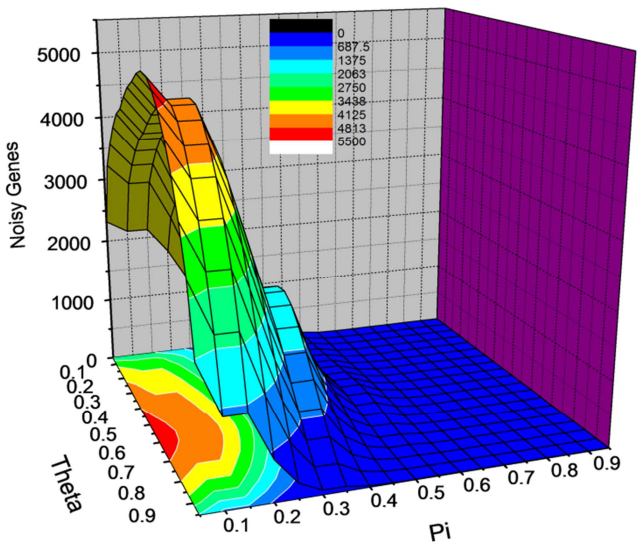


Figure 4. Noisy genes vs. Theta (θ) and Pi (π) parameters (m = 43 samples, n = 22,283 genes). This figure shows the variation in the number of noisy genes depending on the θ and π parameters of a fuzzy pattern for the HC-Salamanca dataset. For a fixed value of parameter θ, the number of noisy genes decreases exponentially when the π parameter grows (since the prior probability that a gene appears in the fuzzy pattern also decreases). For a fixed value of parameter π, the maximum number of noisy genes is symmetrically distributed around θ = 0.5 (since the prior probability of each linguistic label in a fuzzy pattern is nearly equiprobable).

2.4 The cost function of a cluster

The cost function must combine two factors: (*i*) the number of genes in the fuzzy pattern associated with each cluster of the partition and (*ii*) the size of each such cluster. The first factor in the cost function models the genetic coherence of a cluster. Accepting this hypothesis, it is expected that for clusters with equal sizes, the number of genes in a fuzzy pattern will be greater if the genetic coherence of the cluster is higher. The second factor is relevant since it has been experimentally observed that meaningful genes in large clusters (after noisy genes have been filtered out) are several orders of magnitude smaller than meaningful genes computed in small clusters. This fact is reasonable because it will be more probable when the number of possibilities is also further reduced. Therefore, the size factor in the cost function is needed for examining comparable clusters of different size.

Under these assumptions, given the available set of microarrays denoted by $\boldsymbol{X}$, and a partition $\boldsymbol{P} = \{g_1, g_2, \ldots, g_k\}$ of $\boldsymbol{X}$ in k clusters, that is to say, $g_i \subseteq \boldsymbol{X}$ with $1 \le i \le k$, $g_i \cap g_j = \varnothing$ (if $i \ne j$) and $\cup g_i = \boldsymbol{X}$, the cost of a cluster $g_i \in \boldsymbol{P}$ is given by:

$$\cos t(g_i) = \frac{N}{2^{|g_i|}\left(\left|fp(g_i, P, \theta, \pi)\right| - ng(|g_i|, \theta, \pi)\right)}$$

Where $|g_i|$ is the number of microarrays in cluster g_i, $|fp(g_i, \boldsymbol{P}, \theta, \pi)|$ is the size (in genes) of the fuzzy pattern associated with the group g_i with regard to the classification given by $\boldsymbol{P}$ and for specific values of the θ and π parameters, and $ng(|g_i|, \theta, \pi)$ is the number of noisy genes of a group with $|g_i|$ microarrays. Finally, N is the fixed number of probes in a microarray. Therefore, if one tries to minimize the defined cost of a group, $cost(g_i)$, it involves trying to maximize both (*i*) the size of the cluster and (*ii*) its genetic coherence (measured by the number of genes belonging to its fuzzy pattern).

Finally, the cost of a given partition $\boldsymbol{P} = \{g_1, g_2, \ldots, g_k\}$ of $\boldsymbol{X}$ is defined by the sum of the individual costs of each group:

$$\cos t(P) = \sum_{i=1}^{k} \cos t(g_i)$$

Once the cost of a tentative partition of microarrays has been established, one needs to define an algorithmic strategy in order to automatically build random partitions from the set of available microarrays.

2.5 Algorithm

There are certain optimization problems that become unmanageable using combinatorial methods as the number of objects becomes large. The simulated annealing technique which can be regarded as a variant of a local search was first introduced by Metropolis *et al.* [16], and then used in optimization problems by Kirkpatrick *et al.* [17] and Černy [18]. For these problems, the simulated annealing method represents a very effective practical algorithm. Although this technique is unlikely to find the optimum solution, it can often discover a very good one even in the presence of noisy data.

