Automatic Clustering Using a Genetic Algorithm with New Solution Encoding and Operators

Carolina Raposo[1], Carlos Henggeler Antunes[2], and João Pedro Barreto[1]

[1] Institute of Systems and Robotics, Dept. of Electrical and Computer Engineering, University of Coimbra, Portugal
{carolinaraposo,jpbar}@isr.uc.pt

[2] INESC Coimbra, Dept. of Electrical and Computer Engineering, University of Coimbra, Portugal
ch@deec.uc.pt

Abstract. Genetic algorithms (GA) are randomized search and optimization techniques which have proven to be robust and effective in large scale problems. In this work, we propose a new GA approach for solving the automatic clustering problem, ACGA - Automatic Clustering Genetic Algorithm. It is capable of finding the optimal number of clusters in a dataset, and correctly assign each data point to a cluster without any prior knowledge about the data. An encoding scheme which had not yet been tested with GA is adopted and new genetic operators are developed. The algorithm can use any cluster validity function as fitness function. Experimental validation shows that this new approach outperforms the classical clustering methods K-means and FCM. The method provides good results, and requires a small number of iterations to converge.

Keywords: Genetic Algorithms, Clustering, Calinski-Harabasz index, K-means, Fuzzy C-Means.

1 Introduction

Clustering is the problem of classifying an unlabeled dataset into groups of similar objects. Each group, called a cluster, consists of objects that are similar between themselves, according to a certain measure, and dissimilar to objects of other groups, with respect to the same measure. Clustering has had an important role in areas such as computer vision [1], data mining and document categorization [2], and bioinformatics [3].

The major difficulty in the clustering problem is the fact that it is an unsupervised task, i.e., in most cases the structural characteristics of the problem are unknown. In particular, the spatial distribution of data in terms of number, volumes, and shapes of clusters is not known.

A classical clustering method is K-means [4], in which the data is divided into a set of K clusters. This division depends on the initial clustering, and is computed by minimizing the squared error between the centroid of a cluster and its elements. One variant of K-means is the Fuzzy C-Means (FCM), in which the elements of the dataset may belong to more than one cluster with different membership degrees [5]. FCM proceeds by updating the cluster centroids and the membership degrees until a convergence condition is attained. Both methods have proven to perform well in many situations. However, they can easily get stuck in local minima, depending on the initial clustering. Moreover, for problems where the number of clusters is unknown a priori, it is necessary to perform the clustering for different values of K and choose the best solution according to a certain evaluation function. This becomes impractical and tedious in the presence of a large number of clusters.

These issues (local minima and unknown number of clusters) can be tackled using GA [6], which have proven to be effective in search and optimization problems [7]. GA are high level search heuristics that mimic the process of natural evolution, based on the principle of survival of the fittest, by using selection, crossover and mutation mechanisms. Solutions (individuals) are encoded as strings, called *chromosomes*, and, at each iteration, the fitness of solutions in the population is computed for determining the evolutionary process.

In [8] a GA was proposed to find the optimal partition of a dataset, given the number of clusters. More recently, methods based on GA have been proposed to solve the automatic clustering problem, i.e., when no information about the dataset is known [9–13].

We propose a new GA approach for solving the automatic clustering problem, ACGA - Automatic Clustering Genetic Algorithm - which uses a solution encoding based on the one presented in [14]. Two new mutation and one new crossover operators have been developed in order to guarantee the existence of high diversity in the population. As fitness function any internal cluster validity measure can be used, for which the Calinski-Harabasz (CH) index [15] was chosen. An experimental validation using real [16] and synthetic [17] datasets was conducted, and the performance of ACGA was compared with the K-means and FCM algorithms. ACGA outperforms these two classical methods, converging to solutions with higher fitness function values.

The interest and motivation of this study have been provided in the introduction. An overview of the GA approach describing the encoding scheme, and genetic operators is presented in Section 2. Experiments and results are discussed in Section 3. Finally, some conclusions are drawn in Section 4.

2 Overview of the GA Approach

In this Section a new GA approach for solving the automatic clustering problem is presented. An encoding scheme based on [14] is used, and new genetic operators (mutation and crossover) are developed.

