Clustering using Cellular Genetic Algorithms

Nuno Leite[1,3], Fernando Melício[3] and Agostinho C. Rosa[2,3]

[1]*Instituto Superior de Engenharia de Lisboa/ADEETC, Polytechnic Institute of Lisbon, Rua Conselheiro Emídio Navarro, n.° 1, 1959-007, Lisboa, Portugal*

[2]*Department of Bioengineering/Instituto Superior Técnico, Universidade de Lisboa, Av. Rovisco Pais, n.° 1, 1049-001, Lisboa, Portugal*

[3]*Institute for Systems and Robotics/LaSEEB, Instituto Superior Técnico, Universidade de Lisboa, Av. Rovisco Pais, n.° 1, TN 6.21, 1049-001, Lisboa, Portugal*

Keywords: Cellular Genetic Algorithms, Clustering, Classification, Evolutionary Computation, Nature Inspired Algorithms.

Abstract: The goal of the clustering process is to find groups of similar patterns in multidimensional data. In this work, the clustering problem is approached using cellular genetic algorithms. The population structure adopted in the cellular genetic algorithm contributes to the population genetic diversity preventing the premature convergence to local optima. The performance of the proposed algorithm is evaluated on 13 test databases. An extension to the basic algorithm was also investigated to handle instances containing non-linearly separable data. The algorithm is compared with nine non-evolutionary classification techniques from the literature, and also compared with three nature inspired methodologies, namely Particle Swarm Optimization, Artificial Bee Colony, and the Firefly Algorithm. The cellular genetic algorithm attains the best result on a test database. A statistical ranking of the compared methods was made, and the proposed algorithm is ranked fifth overall.

1 INTRODUCTION

Clustering is an important unsupervised learning approach used in applications in areas such as data mining, statistical data analysis, data compression, or vector quantization. The task of clustering is to discover natural groupings of a set of patterns, points, or objects (Jain, 2010).

Clustering approaches are generally categorized into two classes (Jain, 2010): *hierarchical* and *partitional*. In hierarchical clustering, nested clusters are found recursively according to two modes, agglomerative and divisive. In the *agglomerative* mode, the clustering process starts with each data point in its own cluster and then proceeds merging the most similar pair of clusters successively to form a cluster hierarchy. In the *divisive* (top-down) mode, the clustering starts with all the data points in one cluster and then proceeds by recursively dividing each cluster into smaller clusters. Partitional clustering algorithms work differently compared to hierarchical clustering algorithms as they find all the clusters at the same time as a partition of the data without imposing a hierarchical structure. The prototype-based clustering algorithms are the most popular partitional clustering algorithms. In these, each cluster is represented by its center and the objective function used is the sum of squared distances of the data points to the cluster centers they are assigned to (Mirkin, 1996; Borgelt, 2006). An example of a partitional clustering algorithm is the popular k–means algorithm. Although still widely used, the main drawback of the k–means algorithm is that it converges to a local minima from the starting position of the search (Selim and Ismail, 1984).

The clustering process can be done through two ways: by unsupervised learning or by supervised learning. In unsupervised clustering, also called automatic clustering, there's no information about the classes of the patterns, and the clustering is done by finding the natural aggregations in the dataset. On the other way, in supervised clustering, also known as classification, a classifier is learned given a training set with known pattern classes. After the training phase, a test set is classified based on the learned classifier. The learned classifier contains information of the optimal cluster centers that characterize the partitions in the training set. These cluster centers are used to classify the patterns in the test set.

In the recent years, nature inspired algorithms

Leite, N., Melício, F. and Rosa, A..
Clustering using Cellular Genetic Algorithms.
In *Proceedings of the 7th International Joint Conference on Computational Intelligence (IJCCI 2015) - Volume 1: ECTA*, pages 366-373
ISBN: 978-989-758-157-1

have been proposed to solve clustering and classification problems. Examples include Genetic Algorithms (Falkenauer, 1998; Maulik et al., 2000), Memetic Algorithms (Ni et al., 2013), Artificial Bee Colony (ABC) (Karaboga and Ozturk, 2011), Particle Swarm Optimization (PSO) (Falco et al., 2007), Firefly Algorithm (FA) (Senthilnath et al., 2011), and Cuckoo Search Algorithm (Zhao et al., 2014). Recently, hybrid evolutionary algorithms which combine evolutionary algorithms with the k–means algorithm were also proposed (Niknam and Amiri, 2010).