Simulated annealing improves its behaviour through the introduction of two tricks. The first one is the so-called *Metropolis algorithm* [16], in which some poor solutions are accepted when they serve to allow the solver to explore more of the possible solution space. Such bad solutions are tolerated using the criterion that:

$$e^{-\Delta D/T} > U(0, 1)$$

Where ΔD is the variation of the cost function for the current solution and the best one, T stands for a *synthetic temperature* and $U(0, 1)$ is a random number in the interval [0, 1]. The cost function D corresponds to the free energy in the case of annealing a metal (in which case the temperature parameter would actually be the kT, where k is Boltzmann's Constant and T is the physical temperature in the Kelvin absolute temperature scale). If T is large, many bad solutions are accepted and a considerable part of the solution space is accessed. The next solutions to explore are randomly constructed, though more sophisticated techniques can be used.

The second trick is, again by analogy with the annealing of a metal, to lower the *temperature*. After making many changes in the current solution and observing that the cost function declines only slowly, one lowers the temperature limiting while the size of allowed bad solutions. After lowering the temperature several times to reach a smaller value, one may then "quench" the process by accepting only good solutions in order to find the local minimum of the cost function. There are various annealing schedules for lowering the temperature, but the results are generally not very sensitive to the details. These general ideas are the basis of simulated annealing but a comprehensive introduction to the subject can be found in [19].

The application of our simulated annealing approach to cluster microarrays is sketched in Figure 5. First of all, we consider a pool which contains the set of m microarrays that must be clusterized into k different and unknown groups. In the final solution, some microarrays can stay in the pool without being associated with any cluster. Initially, a first solution to the problem (a partition of microarrays) is randomly constructed. All the microarrays of the pool are randomly distributed among k classes, where k is the desired number of clusters in the partition (the whole set of m microarrays are spread proportionally among the k clusters and the pool).

At every step, a neighbour solution is determined by choosing one from the following alternatives (see Figure 6 for the details):

- Moving a randomly chosen microarray from the pool to a cluster (perhaps empty).

- Moving a randomly chosen microarray from a cluster to the pool.
- Exchanging randomly chosen microarrays among clusters.
- Exchanging randomly chosen microarrays among a cluster and the pool.
- Moving a randomly chosen microarray from one cluster to another cluster.

The neighbour solutions of lower cost obtained in this way are always accepted, whereas those solutions with a higher cost are accepted with the following probability:

$$Pr = T_i / (T_i + \delta)$$

Where δ is the cost difference among the new solution and the old solution, and T_i (i = 0, 1, …) represents the temperature of annealing which drops from a value T_0 (the cost of the initial solution) according to the formula $T_{i+1} = T_i \cdot \alpha$, where $\alpha < 1$. Pr implies that large increases in the solution cost (uphill moves) are more likely to be accepted when T_i is high. As T_i approaches zero most uphill moves are rejected.

The general algorithm stops if equilibrium is encountered. We define that equilibrium is reached if, after 50 stages of temperature reduction, the best achieved solution can not be improved. In contrast to the classical approach in which a solution to the problem is taken as the last solution obtained in the annealing process, we memorize the best solution found during the whole annealing process (Cf. lines 13-15 in Figure 5). Moreover, at the beginning of each temperature epoch, the search is restarted from the best solution reached at the moment (Cf. line 6 in Figure 5).

Summing up, the proposed annealing algorithm performs the local search by sampling the neighbourhood randomly. It attempts to avoid becoming prematurely trapped in a local optimum by sometimes accepting low-grade solutions. The acceptance level depends on the magnitude of the increment of the solution cost and on the spent search time. By this reason, and specially, at initial stages, when the temperature

```
   Input:
   → microarray dataset (pool of m microarrays) to be grouped in an unsupervised way
   → number of clusters (k)
   Output:
   ← partition of the original dataset into k clusters
   Require:
   next_solution routine
   cost function
   Steps:
1  current_solution = initial partition {Builds randomly a partition of k clusters with the microarrays in the pool}
2  best_solution = current_solution {initialize the best partition built at the moment}
3  equilibrium_counter = 0 {initialize the counter which controls the annealing epochs without improvement of the best found solution}
4  T = cost(current_solution) {initial temperature of the annealing process}
5  repeat
6      current_solution = best_solution {The annealing epoch starts from the best partition at the moment}
7      for iteration_counter = 1 to m do {An annealing epoch is made up of m attempts, where m is the number of microarrays in the pool}
8          new_solution = next_solution(current_solution) {Builds a new partition from the current one}
9          δ = cost(new_solution) – cost (current_solution) {Computes the difference in cost of the new partition and the current partition}
10         x = u(0, 1) {Generate random x uniformly in the range (0, 1)}
11         if ( δ < 0 ) or ( x < T / (T+δ) ) then {Accept a new solution if it improves the cost or increases the cost but it has a high probability of acceptance, the term T /(T+δ) which depends on the current temperature and the difference of the costs}
12             current_solution = new_solution {update the current solution}
13             if ( cost(new_solution) < cost (best_solution) ) then {if new partition improves best partition at the moment }
14                 best_solution = new_solution {update best partition}
15                 equilibrium_counter = 0 {reinitialize the equilibrium counter}
16     T = T · α {Decrease current temperature multiplying by a constant rate α = 0.95}
17     equilibrium_counter = equilibrium_counter + 1 {increment the number of epochs without improvement}
18 until equilibrium_counter > 50 {stop the annealing process when a stationary state is reached, at least 50 epochs without improvement}
```

Figure 5. General pseudo code of simulated annealing-based clustering algorithm. This algorithm explains the steps involved in partitioning a microarray dataset into *k* clusters by grouping microarrays which maximize its genetic coherence (assessed in terms of the number of genes in their associated fuzzy pattern), using a simulated annealing search algorithm.