Solutions represent different points in the search space, to which quality values are assigned, according to a pre-defined fitness function. The best B solutions of the population are preserved, and pass to the next generation, thus endowing the evolutionary process with a certain degree of elitism. A fixed number of solutions is then selected to act as parents. Crossover is applied to the parent chromosomes, originating new solutions, which are then subject to mutation. The new population is evaluated, and this procedure is repeated until a stopping criterion is satisfied.

Algorithm 1 shows the main steps of the GA. In the next Subsections, the encoding scheme is described, as well as the procedures for generating the initial population, evaluating each solution, selecting the parent population, and performing the crossover and mutation operators.

Algorithm 1: Pseudo-Code of the GA

```
$t \leftarrow 0$;
Generate initial population $P(t)$;
Evaluate $P(t)$;
while Stopping criterion not satisfied do
    Select parent population $P'(t)$ from $P(t)$;
    Apply genetic operators to $P'(t) \rightarrow P(t+1)$;
    Replace random solutions in $P(t+1)$ with the best $B$ solutions in
    $P(t)$;
    Evaluate $P(t+1)$;
    $t \leftarrow t+1$;
Result: Best solution (highest fitness value) of the population in the last
        generation.
```

2.1 Encoding Scheme

In this work, we adopted an encoding scheme based on the one used in [14]. Each solution i is a fixed-length string represented by activation values A_{ij} and cluster centroids $m_{ij}, j = 1, \ldots, K_{max}$, where K_{max} is the maximum number of clusters defined by the user. Then, solution i of the population is represented as:

A_{i1}	A_{i2}	$\ldots$	$A_{iK_{max}}$	m_{i1}	m_{i2}	$\ldots$	$m_{iK_{max}}$

where A_{ij} are activation values defined in the interval $[0, 1]$. An activation value A_{ij} larger than 0.5 means that the cluster with centroid m_{ij} is active, i.e., it is selected for partitioning the dataset. Otherwise, the respective cluster is inactive in chromosome i.

Since each centroid m_{ij} is a d-dimensional vector, where d is the number of features of the data, each solution is a string with total length equal to $K_{max} + dK_{max}$. As an example, for a maximum number of clusters $K_{max} = 3$ and a dataset with $d = 2$ features, the string

0.1	**0.8**	**0.6**	10	0.5	**11**	**0.9**	**14**	**0.3**

represents a solution i with two activated clusters with centroids $m_{i2} = [11, 0.9]$ and $m_{i3} = [14, 0.3]$.

This encoding scheme has advantages over the more popular label-encoding, where each position is an integer value corresponding to one cluster. These include the fact that it may lead to smaller solution vectors, in the presence of large datasets, and solutions with isolated one-element clusters are more difficultly generated, since the labeling of each data point only depends on the location of the centroids. However, this encoding has the disadvantage of requiring one label to be assigned to each data point, before computing the fitness of each solution. This is done by finding the active centroid closest to each point (smallest euclidean distance).

2.2 Initialization of the Population

Each candidate solution in the population is initialized independently, with the activation values and the cluster centroids initialized separately.

The K_{max} activation values A_{ij} are initialized as random values drawn from the standard uniform distribution in the interval $[0, 1]$. In order to guarantee that each initial solution has at least 2 active clusters, a minimum of 2 activation values are forced to be larger than 0.5.

For initializing the K_{max} centroids m_{ij}, the interval in which the data points are contained is determined. More specifically, for each dimension relative to each feature, the minimum and the maximum values in the dataset are determined. Then, for each dimension, K_{max} random values are sampled from the corresponding interval, originating K_{max} points that are the initial centroids.

The size of the population P (number of candidate solutions) is a user-defined value.

2.3 Solution Evaluation

A quality measure is assigned to each solution, which is computed using a fitness function. In this work we chose to define the fitness function as the Calinski-Harabasz (CH) index [15], which uses the quotient between the intra-cluster average squared distance and the inter-cluster average squared distance. For solution s, $\mathsf{CH}(s)$ is computed as in equation (1):

$$\mathsf{CH}(s) = \frac{\sum_{i=1}^{N} ||X_i - m_s||^2 - \sum_{j=1}^{K_s} \sum_{X \in C_{sj}} ||X - m_{sj}||^2}{\sum_{j=1}^{K_s} \sum_{X \in C_{sj}} ||X - m_{sj}||^2} \frac{N - K_s}{K_s - 1}, \tag{1}$$

where N is the number of objects X_i in the dataset, m_s is the centroid of the dataset, K_s is the number of clusters in solution s, and X are the objects in cluster C_{sj} with centroid m_{sj}. $|| \cdot ||$ denotes the L2 norm. Larger inter-cluster distances and smaller intra-cluster distances lead to larger CH values, i.e., higher quality solutions.