In the study presented in this work, a novel approach of cellular genetic algorithms (cGA) (Alba and Dorronsoro, 2008) is proposed for solving the clustering problem. The cGA is based on the parallel cellular model which maintains a greater population diversity compared to the standard evolutionary algorithm, thus contributing to find solutions with better quality (Alba and Dorronsoro, 2008). To the extent of our knowledge, cellular evolutionary algorithms have not yet been applied successfully to the clustering problem. The cGA is applied to classification benchmark problems on 13 typical test databases. The cGA is compared with nine non-evolutionary classification techniques from the literature, and also compared with three nature inspired methodologies, namely PSO (Falco et al., 2007), ABC (Karaboga and Ozturk, 2011), and the FA (Senthilnath et al., 2011).

The remaining of the paper is organized as follows. In Section 2, the clustering problem is formally defined. Section 3 presents the canonical cellular evolutionary algorithm. In Section 4, the adaptation of the cGA for classification is described. Section 5 reports the experimental results. Some conclusions are drawn in Section 6.

2 CLUSTERING AND CLASSIFICATION PROBLEMS

Given a set of multidimensional data, the clustering process consists in finding groups of patterns, or clusters, in the data whose members are more similar to each other than they are to other patterns (Duda et al., 2000). The clustering is done based on some similarity measure (Jain et al., 1999), where the measure based on the distance is generally used.

The clustering problem can be stated as an optimization problem, as in (Marinakis et al., 2008). In order to formulate the clustering problem the following symbols were defined:

- K, is the number of clusters.
- N, is the number of objects.
- $x_i \in \mathcal{R}^n, (i = 1,\ldots,N)$ is the location of the ith pattern.
- $z_j \in \mathcal{R}^n, (j = 1,\ldots,K)$ is the center of the jth cluster, computed as:

$$z_j = \frac{1}{N_j} \sum_{x_i \in C_j} x_i \tag{1}$$

where N_j is the number of objects in the jth cluster denoted by C_j.

The optimization problem is formulated as:

$$\text{Minimise} \quad J(w,z) = \sum_{i=1}^{N} \sum_{j=1}^{K} w_{ij} \parallel x_i - z_j \parallel^2 \tag{2}$$

Subject to

$$\sum_{j=1}^{K} w_{ij} = 1,\ i = 1,\ldots,N \tag{3}$$

$$w_{ij} = 0 \text{ or } 1,\ i = 1,\ldots,N,\ j = 1,\ldots,K. \tag{4}$$

In Eq. (1), each z_j cluster center is computed as the centroid of the region formed by the cluster objects. The goal of the clustering problem, as given in Eq. (2), is to find the clusters centers z_j, such that the sum of the squared Euclidean distances between each pattern and the center of its cluster is minimized.

As mentioned in the introduction, the related problem known as classification is solved in this work. To assess the quality of the classification, the performance measure – classification error percentage (CEP) – was used:

$$CEP = 100 \cdot \frac{\text{Number of misclassified instances}}{\text{Total size of test set}}. \tag{5}$$

3 CELLULAR GENETIC ALGORITHMS

The cellular evolutionary algorithm (cGA) used to solve the Clustering problem relies on the parallel cellular model. In this model, the populations are organized in a special structure defined as a connected graph, in which each vertex is a solution that communicates with its neighbours (Alba and Dorronsoro, 2008). More specifically, individuals are set in a toroidal mesh and are only allowed to recombine with the close neighbours (Figure 1). The population structure used in cellular evolutionary algorithms contributes to the population genetic diversity thus avoiding the premature convergence to a local optimum.