Input:
→ current partition of the original microarray dataset (current_solution)
Output:
← neighbour partition which is built from the input current_solution by randomly choosing one of five possible movements of microarrays between clusters
Require:
-

Steps:

```
1   new_solution = current_solution {The new partition is built from the current partition}
2   Choose randomly two different clusters of new_solution, c_i and c_j
3   Select randomly three microarrays: m_i, m_j and m_k, belonging to c_i, c_j and the pool, respectively
4   u01 = u(0, 1) {Generate random u01 uniformly in the range (0, 1)}
5   if ( u01 < 0.2 ) then
6           move microarray m_i (from cluster c_i) to the pool
7   if ( 0.2 ≤ u01 < 0.4 ) then
8           move microarray m_k (from pool) to cluster c_i
9   if ( 0.4 ≤ u01 < 0.6 ) then
10          exchange microarray m_i (from cluster c_i) with microarray m_j (from cluster c_j)
11  if ( 0.6 ≤ u01 < 0.8 ) then
12          exchange microarray m_i (from cluster c_i) with microarray m_k (from pool)
13  if ( 0.8 ≤ u01 ≤ 1 ) then
14          move microarray m_i (from cluster c_i) to cluster c_j
15  return new_solution
```

Figure 6. Pseudo code of new_solution function. This algorithm explains the steps involved in building a neighbour partition from the current partition by randomly choosing one of the five possible operations.

is high, it has no sense to set the initial solution with a "reasonable" solution (for example, computed by a simple clustering algorithm) because the algorithm perhaps will be accept other solutions quite different from the original one, since the goal of the algorithm is to perform a global exploration of the search space. Only when the search process progresses, the exploitative component of the algorithm dominates over the explorative component, performing a local search around the selected local optimum after these intial stages. Obviously, the convergence time of the the proposed algorithm is higher than other deterministic clustering algorithms, but these algorithms have no capability to escape from local optima. Moreover the computational effor of the proposed clustering technique (up to several hours of running time per execution) since the evaluation of the cost function requires the computation of a fuzzy pattern for each cluster in the current partition.

3. Resutls and Discussion

3.1 The HC-Salamanca dataset

This dataset consists of bone marrow samples from 43 adult patients with de novo diagnosed acute myeloid leukemia (AML) – 10 acute promyelocytic leukemias (APL) with t(15;17), 4 AML with inv(16), 7 monocytic leukemias and 22 nonmonocytic leukemias, according to the WHO classification. All samples contained more than 80% blast cells and they were analyzed using high-density oligonucleotide microarrays (specifically, the Affymetrix GeneChip Human Genome U133A Array) [11]. In [11], hierarchical clustering analysis segregated APL, AML with inv(16), monocytic leukemias and the remaining AML into separate groups, so we consider this partition as the reference classification for validating our proposed technique in the following experimentation. The results of the proposed algorithm with this dataset are depicted in Figure 7.

Figure 7 shows for each available microarray the percentage of the times it has been grouped together with other microarrays belonging to the reference groups (APL, AML with inversion, Monocytic and Other AML) in the ten executions of the whole algorithm.

As can be seen in Figure 7, each sample has a different percentage of membership to each one of the reference groups. From this representation it can also be seen that the APL group is the most promising cluster since the algorithm has clustered together (in an unsupervised way) the majority of samples from this class in its ten executions. This result is consistent with the fact that this pathology is the best characterized class among the AML subtypes and, therefore, there is a high probability that microarrays within this subtype are well labelled in the reference classification. The Other-AML category seems to be another class, at least different from other clusters except the AML with-inversion group. This is the uncertain subtype of AML, since it contains those samples which are not classified within other groups. In the same way that the Other-AML group (but to a lesser degree), the monocytic leukemias seem to be another possible group. Finally, the AML with inversion is the most doubtful class since samples from this group are misclassified among the Other-AML and monocytic groups. This fact can be due