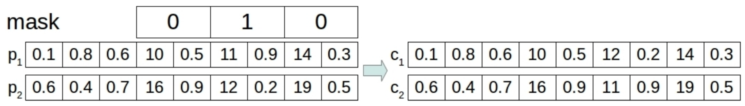


Fig. 1. Example of the crossover between two parent solutions p_1 and p_2, originating two offspring c_1 and c_2. The solutions correspond to maximum number of clusters $K_{max} = 3$ and $d = 2$ features.

Note that for one-cluster solutions, the term $K_s - 1$ is substituted by 1, and we have the relation

$$\sum_{i=1}^{N} ||X_i - m_s||^2 = \sum_{j=1}^{K_s} \sum_{X \in C_{sj}} ||X - m_{sj}||^2, \tag{2}$$

leading to a CH value of 0. Thus, it is not necessary to explicitly deal with one-cluster solutions since they are automatically assigned a null quality value.

2.4 Selection of the Parent Solutions

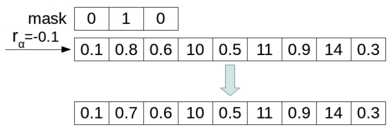
Before applying the genetic operators, a set of parent solutions must be selected. This is done in a tournament scheme by selecting P_p random solutions from the population and finding the one with the highest fitness, which will act as a parent solution. This procedure is repeated P times, originating a parent population of the same size as the original population.

2.5 Genetic Operators

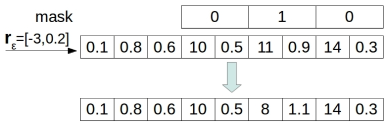
The genetic operators should be suited to the encoding scheme. Their objective is to ensure diversity in the population, such that better solutions can be produced through the evolutionary process. In this work, one new crossover scheme is presented, as well as two new mutation operators.

Crossover. For each pair of parent solutions in the population crossover is applied, originating two offspring. A binary mask of length K_{max} is created, where 1 occurs with a probability p_{cross}. The centroids in the parent solutions corresponding to the positions in the mask with a unitary value are exchanged, originating two children. Figure 1 shows an example of the crossover between parent solutions p_1 and p_2 yielding offspring c_1 and c_2.

Mutation. Each solution produced by the crossover operator goes through a mutation process. Two different types of mutation have been considered: mutation in the activation values, and mutation in the centroids.



(a) An example of the activation value mutation.



(b) An example of the centroid mutation.

Fig. 2. Examples of the two different types of mutation proposed in this work: mutation in the activation values, and centroid mutation. The solution vectors correspond to maximum number of clusters $K_{max} = 3$, and $d = 2$ features.

Activation Value Mutation

Similarly to the crossover operator, a binary mask of length K_{max} is created, where 1 occurs with a probability p_{mut}. Let α be a user-defined parameter that controls the perturbation of the activation values. For each unitary value in the mask, a random value r_α is drawn in the interval $[-\alpha, \alpha]$. The activation value in the corresponding position is modified by adding r_α: $A'_{ij} = A_{ij} + r_\alpha$, where A'_{ij} is the modified activation value.

If the resulting activation value A'_{ij} does not belong to the interval $[0, 1]$, it is forced to 0 or 1:

$$A'_{ij} = \begin{cases} 1 & \text{if } A'_{ij} > 1 \\ 0 & \text{if } A'_{ij} < 0 \end{cases}. \tag{3}$$

Figure 2(a) shows an example of the application of mutation to the activation values. The centroid values remain unaltered, and only randomly chosen activation values, with probability p_{mut}, are altered.

Since the crossover operator does not influence the activation values, this type of mutation is the only possible way to achieve diversity in the number of clusters between solutions in consecutive iterations of the algorithm.

Centroid Mutation

The mutation operator applied to the centroids is similar to the activation value mutation, with the difference that the perturbation is a d-dimensional vector created according to the range of values of each feature of the data points.