Algorithm 1 depicts the steps of the general cellular genetic algorithm. First, the initial population is

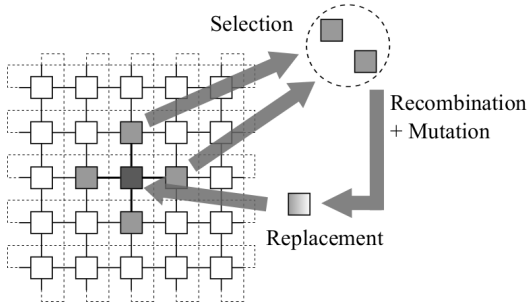

Figure 1: The parallel cellular model for evolutionary algorithms.

generated and evaluated. Then, while the stop condition is not achieved (e.g., maximum number of iterations is achieved, fitness value of the best individual is within a desired interval), the following steps are executed sequentially for each population's individual: (1) obtain the individual's neighbours according to a specified neighbourhood, (2) select individuals to recombine from the neighbours' set, (3) recombine (crossover and mutation) the selected offspring, and (4) put the best offspring in an auxiliary population. After doing steps (1) to (4) for all the elements of the population, the auxiliary population replaces the original population. In this version of the cGA, all the individuals are updated at the same time, so the the algorithm is called *synchronous* cGA. Figure 2 illustrates six typical neighbourhoods used in the cGA.

Algorithm 1: Pseudo-code of the canonical cGA.

```
 1: procedure cGA(cga)        // Parameters in 'cga'
 2:     GenerateInitialPopulation(cga.pop);
 3:     Evaluation(cga.pop);
 4:     while !StopCondition() do
 5:         for indiv ← 1 to cga.popSize do
 6:             neighs ← GetNeighs(cga, pos(indiv));
 7:             par ← Selection(neighs);
 8:             offs ← Recombination(cga.Pc, par);
 9:             offs ← Mutation(cga.Pm, offs);
10:             Evaluation(offs);
11:             Replace(pos(indiv), offs, auxPop);
12:         end for
13:         cga.pop ← auxPop;
14:     end while
15: end procedure
```

In the implemented approach, it was used the L5 neighbourhood type, also known as the von Neumann neighbourhood. This neighbourhood promotes a smother actualisation of the populations compared with the other neighbourhoods. In the selection procedure (Figure 1), one of the four neighbours of the central individual is selected with a binary tournament. The chosen neighbour and the middle individual are then recombined producing two offspring solutions. A mutation operator is then applied to these two solutions and the best offspring is selected. The offspring solution is compared with the original individual, and, if it is better, it will replace the original solution. Otherwise, the original solution is maintained.

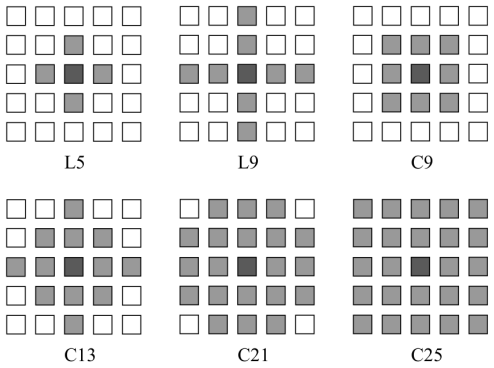

Figure 2: Typical neighbourhoods used in cellular evolutionary algorithms.

4 CGA APPLIED TO CLASSIFICATION

In this section, the cGA adaptation to tackle classification problems is described.

4.1 Encoding

Given an instance with C classes and N attributes, the classification problem considered consists in finding the optimal positions of the C centroids in the N-dimensional space. The i-th individual of the population is encoded as follows:

$$(\vec{p}_i^{\,1}, \ldots, \vec{p}_i^{\,C}) \tag{6}$$

where the j-th centroid position is formed by N real numbers representing the coordinates in the N-dimensional space (Falco et al., 2007):

$$\vec{p}_i^{\,j} = \{p_{1,i}^j, \ldots, p_{N,i}^j\}. \tag{7}$$

So, each population individual is composed by $C * N$ components, each represented by a real number.