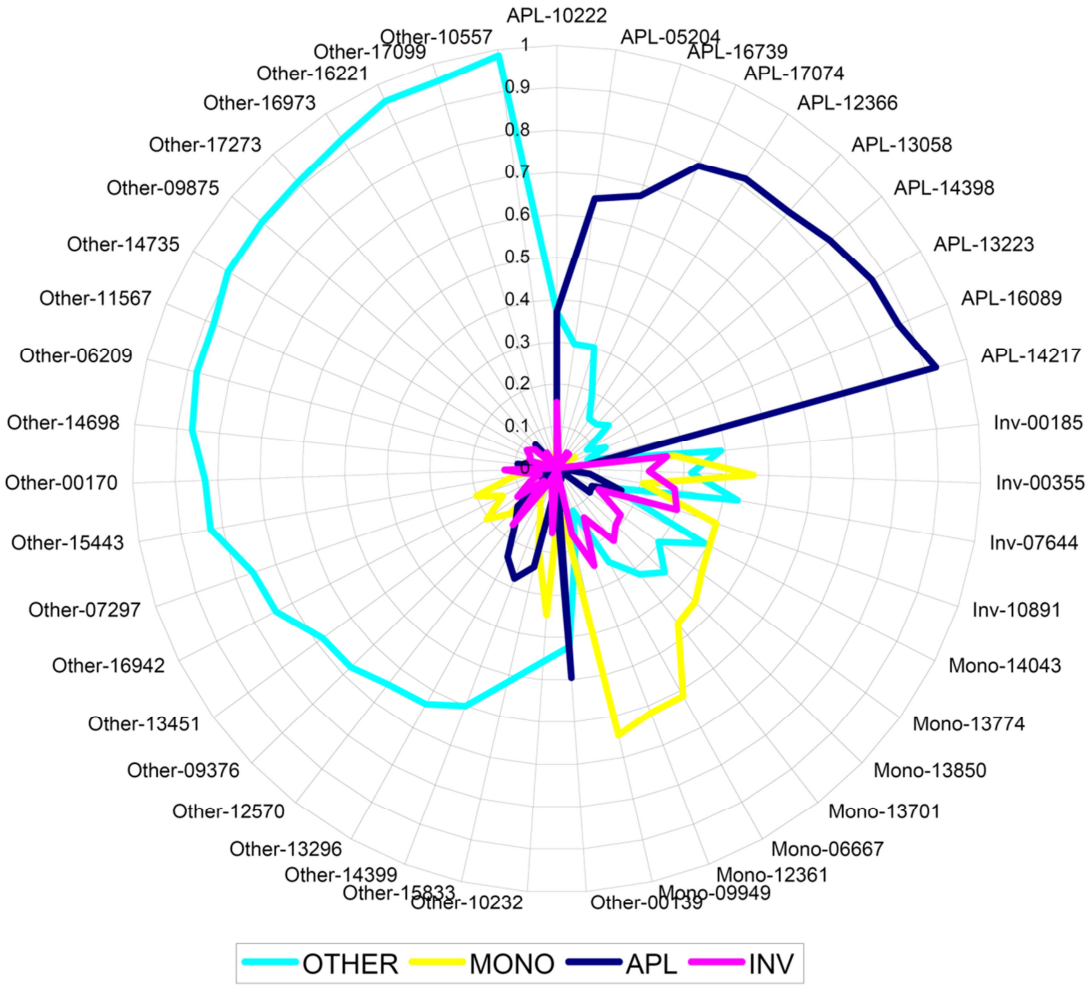


Figure 7. Degree of affinity of microarrays from the HC-Salamanca dataset with regard to the reference groups. The radial chart shows the same number of graphs as groups in the reference clustering. Each graph indicates the percentage of times in which a given microarray has been clustered with other microarrays in a reference cluster. As can be observed, the APL and Other-AML clusters are clearly differentiated, only the samples Other-10232 and APL-10222 have the highest percentage less than 50%, whereas Other-00139 is mainly grouped with samples in the APL group. The monocytic group is identified to a lesser degree but can still be differentiated. Samples from the AML with inversion group are confused with those belonging to the monocytic and Other-AML groups. Results are coherent with the hierarchical clustering reported in [11].

to the reduced number of available samples or to the lack of genetic coherence within this group, since the classification of these samples was performed by examining the karyotype by an expert.

The percentage of times (on average) in which microarrays of each reference cluster have been grouped together with microarrays belonging to different classes is shown in each row of Table 1.

Table 1 can be interpreted as a confusion matrix

Table 1. Confusion matrix for the HC-Salamanca dataset.

		Predicted class			
		APL	*Inv*	*Mono*	*Other*
True class	*APL*	76.19%	2.71%	2.18%	18.92%
	Inv	7.79%	26.49%	33.66%	32.06%
	Mono	3.11%	17.81%	51.73%	27.35%
	Other	8.62%	5.56%	8.70%	77.12%

numerically supporting the facts commented above, since the APL and Other-AML groups are the better identified pathologies (in an average percentage of 76.19% and 77.12% for all their samples and runs of the algorithm), followed by the monocytic leukemias (with an average percentage of 51.73%). As mentioned above, the AML with-inversion group is confused in a mean percentage of 33.66% and 32.06% with samples from monocytic and Other-AML groups, respectively.

If we consider that the highest percentage for each microarray determines the cluster to which it belongs, the final clustering obtained by our simulated annealing-based algorithm is shown in Table 2.

Assuming as "ground truth" the clustering given by experts, the performance of the clustering process can be tested by comparing the results given in both tables.

Some commonly used indices such as the *Rand index* and the *Jaccard coefficient* [20] have been defined to measure the

degree of similarity between two partitions. For the clustering given by our experiments, the *Rand index* was 0.90 and the *Jaccard coefficient* was 0.77.