Let ϵ be a user-defined parameter that controls the change in amplitude of the centroids. For each dimension w of the data points, $w = 1, \ldots, d$, let $g_w = h_w - l_w$ be the difference between the largest value relative to feature w found in the dataset, h_w, and the smallest value relative to the same feature, l_w. By defining $t_w = \epsilon g_w$ for each feature, a vector $\mathbf{r}_\epsilon = [r_{\epsilon 1}, \ldots, r_{\epsilon d}]$ can be created by drawing random values $r_{\epsilon w}$ from the intervals $[-t_w, t_w]$.

The centroids m_{ij} that correspond to unitary values in the binary mask (created as in the activation value mutation scheme) are modified by adding the perturbation vector $\mathbf{r}_\epsilon$: $m'_{ij} = m_{ij} + \mathbf{r}_\epsilon$, where m'_{ij} is the modified centroid.

Again, if the new centroid exceeds the boundary values of the data point values in any dimension, it is forced to the corresponding boundary value:

$$m'_{ijw} = \begin{cases} h_w & \text{if } m'_{ijw} > h_w \\ l_w & \text{if } m'_{ijw} < l_w \end{cases}, \forall w = 1, \ldots, d. \tag{4}$$

Figure 2(b) shows an example of the application of mutation to the centroids. In this case, only randomly chosen centroids are altered.

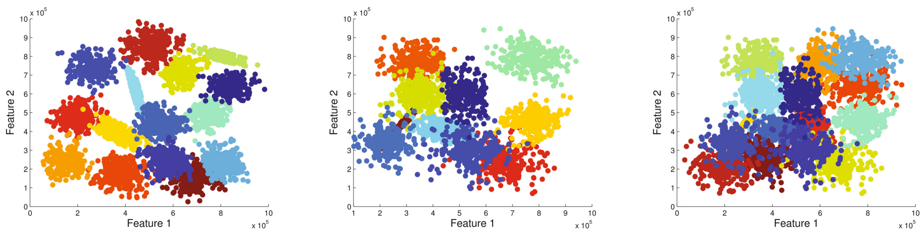
The activation value mutation may drastically change the number of clusters in a solution, originating child solutions considerably different from the parent solutions. This may be undesirable, since it easily leads to solutions with significantly lower fitness values. Thus, this type of mutation is applied less often than the centroid mutation. The algorithm proceeds until either a maximum number of iterations I_{max}, or a fixed number of iterations without improvement of the best solution in the population I_{ni} are reached.

3 Experiments and Results

In this Section, we compare the performance of ACGA with two classical clustering methods: K-means [4] and FCM [5]. Since these two methods require the user to input the number of clusters, we performed the clustering for a varying number of clusters, and chose the solution with the highest quality value, using the same fitness function used in the GA (CH index). We tested for $K = 2, \ldots, K_{max}$, with $K_{max} = 20$, clusters. Experiments were performed using the datasets in Table 1. The first three are synthetic datasets downloaded from [17], presenting different degrees of cluster overlapping, as can be seen in Figure 3. The remaining sets are real-life datasets, which were obtained from [16], and are commonly used for the validation of clustering methods. The parameters refer to the original labeling provided with the datasets. Table 1 shows the number of clusters K, the number of objects N, and the number of features d for each dataset.

Table 1. Description of the datasets used in the experiments. K is the number of clusters, N is the number of data points, and d is the number of features.

Dataset	K	N	d	Avg Intra Cluster Dist.	Avg Inter Cluster Dist.	CH value
S2	15	5000	2	42294.60	182459.12	12541
S3	10	3254	2	54507.97	174914.69	5002
S4	15	5000	2	55312.63	149595.93	5385
Iris	3	150	4	0.67	2.15	486
B. Cancer	2	638	9	2.66	4.65	303
Seeds	3	210	7	1.58	3.85	310



(a) Dataset S2 (b) Dataset S3 (c) Dataset S4

Fig. 3. The three synthetic datasets used in the experiments, presenting different degrees of cluster overlapping. Colors are used to identify the clusters.