4.2 Fitness Function

Each solution i is evaluated using the weighted fitness function $\psi_3(i)$ from (Falco et al., 2007):

$$\psi_3(i) = \frac{1}{2}\left(\frac{\psi_1(i)}{100.0} + \psi_2(i)\right), \tag{8}$$

where $\psi_1(i)$ is given by

$$\psi_1(i) = \frac{100.0}{D_{Train}} \sum_{j=1}^{D_{Train}} \delta(\vec{x}_j) \quad (9)$$

$$\delta(\vec{x}_j) = \begin{cases} 1 & \text{if } \mathrm{CL}(\vec{x}_j) \neq \mathrm{CL}_{\mathrm{known}}(\vec{x}_j), \\ 0 & \text{otherwise.} \end{cases} \quad (10)$$

and $\psi_2(i)$ is given by

$$\psi_2(i) = \frac{1}{D_{Train}} \sum_{j=1}^{D_{Train}} d\left(\vec{x}_j, \vec{p}_i^{\,\mathrm{CL}_{\mathrm{known}}(\vec{x}_j)}\right). \quad (11)$$

As described in (Falco et al., 2007), the function $\psi_1(i)$ is computed in two steps. In the first step, the training set patterns are assigned to the class of the closest centroid in the N-dimensional space. Then, the fitness of the i-th individual is computed as the percentage of incorrectly assigned instances on the training set, i.e., counts the number of patterns $\vec{x}_j$ for which the assigned class $\mathrm{CL}(\vec{x}_j)$ is different from the true class $\mathrm{CL}_{\mathrm{known}}(\vec{x}_j)$. The function $\psi_2(i)$ is computed in one step as the sum of the Euclidean distances between each pattern $\vec{x}_j$ in the training set and the centroid of the true cluster ($\vec{p}_i^{\,\mathrm{CL}_{\mathrm{known}}(\vec{x}_j)}$).

When computing the distance, the components of the i-th individual are normalized using the Min-Max Normalization.

4.3 Population Initialization

The components of each individual in the population are initialized with random numbers in the range $[0,1]$ generated with uniform distribution. Notice that the components could have been initialized with any random numbers, or even with the components of random chosen patterns. Empirical tests were undertaken with different ranges ($[0,1]$, $[0,5]$, $[0,50]$, and components of random patterns). The results showed that there are not significant differences between the methods.

4.4 Crossover

Given two parent solutions, the crossover recombination process exchanges, in a probabilistic fashion, the information between the parent chromosomes producing two offspring chromosomes. In this work, single-point crossover with crossover probability P_c was used.

4.5 Mutation

The mutation operator used was based on that published in (Maulik et al., 2000). For a fixed probability P_m, a gene is selected randomly. A number δ in the range $[-1,1]$ is generated with uniform distribution. If the value at a gene position is v, after mutation it becomes

$$v = \begin{cases} v \cdot (1 + 2 \cdot \delta), & v \neq 0, \\ v + 2 \cdot \delta, & v = 0. \end{cases} \quad (12)$$

With this mutation, the value of the selected gene, v, is increased or decreased in small random amounts.

Table 1: Properties of the examined databases.

	N	D	C	miss
Balance	4	625	3	No
Credit	15	690	2	Yes
Dermatology	34	366	6	Yes
Diabetes	8	768	2	No
E.Coli	7	327	5	No
Glass	9	214	6	No
Heart	13	303	2	Yes
Horse Colic	27	368	2	Yes
Iris	4	150	3	No
Thyroid	5	215	3	No
WDBCancer	30	569	2	No
WDBCancer-Int	9	699	2	Yes
Wine	13	178	3	No

4.6 Handling Non-linearly Separable Problems

As pointed out by De Falco et al. (Falco et al., 2007) in their work, the use of centroids to approach the classification problem is not adequate in some situations. The authors list three cases in which an algorithm based on the concept of centroids and distance to separate the samples into the correct classes is very difficult or even impossible. The first one is when instances of a class are located in two distinct regions of the N-dimensional space, and these instances are separated by instances of a second class. The second situation occurs when a region comprising instances of a class are completely surrounded by a region containing instances belonging to another class. The third one is related to the existence of confusion areas in the search space, i.e., parts of the search space containing examples belonging to two or more classes. The first two cases could be solved by considering two or more centroids per class. The third case cannot be solved effectively with the centroid approach.

To solve the first two cases mentioned in the previous paragraph, the cGA chromosome was extended by considering K clusters per class ($K \geq 1$).

5 EXPERIMENTS AND RESULTS

5.1 Settings

The evaluation of the proposed cGA was carried out on 13 well known databases taken from the UCI Machine Learning Repository (Lichman, 2013). The characteristics of the 13 examined UCI databases are depicted in Table 1. For each database it is indicated the number N of attributes composing each instance, the total instance number D, the number of classes C into which is divided, and a flag *miss* indicating if there is missing data in the database. More information about the test databases can be obtained from (Falco et al., 2007).