Table 2. Final clustering for the HC-Salamanca dataset

APL	APL-05204, APL-10222, APL-12366, APL-13058, APL-13223, APL-14217, APL-14398, APL-16089, APL-16739, APL-17074, **Other-00139**
Mono	**Inv-00355, Inv-10891**, Mono-06667, Mono-09949, Mono-12361, Mono-13701, Mono-13774, Mono-13850, Mono-14043
Other	**Inv-00185, Inv-07644**, Other-00170, Other-06209, Other-07297, Other-09376, Other-09875, Other-10232, Other-10557, Other-11567, Other-12570, Other-13296, Other-13451, Other-14399, Other-14698, Other-14735, Other-15443, Other-15833, Other-16221, Other-16942, Other-16973, Other-17099, Other-17273

In order to compare with other standard methods, a new clustering has been computed using the hierarchical clustering algorithm (with $k = 8$, an average linkage strategy, and without any low-variance filter). In order to compute the *Rand index* and *Jaccard coefficient* with regard to this reference partition, a final partition has been computed assigning to each sample in a cluster the reference label which is the most frequent in the cluster. This final clustering obtained from the original results of the hierarchical clustering algorithm for the HC-Salamanca dataset is shown in Table 3.

Table 3. Final clustering computed from the results of the hierarchical clustering algorithm for the HC-Salamanca dataset.

APL	APL-05204, APL-10222, APL-12366, APL-13058, APL-13223, APL-14217, APL-14398, APL-16089, APL-16739, APL-17074, **Other-00139**
Mono	**Inv-00185, Inv-00355, Inv-07644, Inv-10891,** Mono-06667, Mono-09949, Mono-12361, Mono-13701, Mono-13774, Mono-13850, Mono-14043, **Other-10232, Other-10557, Other-13451, Other-15443, Other-15833**
Other	Other-00170, Other-06209, Other-07297, Other-09376, Other-09875, Other-11567, Other-12570, Other-13296, Other-14399, Other-14698, Other-14735, Other-16221, Other-16942, Other-16973, Other-17099, Other-17273

In this case, the computed *Rand index* is 0.79 (against 0.90 of our proposal) and the *Jaccard coefficient* is 0.53 (against 0.77). Besides the worse behavior of the hierarchical clustering algorithm from a quantitative viewpoint (derived from the values of these two indexes), it can be observed from Table 3 that hierarchical clustering algorithm and our algorithm agree that the Other-00139 sample is very similar to the APL samples, but hierarchical clustering algorithm does not distinguish clearly among samples from monocyte, with-inversion and other groups.

3.2 The Armstrong dataset

In [12] the authors proposed that lymphoblastic leukemias with MLL translocations (mixed-lineage leukemia) constitute a distinct disease, denoted as MLL, and show that the differences in gene expression are robust enough to classify leukemias correctly as MLL, acute lymphoblastic leukemia (ALL) or acute myeloid leukemia (AML). The public dataset of this work, herein referred to as the Armstrong dataset, has been also used to test our proposal. The complete group of samples consists of 24 patients with B-Precursor ALL (ALL), 20 patients with MLL rearranged B-precursor ALL (MLL) and 28 patients with acute myeloid leukemia (AML). All the samples were analyzed using the Affymetrix GeneChip U95a which contains 12600 known genes.

The results of the proposed clustering algorithm working with this dataset are shown in Figure 8. As in the previous examples, Figure 8 shows the percentage of times that each available microarray has been grouped together with other microarrays belonging to the reference groups (ALL, AML and MLL) in the ten executions of the algorithm.

The percentage of times (on average) in which microarrays of each reference cluster have been grouped together with microarrays of different classes is shown in Table 4. These percentages can be considered as an estimation of the overlapping area of the membership functions of any two potential groups in the sector associated to a true class.

As can be seen in Figure 8 (by analyzing the overlapping areas of membership graphs in the associated sectors to each one of the reference groups) the AML group is clearly distinguished from the ALL and the MLL groups (the confusion with regard to the ALL group is marginal being slightly larger with regard to the MLL group). The ALL group is clearly differentiated from the AML group. The main component of confusion in this group is from the MLL group (showing a clear overlap of the MLL and ALL membership graphs in the sector associated with the true ALL samples). Finally, the MLL can be distinguished to a lesser extent with respect to the other groups, the sources of confusion, in decreasing order, being the ALL and AML groups, respectively.

Therefore, the above assertions based on the interpretation of Figure 8, are numerically supported by the results shown in Table 4.

As in the HC-Salamanca dataset, if the highest percentage for each sample determines the cluster of the microarray, the final clustering obtained by our simulated annealing-based algorithm is shown in Table 5.