Table 2. User-defined values used in the experiments

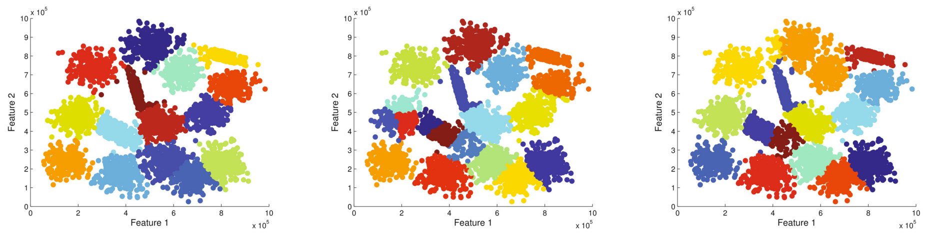
Maximum number of clusters, K_{max}	20
Population size, P	40
No. of solutions chosen for parent selection, P_p	3
Crossover probability, p_{cross}	0.3
Mutation probability, p_{mut}	0.1
Activation value mutation parameter, α	0.1
Centroid mutation parameter, ϵ	0.3
No. of elite solutions, B	4
Maximum no. of iterations, I_{max}	5000
Maximum no. of iterations without improvement, I_{ni}	500

Moreover, the average intra-cluster and inter-cluster distances were computed. For each cluster, the intra-cluster distance is the mean of the distances of all data points to the cluster centroid. The inter-cluster distance is computed as the minimal distance between cluster centroids, for each cluster. Table 1 also shows the CH value, computed using equation (1). For all the datasets, 20 independent runs of each method were performed. Since K-means and FCM depend on the initial clustering which is obtained by randomly sampling the dataset, and generating random membership values, respectively, slightly different results may be obtained in different runs. For ACGA, the values in Table 2 were assigned to the parameters described in Section 2. Table 3 shows the results obtained for all the datasets in Table 1, using K-means, FCM, and ACGA. The last column of Table 3 presents the values for the Adjusted Rand (AR) index [18], which is a measure of agreement between two partitions and is frequently used in cluster validation. We computed the AR index between the original labelings and the ones obtained with each method. This index indicates how similar the partition obtained is to the original one. For two random partitions, the expected value of the AR index is 0, and its maximum value is 1, obtained when comparing equal partitions.

Table 3. Results obtained over 20 independent runs using ACGA, K-means, and FCM. The first line of each dataset corresponds to the "ground truth" values in Table 1.

Dataset	Method	Avg. No. Clu.	Avg. Inter Cluster	Avg. CH	Avg. AR
S2	-	15	182459.12	12541	-
	ACGA	15.000±000	183081.207±316.112	13461.515±420.630	0.937±0.003
	K-means	16.967±1.426	148383.088±10516.459	10658.228±1697.369	0.859±0.048
	FCM	15.233±0.430	171068.182±5657.558	13106.362±924.506	0.926±0.027
S3	-	10	174914.69	5002	-
	ACGA	10.000±0.000	177104.685±420.960	6307.633±16.153	0.806±0.003
	K-means	10.267±1.112	173354.951±15347.364	5482.044±584.628	0.736±0.042
	FCM	10.133±0.434	170345.561±9489.542	6091.281±387.783	0.785±0.041
S4	-	15	149595.93	5385	-
	ACGA	15.000±0.000	152034.821±358.558	7865.943±12.925	0.724±0.003
	K-means	15.933±1.460	138291.741±7920.736	6788.132±605.469	0.664±0.035
	FCM	15.567±0.626	141600.223±4524.683	7570.782±339.299	0.713±0.019
Iris	-	3	2.15	486	-
	ACGA	3.00±0.000	2.185±0.000	506.297±0.000	0.886±0.000
	K-means	3.067±0.450	1.932±0.319	473.496±75.401	0.791±0.158
	FCM	3.00±0.000	1.940±0.000	506.297±0.000	0.886±0.000
Breast Cancer	-	2	4.65	303	-
	ACGA	2.000±0.000	11.793±1.117	1026.084±0.351	0.847±0.004
	K-means	2.000±0.000	13.816±0.000	1026.262±0.000	0.846±0.000
	FCM	2.000±0.000	13.886±0.000	1023.939±0.000	0.830±0.000
Seeds	-	3	3.85	310	-
	ACGA	3.000±0.000	3.107±0.153	329.191±3.108	0.704±0.012
	K-means	3.000±0.000	3.988±0.000	328.267±0.000	0.706±0.000
	FCM	3.000±0.000	3.970±0.000	329.918±0.000	0.694±0.000