The algorithm was programmed in the C++ language and was based on the ParadisEO framework (Cahon et al., 2004). The experiments were made on an Intel Core i7-2630QM (CPU @ 2.00 GHz with 8 GB RAM) PC running Ubuntu 14.04 LTS – 64 bit OS. The Linux kernel used was: 3.13.0-49-generic. The compiler used was: GCC v. 4.8.2.

In this section, all the statistical tests were performed with a 95% confidence level. The population size was set to 100 individuals distributed on a 5×20 rectangular grid. The neighbourhood used was the von Neumann neighbourhood. The crossover and mutation probabilities, respectively, P_c and P_m, were set equal to 0.6 and 0.1. The number of generations of the cGA was set to 1000.

The parameter values were chosen empirically taking into account a reasonable balance of obtained solution quality versus total time taken. The cGA was executed 20 times on each dataset. The execution time range from few seconds to about a minute depending on the dataset size. These times are within the times published by (Falco et al., 2007).

The source code and the datasets used are publicly available on the following Git repository: https://github.com/nunocsleite/cellularGA-clustering.

The cGA results were compared with results produced by nine classical classification techniques, and with Particle Swarm Optimization (PSO) (Falco et al., 2007), Artificial Bee Colony (ABC) (Karaboga and Ozturk, 2011), and Firefly Algorithm (FA) (Senthilnath et al., 2011). In order to perform the classification, the datasets were divided into 75% for training and 25% for testing. The samples were selected randomly, and divided in a balanced way, in order to guarantee that the percentage of samples in each class for the testing and training partitions was approximately the same. The same transformations to the tested databases as in (Falco et al., 2007) were carried out (transformation on attributes with missing data, data shuffling, and E.Coli database class removal).

5.2 Comparison with Other Classification Techniques

Table 2 compares the cGA against three nature based metaheuristics. Table 3 shows the results of the cGA on the tested datasets and comparison with the nine classification methods from literature as published in (Falco et al., 2007). The tables present the average percentages of incorrect classification on the testing set for each algorithm. For every method except for ABC, the number of runs reported is 20. For the ABC, the authors report five runs. From the registered results it can be observed that the cGA is competitive with the other techniques, attaining the best value on the Diabetes dataset.

To assess the effectiveness of the analysed techniques, a statistical ranking was performed (Table 4). The results of the statistical tests and analysis were produced using the Java tool of (García and Herrera, 2008). The cGA is positioned in the fifth position. Relating the metaheuristics, the cGA is superior to the PSO approach. Relating the other classification methods, only the Bayesian network (Bayes Net) and the MultiLayer Perceptron Artificial Neural Network (MLP ANN) are superior to the cGA.

Table 2: Average percentages of incorrect classification on the testing set. For each test database, the best value obtained among all the techniques is presented in bold.

	cGA	PSO	ABC	FA
Balance	12.02	13.12	15.38	14.10
Credit	13.34	18.77	13.37	12.79
Dermatology	4.95	6.08	5.43	5.43
Diabetes	**21.61**	21.77	22.39	21.88
E.Coli	14.32	13.90	13.41	**8.54**
Glass	37.36	38.67	41.50	37.74
Heart	18.60	15.73	14.47	**13.16**
Horse colic	33.91	35.16	38.26	32.97
Iris	1.62	5.26	**0.00**	**0.00**
Thyroid	3.49	3.88	3.77	**0.00**
WDBCancer	4.26	3.49	2.81	**1.06**
WDBCancer-Int	1.61	2.64	**0.00**	**0.00**
Wine	2.84	2.88	**0.00**	**0.00**

Figure 3 ilustrates the evolution of the train and classification error as a function of the number of iterations for the Balance dataset. Observing Figure 3 it can be concluded that for the Balance dataset the algorithm steadily improves the training and testing

Table 3: Average percentages of incorrect classification on the testing set. For each test database, the best value obtained among all the techniques is presented in bold.