As in the previous experiment, assuming the clustering given by experts is the "ground truth", the performance of the clustering process can be examined by comparing the results given in both tables. In this case, the *Rand index* and the *Jaccard coefficient* for experiments carried out are 0.89 and

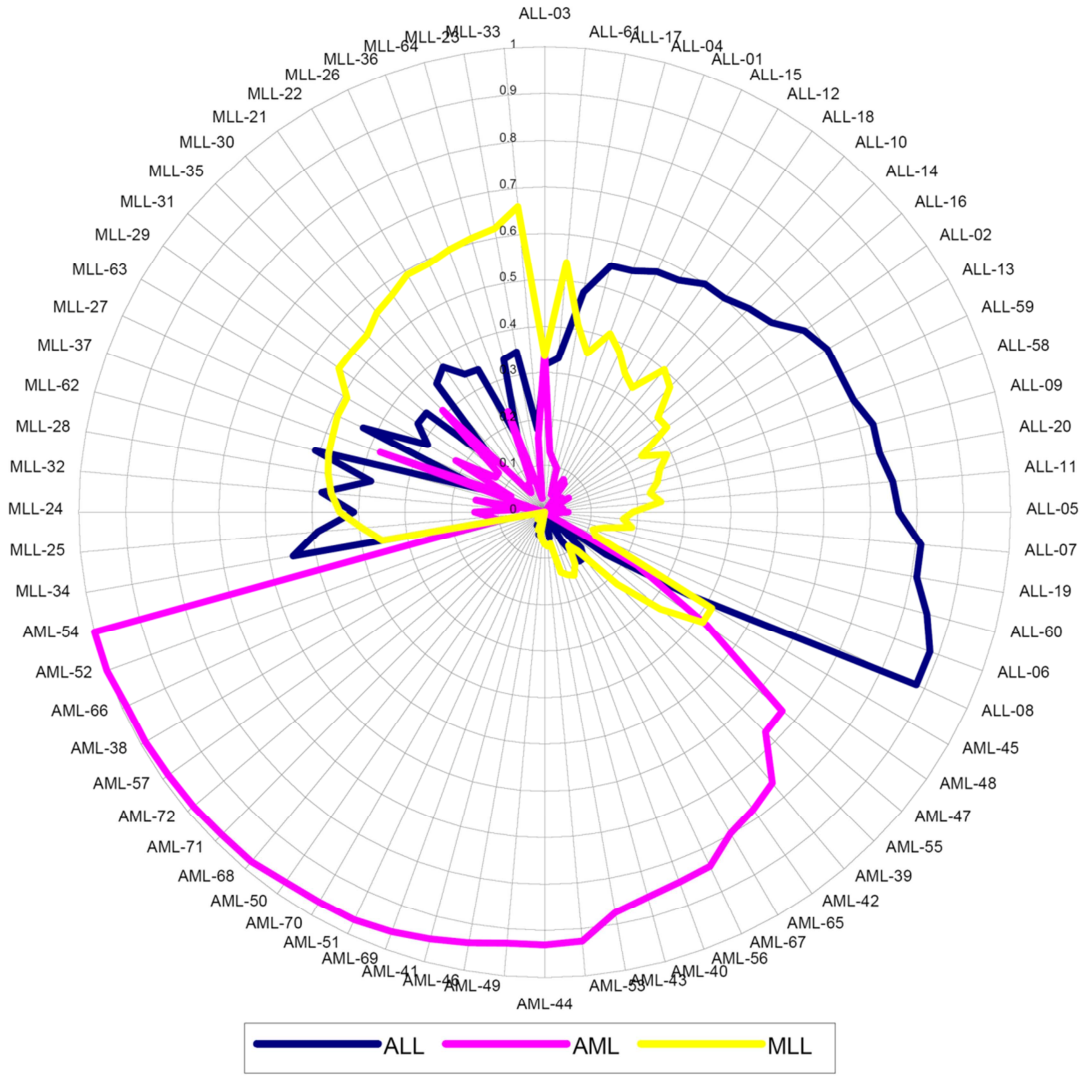


Figure 8. Degree of affinity microarrays from the Armstrong dataset with regard to the reference groups. The radial chart shows three membership graphs (one for each reference group) in the domain of available microarrays in the Armstrong dataset. From this figure the specific samples which are usually grouped with microarrays in other reference groups can be identified: the AML-45 and ALL-61 samples in the group of the MLL samples, the ALL-03 sample (the most doubtful) in the AML and MLL groups, and the MLL-25, MLL-32, MLL-34 and MLL-62 samples in the ALL group.

0.72, respectively. These indices are comparable to those obtained with the HC-Salamanca dataset. Moreover, if we assume a binary classification (considering the ALL and MLL groups as a unique partition of the lymphoblastic leukemias, the results are improved to 0.95 for the *Rand index* and 0.90 for the *Jaccard coefficient*, which are very close to the perfect match in a situation where the reference groups are also completely consistent.

Table 4. Confusion matrix for the Armstrong dataset.

		Predicted class		
		ALL	***AML***	***MLL***
True class	***ALL***	65.88%	5.16%	28.95%
	AML	4.42%	86.40%	9.18%
	MLL	34.74%	12.85%	52.41%

The hierarchical clustering algorithm (with k = 9, an average linkage strategy, and without any low-variance filter) was also executed to compute another clustering. As in previous section, from the original results of the hierarchical clustering algorithm, a final partition was built considering the most frequent reference label in a cluster to assign it to each sample in this cluster. The final clustering for the Armstrong dataset and computed from the original results of the hierarchical clustering algorithm are shown in Table 6.