By comparing the AR index in Table 3, it can be seen that ACGA generally outperforms both K-means and FCM. As an example, an AR value of 0.886 obtained for the iris dataset means that 6 data points were misclassified, corresponding to 4%. The superiority of ACGA is clear by comparing the results obtained for all synthetic datasets, where both K-means and FCM failed to partition the dataset in the correct number of clusters in every run. Figure 4 shows a partition of the S2 dataset obtained with the three methods, where each cluster is identified by a color. Comparing the results to the original clusters in Figure 3(a), it can be seen that our solution (Figure 4(a)) yielded the correct number of clusters, with only a few misclassified objects. K-means (Figure 4(b)) performed poorly since it was not capable of finding the optimal number of clusters. FCM (Figure 4(c)) produced a result considerably better than K-means. However, it still failed to correctly classify many data points. Moreover, ACGA leads to smaller intra-cluster distances and larger inter-cluster distances, and consequently larger CH values. Due to cluster overlapping, the CH values computed using the original partitions (Table 1) are, in some cases, lower than the ones obtained with ACGA. Also, it presents much smaller standard deviations than both K-means and FCM, especially in the synthetic datasets. This means that ACGA is capable of producing stable results, independently of the initialization, converging to very similar solutions in different runs.



(a) ACGA. (b) K-means. (c) Fuzzy C-Means.

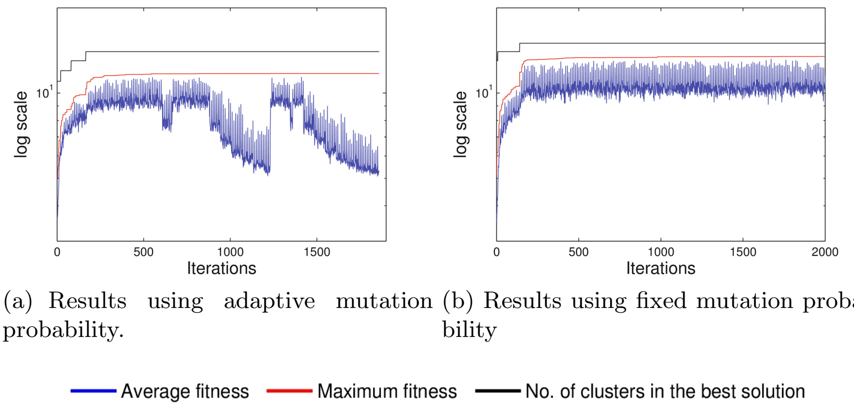
Fig. 4. Results obtained for dataset S2 (refer to Figure 4) using ACGA, K-means, and FCM. Clusters are identified by colors.

Table 4. Average number of iterations required to reach the results in Table 3

Dataset	S2	S3	S4	Iris	B. Cancer	Seeds
Iterations	2936.5	2199.5	3316.0	15.6	588.5	353.3

Table 4 shows the average number of iterations necessary to achieve the solutions in Table 3. The values were obtained by subtracting the number of iterations without improvement of the best solution in the population, I_{ni}, to the total number of iterations. It can be seen that for smaller and "well-behaved" datasets, ACGA is fast and converges after a small number of iterations. For larger datasets, such as S2, it requires approximately 3000 iterations.

Generally, using parameters such as the mutation probability that adapt to the evolution of the algorithm is a good technique since higher diversity in the population can be achieved, preventing the algorithm to get stuck in local minima. Thus, we tested the influence of an increase in p_{mut} triggered by reaching a certain number of iterations without improvement of the best solution in the population. Results obtained with dataset S2 are shown in Figure 5(a), which plots the average (blue) and maximum (red) quality of the population, as well as the number of clusters of the best solution (black), in each iteration. It can be seen that, despite increasing diversity in the population, applying mutation at higher rates leads to a decrease in the average quality of population, which is clear by observing the evolution in the last 500 iterations. Thus, we kept all parameters fixed after a tuning process, and the algorithm behaves as depicted in Figure 5(b), where the average quality curve presents always the same pattern. The maximum quality in the population does not decrease due to elitism: the best B solutions in each generation pass to the next generation unaltered. This moderate degree of selective pressure due to elitism contributed to improve the results without reducing exploration. Also, the high population diversity is evinced by the significant changes in the average fitness in consecutive iterations of the algorithm. The curves show that significant changes in the maximum quality are generally caused by changes in the number of clusters.