	cGA	Bayes Net	MLP ANN	RBF	KStar	Bagging	Multi Boost	NBTree	Ridor	VFI
Balance	12.02	19.74	**9.29**	33.61	10.25	14.77	24.20	19.74	20.63	38.85
Credit	13.34	12.13	13.81	43.29	19.18	**10.68**	12.71	16.18	12.65	16.47
Dermatology	4.95	**1.08**	3.26	34.66	4.66	3.47	53.26	1.08	7.92	7.60
Diabetes	**21.61**	25.52	29.16	39.16	34.05	26.87	27.08	25.52	29.31	34.37
E.Coli	14.32	17.07	13.53	24.38	18.29	15.36	31.70	20.73	17.07	17.07
Glass	37.36	29.62	28.51	44.44	**17.58**	25.36	53.70	24.07	31.66	41.11
Heart	18.60	18.42	19.46	45.25	26.70	20.25	18.42	22.36	22.89	18.42
Horse colic	33.91	30.76	32.19	38.46	35.71	**30.32**	38.46	31.86	31.86	41.75
Iris	1.62	2.63	**0.00**	9.99	0.52	0.26	2.63	2.63	0.52	**0.00**
Thyroid	3.49	6.66	**1.85**	5.55	13.32	14.62	7.40	11.11	8.51	11.11
WDBCancer	4.26	4.19	2.93	20.27	2.44	4.47	5.59	7.69	6.36	7.34
WDBCancer-Int	1.61	3.42	5.25	8.17	4.57	3.93	5.14	5.71	5.48	5.71
Wine	2.84	**0.00**	1.33	2.88	3.99	2.66	17.77	2.22	5.10	5.77

Table 4: Average Rankings of the algorithms (higher ranking values are better). The Friedman test on these results has a p-value of $2.002 \times 10^{-8} \ll 0.05$ making these results statistically significant.

Algorithm	Average ranking value
FA	10.58
MLP ANN	9.19
ABC	9.12
Bayes Net	8.81
cGA	8.62
Bagging	8.23
PSO	7.50
KStar	6.35
NBTree	6.31
Ridor	5.62
MultiBoost	4.27
VFI	4.12
RBF	2.31

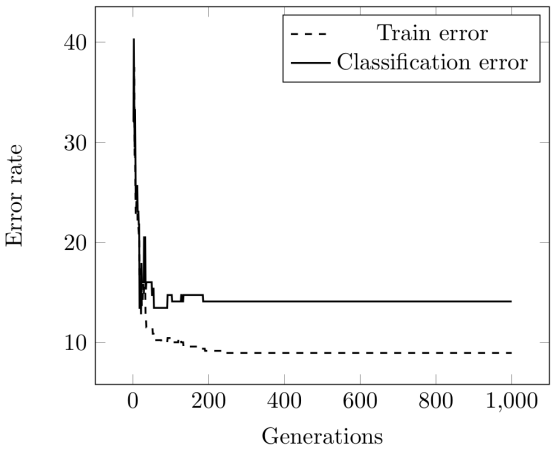

Figure 3: Evolution of the train and classification error as a function of the number of iterations for the Balance dataset.

error in the first 30 generations. Then, it suffers from a slight overfitting effect as the train error continues to decrease and the test error increases.

5.3 Train with Cross-validation

To assess if the results of using a simple split were biased or not, the classifier was trained using the cross-validation technique. The instances composing each dataset were split in the following manner: 75% of the instances for training the classifier, and 25% for testing the learned classifier. The training set was further divided into a training partition (84%), and a development partition (16%). The results for the 13 UCI datasets are presented in Table 5. The cross-validation results are slightly different from those with simple split but the deviation is not significant except for the Balance and Glass datasets.

5.4 Non-linearly Separable Dataset

In this section, an experiment with a non-linearly separable synthetic dataset was made. This dataset (called 'rings' (Lourenço et al., 2015)) has $N = 450$ samples classified into $C = 3$ classes (Figure 4). There are 200 samples of class 1, 200 samples of class 2, and the remaining 50 samples belong to class 3.

In order classify this dataset a variable number of clusters per class, instead of a single one, was used. Figures 5, 6, and 7, illustrate the use of one, three, and four clusters per class, respectively. As can be ob-

Table 5: Execution of cGA with simple split and using split into three partitions (train, validation, and test partitions) with cross-validation (CV) on the validation partition. 20 runs were executed.

