A genetic algorithm for cluster analysis

Article *in* Intelligent Data Analysis · February 2003

DOI: 10.3233/IDA-2003-7103 · Source: DBLP

2 authors:

Eduardo R Hruschka
University of São Paulo
114 PUBLICATIONS **4,078** CITATIONS
SEE PROFILE

Nelson Ebecken
Federal University of Rio de Janeiro
346 PUBLICATIONS **2,451** CITATIONS
SEE PROFILE

Intelligent Data Analysis 7 (2003) 15–25
IOS Press

A genetic algorithm for cluster analysis

Eduardo R. Hruschka[a] and Nelson F.F. Ebecken[b]
COPPE / Federal University of Rio de Janeiro, Brasil
[a]*E-mail: erh@onda.com.br*
[b]*E-mail: nelson@ntt.ufrj.br*

Received 12 February 2002
Revised 7 May 2002
Accepted 27 May 2002

Abstract. This paper describes a new approach to find the right clustering of a dataset. We have developed a genetic algorithm to perform this task. A simple encoding scheme that yields to constant-length chromosomes is used. The objective function maximizes both the homogeneity within each cluster and the heterogeneity among clusters. Besides, the clustering genetic algorithm also finds the right number of clusters according to the Average Silhouette Width criterion. We have also developed specific genetic operators that are context-sensitive. Four examples are presented to illustrate the efficacy of the proposed method.

Keywords: Clustering, genetic algorithms, data mining

1. Introduction

Clustering is a task in which one seeks to identify a finite set of categories or clusters to describe the data. The literature of cluster analysis is huge and good references can be found in [1]. Clustering methods are useful for data reduction, for developing classification schemes and for suggesting or supporting hypotheses about the structure of the data. Thus, these methods have been used in a wide range of disciplines, such as psychiatry [2], market research [3], archaeology [4], pattern recognition [5], engineering [6], medicine [7] and market segmentation [8]. Arabie and Hubert [1] also mention three substantive areas where clustering techniques are applied: sociometry and social psychology, market structure analysis and evolutionary trees. This work describes a new clustering method, which is based on genetic algorithms and that can be useful for data mining tasks.

A generic description of the clustering objective is to maximize homogeneity within each cluster while maximizing heterogeneity among different clusters [1], in a way that objects that belong to the same cluster are more similar than objects that belong to different clusters. To cluster objects according to their similarities, a measure of how they resemble should be defined. Usually, however, this problem is tackled in an indirect way. In this sense, distance measures are used to quantify the dissimilarity between two objects in a way such that similar objects have small dissimilarity measures. In this sense, several dissimilarity measures are employed for clustering like, for example, the commonly used Euclidean distance.

Basically, there are three kinds of clustering techniques: overlapping, hierarchical and partitioning. This work deals with the partitioning approach, which assigns each object to exactly one cluster. Thus,

this work considers that clustering involves the partitioning of a set X of objects into a collection of mutually disjoint subsets C_i of X. Formally, let us consider a set of N objects $X = \{X_1, X_2, \ldots, X_N\}$ to be clustered, where each $X_i \in \mathcal{R}^\rho$ is an attribute vector consisting of "ρ" real measurements. The objects must be clustered into non-overlapping groups $C = \{C_1, C_2, \ldots, C_k\}$ where k is the number of clusters, such that:

$$C_1 \cup C_2 \cup \ldots \cup C_k = X, \quad C_i \neq \varnothing, \quad \text{and} \quad C_i \cap C_j = \varnothing \quad \text{for} \quad i \neq j. \tag{1}$$

Many methods described in the literature assume that the k value is given by the user [9–11]. Thus, these methods search for k clusters of similar objects according to a predefined criterion. Doing so, the number of ways of classifying N objects into k clusters is given by [12]:

$$NW(N,k) = \frac{1}{k!}\sum_{i=0}^{k}(-1)^i \binom{k}{i}(k-i)^N \tag{2}$$

For example, there are $NW(25,5) = 2,436,684,974,110,751$ different ways of classifying 25 objects into 5 clusters. Thus, it is not even easy to find the best clustering when the k value is known. On top of that, it is also very difficult to define the k value that represents the best clustering. A common approach is to run a clustering algorithm several times and, based on these results, to choose the best k value that provides the most natural clustering. This strategy assumes domain knowledge and usually has the disadvantage of searching for the best solution in a small subset of the solution space – these subsets are defined by the number of clusters, by the clustering criteria and by the clustering method [9]. Consequently, the probabilities of success of these methods are small. In data mining applications, the k value is hardly known and this fact leads us to the problem of optimizing this value according to a numeric criterion. However, if the best value of k is unknown, the number of ways of grouping N objects into k clusters is [12]:

$$\sum_{i=1}^{n} NW(N,k) \tag{3}$$

Considering 25 objects, this number represents approximately 4×10^{18} different clusters [9]. In a formal way, the problem with finding an optimal solution to the partition of N data into k classes is NP-complete [14] and because the number of distinct partitions of N objects into k clusters increases approximately as $k^N/k!$, attempting to find a globally optimum solution is usually not computationally feasible [1].

Genetic algorithms are widely believed to be effective on NP-complete global optimization problems and they can provide good sub-optimal solutions in reasonable time [14]. Thus, we believe that a clustering genetic algorithm can provide a way of finding the right clustering. Some recent approaches that describe the application of genetic algorithms to clustering problems can be found in [9,13–18]. Other methods can be found in [10,14,19–29]. The next section shows the problems caused by the application of classic genetic algorithms in clustering problems and also describes the proposed method. In the third section we describe simulation results on four different data sets and in the fourth section we present the conclusions.

2. Clustering genetic algorithm

There are several complications concerning the application of traditional genetic algorithms for clustering problems. The use of the simple encoding scheme that yields to constant-length chromosomes usually causes the problem of redundant codification and context insensitivity [13]. We have developed a genetic algorithm specifically designed for clustering problems. This algorithm allows the utilization of simple encoding scheme and avoids all of their problems.

2.1. Individual representation

Let us consider N objects to be clustered. Thus, the phenotypes, which are the possible solutions to the clustering problem, are represented as one dimensional integer arrays with $(N + 1)$ elements. These arrays are called genotypes, which are the representations of the possible solutions. As each data unit can be numbered from 1 to N, the ith element (ith gene) of a genotype represents the ith data unit, whereas the last gene represents the number of clusters. Therefore, each gene of an individual has a value over the alphabet $\{1, 2, \ldots, k\}$, where k is the maximum number of clusters. For example, considering a dataset with 20 objects one can get the following clustering:

Genotype1: $2234512345332145 4552\ 5$

This means that 5 objects {1,2,7,13,20} form the cluster whose label is two, and the cluster whose label is 1 has 2 objects {6,14}. In other words, the 6th and the 14th objects of the dataset form the cluster whose label is one. Besides, the last gene corresponds to the number of clusters encoded by the solution, which in this case is five. The first problem with the straightforward encoding scheme above is that it is highly redundant (there are 5! chromosomes encoding the same solution to the original problem), i.e. the encoding is one-to-many. Considering the genotype 1, one can find more chromosomes that also correspond to the same phenotypes like, for example:

Genotype2: $3345123451443251 5113\ 5$

Genotype3: $4451234512554312 1224\ 5$

Genotype4: $5512345123115423 2335\ 5$

Genotype5: $1123451234221534 3441\ 5$

Therefore, the space size that the genetic algorithm has to search is much larger than the original space of solutions. In fact, this artificial enlargement of the original search space can lead to a severe loss of power of the genetic algorithm [13]. Indeed, the objective function should depend exclusively on the grouping of the objects, rather than on the group label, that is, what matters is not the cluster label but the objects it contains. Besides, the problem of redundant codification depends not only on the applied codification scheme but also on the employed operators. Thus, we developed operators that are specific for clustering problems. These operators were designed to avoid the problem of redundant codification as well as the problem of context insensitivity [13], which is the undesirable effect of casting context-dependent information out of context under the standard crossover. For example, if genotype 2 was crossed over with genotype 3, the resulting genotypes should represent phenotypes equal to their parents, because these represent the same solution to the clustering problem. But this rarely happens. In

1) Initialize a population of random genotypes;
2) Evaluate each genotype in the population;
3) Apply a linear normalization;
4) Select genotypes by proportional selection;
5) Apply the crossover and mutation operators;
6) Replace the old genotypes by the ones formed by 5);
7) If the convergence is attained, stop; if not, go to step 2).