(a) Results using adaptive mutation probability.

(b) Results using fixed mutation probability

Fig. 5. Evolution of the fitness and number of clusters of the solutions in the population for dataset S2. The figures show the maximum (red) and average (blue) fitnesses of the population, as well as the number of clusters of the best solution in each iteration (black). Fitness values have been scaled by a factor of 10^{-3}.

4 Conclusion

We present a new GA approach, ACGA, for automatically finding the optimal number of clusters in real and synthetic datasets, with different degrees of overlapping, and correctly assigning each data point to a cluster. It uses an encoding scheme that, to the best of our knowledge, had never been incorporated into GA. This required the development of new genetic operators that ensure diversity in the population. ACGA does not require any information about the data, and is able to outperform two classical clustering methods: K-means and FCM. Experiments included three synthetic and three real datasets, and results show that ACGA leads to partitions very similar to the original ones, requiring a small number of iterations to converge.

Acknowledgments. This work was financially supported by a PhD grant (SFRH/BD/88446/2012) from the Portuguese Foundation for Science and Technology (FCT). The author Carlos Henggeler Antunes acknowledges the support of FCT project PEst-OE/EEI/UI0308/2014 and QREN Mais Centro Program iCIS project (CENTRO-07-ST24-FEDER-002003).

References

1. Belahbib, F., Souami, F.: Genetic algorithm clustering for color image quantization. In: 3rd European Workshop on Visual Information Processing (EUVIP), pp. 83–87 (2011)

2. Mecca, G., Raunich, S., Pappalardo, A.: A New Algorithm for Clustering Search Results. Data and Knowledge Engineering 62, 504–522 (2007)
3. Valafar, F.: Pattern Recognition Techniques in Microarray Data Analysis: A Survey. Annals of New York Academy of Sciences 980, 41–64 (2002)
4. Hartigan, J., Wong, M.: Algorithm AS 136: A K-Means Clustering Algorithm. Applied Statistics 28(1), 100–108 (1979)
5. Bezdek, J., Ehrlich, R., Full, W.: FCM: The fuzzy c-means clustering algorithm. Computers and Geosciences 10(2-3), 191–203 (1984)
6. Holland, J.: Genetic algorithms. Scientific American (1992)
7. Srinivas, M., Patnaik, M.: Genetic algorithm: A survey. IEEE Computer 27(6), 17–26 (1994)
8. Murthy, C., Chowdhury, N.: In search of optimal clusters using GA. Pattern Recognition Letters 17, 825–832 (1996)
9. Tseng, L., Yang, S.: A genetic approach to the automatic clustering problem. Pattern Recognition 34(2), 415–424 (2001)
10. Agustin-Blas, L., Salcedo-Sanz, S., Jimenez-Fernandez, S., Carro-Calvo, L., Del Ser, J., Portilla-Figueras, J.A.: A new grouping GA for clustering problems. Expert Systems with Applications 39(10) (2012)
11. Sheikh, R., Raghuwanshi, M., Jaiswal, A.: Genetic Algorithm Based Clustering: A Survey. In: First International Conference on Emerging Trends in Engineering and Technology, vol. 2(6), pp. 314–319 (2008)
12. Liu, Y., Wu, X., Shen, Y.: Automatic clustering using genetic algorithms. Applied Mathematics and Computation 218(4), 1267–1279 (2011)
13. He, H., Tan, Y.: A two-stage genetic algorithm for automatic clustering. Neurocomputing 81, 49–59 (2012)
14. Das, S., Abraham, A., Konar, A.: Automatic Clustering Using an Improved Differential Evolution Algorithm. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 38(1), 218–237 (2008)
15. Calinski, R., Harabasz, J.: A dendrite method for cluster analysis. Communications in Statistics 3(1), 1–27 (1974)
16. Asuncion, A., Newman, J.: UCI Machine Learning Repository. University of California, Department of Information and Computer Science, Irvine, CA (2007), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
17. Speech and Image Processing Unit. Clustering datasets, `http://www.cs.joensuu.fi/sipu/datasets/`
18. Hubert, L., Arabie, P.: Comparing Partitions. Journal of Classification (2), 193–218 (1985)