In this case, the *Rand index* is 0.84 (against the 0.89 of our proposal) and the *Jaccard coefficient* is 0.64 (against 0.72), which reveal a worse behaviour of the hierarchical clustering algorithm against our clustering technique with regard to the reference partition given by the experts. From Table 6 it can be also observed that our clustering technique and the

Table 5. Final clustering for the Armstrong dataset.

ALL	ALL-01, ALL-02, ALL-04, ALL-05, ALL-06, ALL-07, ALL-08, ALL-09, ALL-10, ALL-11, ALL-12, ALL-13, ALL-14, ALL-15, ALL-16, ALL-17, ALL-18, ALL-19, ALL-20, ALL-58, ALL-59, ALL-60, **MLL-25, MLL-32, MLL-34, MLL-62**
AML	**ALL-03**, AML-38, AML-39, AML-40, AML-41, AML-42, AML-43, AML-44, AML-46, AML-47, AML-48, AML-49, AML-50, AML-51, AML-52, AML-53, AML-54, AML-55, AML-56, AML-57, AML-65, AML-66, AML-67, AML-68, AML-69, AML-70, AML-71, AML-72
MLL	**ALL-61, AML-45**, MLL-21, MLL-22, MLL-23, MLL-24, MLL-26, MLL-27, MLL-28, MLL-29, MLL-30, MLL-31, MLL-33, MLL-35, MLL-36, MLL-37, MLL-63, MLL-64

hierarchical clustering algorithm have a similar behaviour for the AML group (in the case of hierarchical algorithm is perfect) but it gets worse for the MLL group (decreasing considerably the number of samples in this cluster) and the ALL group (increasing the confusion with regard to the MLL group).

3.3 Discussion

The aim of the experiments reported in the previous section is to test the validity of the proposed clustering method. Dealing with unsupervised classification, it is very difficult to test the ability of a method to perform the clustering since there is no supervision of the process. In this sense, the classification into different groups proposed by the authors in [11-12] is assumed to be the reference partition of samples in our work. This assumption may be questionable in some cases, since the reference groups are not well established. For example, in the HC-Salamanca dataset the AML with-inversion group is established by observation of the karyotype of cancer cells, but there is no other evidence (biological, genetic) suggesting that this group corresponds to a distinct disease.

Table 6. Final clustering computed from the results of the hierarchical clustering algorithm for the Armstrong dataset.

ALL	ALL-01, ALL-02, ALL-03, ALL-04, ALL-05, ALL-06, ALL-07, ALL-08, ALL-09, ALL-10, ALL-11, ALL-12, ALL-13, ALL-14, ALL-15, ALL-16, ALL-17, ALL-18, ALL-19, ALL-20, ALL-58, ALL-59, ALL-60, ALL-61, **MLL-21, MLL-22, MLL-23, MLL-24, MLL-25, MLL-26, MLL-27, MLL-28, MLL-29, MLL-31, MLL-33, MLL-34**
AML	AML-38, AML-39, AML-40, AML-41, AML-42, AML-43, AML-44, AML-45, AML-46, AML-47, AML-49, AML-50, AML-51, AML-52, AML-53, AML-54, AML-55, AML-56, AML-57, AML-65, AML-66, AML-67, AML-68, AML-69, AML-70, AML-71, AML-72
MLL	**AML-48**, MLL-30, MLL-32, MLL-35, MLL-36, MLL-37, MLL-62, MLL-63, MLL-64

Even so, the assumption of these prior partitions as reference groups is the only way to evaluate the similarity (or dissimilarity) of the results computed by the proposed method based on existing knowledge. As it turns out, there is no perfect match among the results of our proposed method and the reference partitions, but they are compatible with the current knowledge of each dataset. For example, for the HC-Salamanca dataset the better characterized groups are the APL and Other-AML groups, the worst is the AML with inversion group, and there is some confusion of the monocytic AML with the AML with-inversion and Other-AML groups. These results are compatible with the state-of-the-art discussed in [11], where the APL group is the better characterized disease (it can be considered as a distinct class), the monocytic AML is a promising disease (in [11] the authors try to show differences in gene expression of this class with regard the others), the AML with inversion in chromosome 16 is the weaker class, and the Other-AML group acts as the dumping ground for the rest of samples which are not similar enough to the other possible classes. For the Armstrong dataset, the AML group is clearly separated from the MLL and ALL groups. It is not surprising since the myeloid leukemia (AML) and lymphoblastic leukaemias (MLL and ALL) represent distinct diseases. Some confusion is present among ALL and MLL groups, but this result is compatible with the assumption (which authors test in [12]) that the MLL group is a subtype of the ALL disease.

Moreover, the results shown in Tables 1 and 4 (by rows) are an estimation of the overlapping area between the *i*-th membership graph (associated with the *i*-th predicted group) and any *j*-th membership graph (see Figures 7 and 8 for a geometrical interpretation) taking into consideration the samples in the *i*-th true cluster. Therefore, according to the affinity graphs shown in Figures 7 and 8, these percentages can be also interpreted as a measure of the similarity/dissimilarity among predicted groups.