Dataset	Simple split	Split with CV
Balance	**12.02**	15.77
Credit	13.34	**12.62**
Dermatology	**4.95**	5.60
Diabetes	21.61	**21.41**
E.Coli	14.32	**13.09**
Glass	37.36	**33.68**
Heart	**18.60**	21.93
Horse colic	33.91	**32.88**
Iris	1.62	**1.35**
Thyroid	**3.49**	3.58
WDBCancer	4.26	**3.56**
WDBCancer-Int	**1.61**	3.74
Wine	**2.84**	3.30

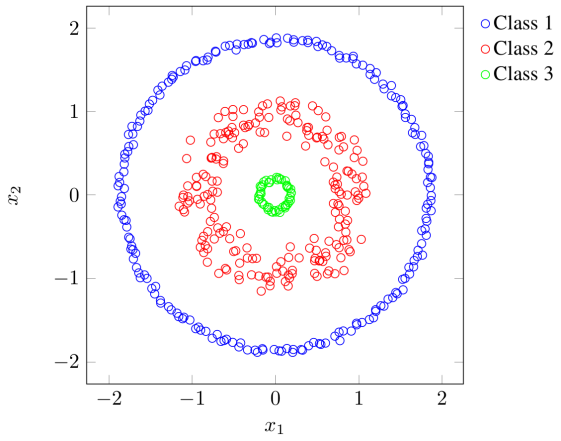

Figure 4: Distribution of samples into classes for the synthetic 'rings' dataset.

served, the classification error is significantly reduced with the addition of more clusters.

Although the addition of more clusters could solve this specific clustering problem, it couldn't improve the classification accuracy on the UCI datasets. This indicate that the UCI datasets probably belong to the third type mentioned in Section 4.6, where the examples of each class are mixed into confusion regions, that are difficult to tackle by a centroid based algorithm.

6 CONCLUSIONS

In this work, a Cellular Genetic Algorithm has been used to tackle the problem of classification of patterns into clusters. The algorithm was tested on 13

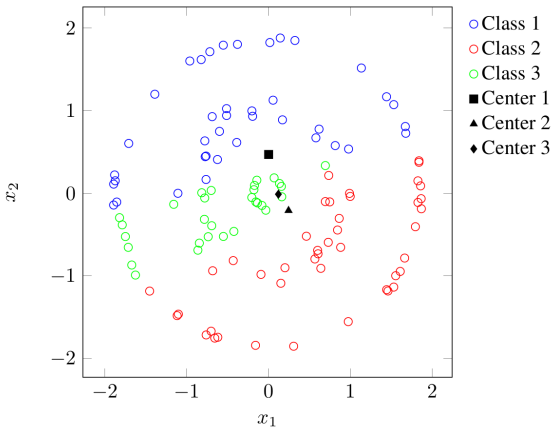

Figure 5: Classification of the 'rings' dataset using one cluster per class. The obtained classification error is equal to 54.4643% (61 misclassified examples out of 112).

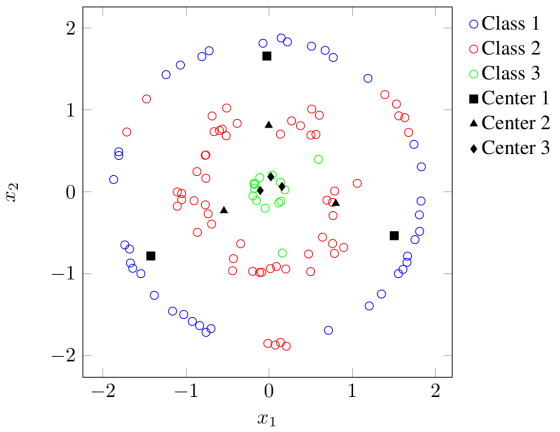

Figure 6: Classification of the 'rings' dataset using three clusters per class. The obtained classification error is equal to 11.6071% (13 misclassified examples out of 112).