4. Conclusions

The simulated annealing-based algorithm presented in this work is a new algorithm for iterative class discovery that uses fuzzy logic for informative gene selection. An intrinsic advantage of the proposed method is that, assuming the percentage of times in which a given microarray has been grouped with samples of other potential classes, the degree of membership of that microarray to each potential group can be deduced. This fact allows a fuzzy clustering of the available microarrays which is more suitable for the current state-of-the-art in gene expression analysis, since it will be very unlikely to state (without uncertainty) that any available microarray only belongs to a unique potential cluster. In this case, the proposed method can help to assess the degree of affinity of each microarray with potential groups and to guide the analyst in the discovery of new diseases.

In addition, the proposed method is also an unsupervised technique for gene selection when it is used in conjunction

with the concept of discriminant fuzzy pattern (DFP) introduced in [13]. Since the selected genes depend on the resulting clustering (they are the genes in the computed DFP obtained from all groups) and the clustering is obtained by maximizing the cost function (which is based on the notion of genetic coherence and assessed by the number of genes in the fuzzy pattern of each cluster), then the selected genes jointly depend on all the genes in the microarray, and the proposed method can be also considered a multivariate method for gene selection.

Finally, the proposed technique, in conjunction with our previous developed geneCBR platform, represents a more sophisticated tool which integrates three main tasks in expression analysis: clustering, gene selection and classification. In this context, all the proposed methods are non-parametric (they do not depend on assumptions about the underlying distribution of available data), unbiased with regard to the basic computational facility used to construct them (the notion of fuzzy pattern) and with the ability to manage imprecise (and hence, uncertain) information, which is implicit in available datasets in terms of degree of membership to linguistic labels (expressions levels, potential categories, etc.).

Acknowledgements

This work has been partially funded by the Spanish Ministry of Science and Innovation, the Plan E from the Spanish Government, the European Union from the ERDF (TIN2009-14057-C03-02), Xunta de Galicia for Angeles Alvariño fellowship to D.G-P., and JCyL (Government of the autonomous community of Castile and León, ref. VA100A08).

References

1. E.P. Xing, R.M. Karp, Bioinformatics 17 (2001) S306-315.
2. [2] D. Jiang D, C. Tang , A. Zhang, IEEE T Knowl Data En 16 (2004) 1370-1386.
3. O. Alter, P.O. Brown, D. Botstein, P Natl Acad Sci USA 97 (2000) 10101-10106.
4. C.H.Q. Ding, in: Proceedings of the Sixth Annual International Conference on Computational Molecular Biology, ACM, 2002, pp. 127-136.
5. K.Y. Yeung, W.L. Ruzzo, Bioinformatics 17 (2001) 763-774.
6. A. Ben-Dor, N. Friedman, Z. Yakhini, in: Proceedings of the Fifth Annual International Conference on Computational Biology, ACM, 2001, pp. 31-38.
7. A. von Heydebreck, W. Huber, A. Poustka, M. Vingron, Bioinformatics 17 (2001) S107-114.
8. C. Tang, A. Zhang, M. Ramanathan, Bioinformatics 20 (2004) 829-838.
9. S. Varma, R. Simon, BMC Bioinformatics,5 (2004) 126.
10. D. Glez-Peña, F. Díaz, J.R. Méndez, J.M. Corchado, F. Fdez-Riverola, Lecture notes in Computer Science 5518 (2009) 972-978.
11. N.C. Gutiérrez, R. López-Pérez, J.M. Hernández, I. Isidro, B. González, M. Delgado, E. Fermiñán, J.L. García, L. Vázquez, M. González, J.F. San Miguel, Leukemia 19 (2005) 402-409.
12. S.A. Armstrong, J.E. Staunton, L.B. Silverman, R. Pieters, M.L. den Boer, M.D. Minden, S.E. Sallan, E.S. Lander, T.R. Golub, S.J. Korsmeyer, Nat Genet 30 (2002) 41-47.
13. F. Díaz, F. Fdez-Riverola, J.M. Corchado, Comput Intell 22 (2006) 254-268.
14. F. Fdez-Riverola, F. Díaz, M.L. Borrajo, J. C. Yáñez, J.M. Corchado, in: H. Muñoz-Avila, F. Ricci (Eds.), Proceedings of the 6th International Conference on Case-Based Reasoning, Springer, 2005, pp. 191-205.
15. D. Glez-Peña, F. Díaz, J.M. Hernández, J.M. Corchado, F. Fdez-Riverola, BMC Bioinformatics 10 (2009) 187.
16. N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, J Chem Phys 21 (1953) 1087-1092.
17. S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Science 220 (1983) 671-680.
18. V. Černý V, J Optimiz Theory App 45 (1985) 41-51.
19. C.R. Reeves CR, Modern heuristic techniques for combinatorial problems, McGraw-Hill , London, 1995.
20. M. Halkidi, Y. Batistakis, M. Vazirgiannis, J Intell Inf Syst 17 (2001) 107-145.