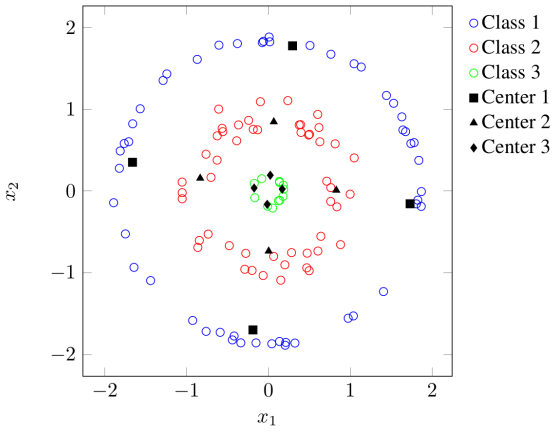

Figure 7: Classification of the 'rings' dataset using four clusters per class. The obtained classification error is equal to 0%.

diverse benchmarks and compared with nine classification techniques and three metaheuristics.

Experiments show that the cGA is competitive with the chosen special purpose classification techniques and also with the general purpose metaheuristics, attaining the best result on one problem instance. The cGA was ranked in fifth place.

Regarding future developments, two points were considered. The first is test other crossover and mutation operators, and verify if the hybridization of the cGA with local search metaheuristics is able to improve the basic cGA. The second point aims at performing a more rigorous study of the population grid layout, size, and neighbours used.

ACKNOWLEDGEMENTS

This work was partially supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) under project [UID/EEA/50009/2013], and by the PROTEC Program funds under the research grant [SFRH/PROTEC/67953/2010].

REFERENCES

Alba, E. and Dorronsoro, B. (2008). *Cellular Genetic Algorithms*. Springer Publishing Company, Incorporated, 1st edition.

Borgelt, C. (2006). *Prototype-based classification and clustering*. PhD thesis, Otto-von-Guericke-Universität Magdeburg, Universitätsbibliothek.

Cahon, S., Melab, N., and Talbi, E.-G. (2004). Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics*, 10(3):357–380.

Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification (2Nd Edition)*. Wiley-Interscience.

Falco, I. D., Cioppa, A. D., and Tarantino, E. (2007). Facing classification problems with particle swarm optimization. *Appl. Soft Comput.*, 7(3):652–658.

Falkenauer, E. (1998). *Genetic Algorithms and Grouping Problems*. John Wiley & Sons, Inc., New York, NY, USA.

García, S. and Herrera, F. (2008). An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694.

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651 – 666. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR)19th International Conference in Pattern Recognition (ICPR).

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323.

Karaboga, D. and Ozturk, C. (2011). A novel clustering approach: Artificial bee colony (abc) algorithm. *Appl. Soft Comput.*, 11(1):652–657.

Lichman, M. (2013). UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences.

Lourenço, A., Bulò, S. R., Rebagliati, N., Fred, A. L. N., Figueiredo, M. A. T., and Pelillo, M. (2015). Probabilistic consensus clustering using evidence accumulation. *Machine Learning*, 98(1-2):331–357.

Marinakis, Y., Marinaki, M., Doumpos, M., Matsatsinis, N., and Zopounidis, C. (2008). A hybrid stochastic geneticgrasp algorithm for clustering analysis. *Operational Research*, 8(1):33–46.

Maulik, U., Bandyopadhyay, S., and B, S. (2000). Genetic algorithm-based clustering technique. *Pattern Recognition*, 33:1455–1465.

Mirkin, B. (1996). *Mathematical Classification and Clustering*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Ni, J., Li, L., Qiao, F., and Wu, Q. (2013). A novel memetic algorithm and its application to data clustering. *Memetic Computing*, 5(1):65–78.

Niknam, T. and Amiri, B. (2010). An efficient hybrid approach based on pso, aco and k-means for cluster analysis. *Appl. Soft Comput.*, 10(1):183–197.

Selim, S. Z. and Ismail, M. A. (1984). K-means-type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6(1):81–87.

Senthilnath, J., Omkar, S., and Mani, V. (2011). Clustering using firefly algorithm: Performance study. *Swarm and Evolutionary Computation*, 1(3):164 – 171.

Zhao, J., Lei, X., Wu, Z., and Tan, Y. (2014). Clustering using improved cuckoo search algorithm. In Tan, Y., Shi, Y., and Coello, C., editors, *Advances in Swarm Intelligence*, volume 8794 of *Lecture Notes in Computer Science*, pages 479–488. Springer International Publishing.