Fig. 1. Clustering Genetic Algorithm (CGA).

order to make it more easily observed, let us apply the two-point crossover to the following genotypes, which represent the same phenotypes:

Genotype6: $2|2222|111113333344444\ 4$

Genotype7: $4|4444|333335555511111\ 4$

Applying the two-point crossover to the genotypes 6 and 7 as above illustrated will result in the following genotypes:

Child1: $2\ 4444\ 111113333344444\ 4$

Child2: $4\ 2222\ 333335555511111\ 5$

One can observe that these children are different from their parents, that is, they represent different phenotypes. On top of that, it is verified that the second child represents five clusters, whereas their parents represent just four. Considering clustering methods, this problem happens when one is dealing with the classic genetic operators of recombination and mutation. These problems can be solved by the Consistent Algorithm [18], which is a renumbering algorithm. Applying this algorithm, the genotypes $\{1,2,3,4,5\}$ will result in the same one: $11234512342215343441\ 5$.

The Consistent Algorithm allows the suitable application of classic genetic algorithms in clustering problems, avoiding the problems of redundant codification and context insensibility, as can be seen in [15–18]. Although the obtained results were reasonable, we believe that it could be possible to improve the clustering genetic algorithm performance by using genetic operators specific for clustering problems. Besides, another additional problem is that traditional genetic algorithms, when applied in clustering tasks, often perform just a random search for the best solution.

Considering all these aspects, we propose a genetic algorithm designed specifically for clustering problems, which we called the Clustering Genetic Algorithm (CGA). Basically, we have employed the same codification scheme as previously explained, but we have developed new genetic operators that are group-oriented, i.e. context-sensitive and this is the main reason for applying the operators we have developed. The main steps employed by the CGA are described in Fig. 1.

2.2. Objective function

Determining the optimal number of clusters in a dataset is one of the most difficult aspects in the clustering process [9]. Most of the approaches found in the literature determine the right number of clusters according to criteria based on given clusterings. However, the CGA is suppose to optimize not only the clusterings for a given number of clusters but also the number of clusters, i.e. the objective function must maximize the homogeneity within each cluster, the heterogeneity among clusters, and also must be the highest one when the solution represents the right number of clusters.

The homogeneity within each cluster can be obtained by minimizing the distances among its objects. For example, the k-means algorithm tries to minimize the sum of square Euclidean distances between objects and their cluster centroids [27]. In this sense, the heterogeneity among different clusters can be measured by computing the distances among their centroids, and these distances should be maximized.

The CGA employs an objective function based on the Average Silhouette Width developed by Kaufman and Rousseeuw [10]. In principle, this concept was developed in order to choose the right number of clusters considering that there are different clusterings to be evaluated. However, we believe that this approach can be used as the objective function for the CGA. As genetic algorithms work in parallel with a certain number of solutions, it is possible to work with genotypes representing different clusterings both in terms of the number of clusters and of the clustering structure itself. Basically, let us consider an object "i" belonging to cluster A. So the average dissimilarity of "i" to all other objects of A is denoted by $a(i)$, i.e.:

$$a(i) = \frac{\sum_{j=1}^{L} d_{ij}}{(L-1)} \tag{4}$$

where d_{ij} is the dissimilarity between object "i" and object "j" and "L" is the number of objects that belong to cluster A. Thus, the more similar the objects of A are, the smaller the $a(i)$ value is. Now consider a different cluster C and let us calculate the average dissimilarity of "i" to all objects of C, which will be here denoted by $d(i, C)$, i.e.:

$$d(i, C) = \frac{\sum_{j=1}^{M} d_{ij}}{M} \tag{5}$$

where "M" is the number of objects of cluster C. After computing the $d(i, C)$ for all clusters $C \neq A$ we select the smallest of those, which will be here called $b(i)$, i.e.:

$$b(i) = \min\ d(i, C),\ C \neq A. \tag{6}$$

This number represents the average dissimilarity of "i" to its neighbor cluster. Thus, the higher the $b(i)$ value, the higher is the heterogeneity among object "i" and the objects of the neighbor cluster. Now one defines the silhouette $s(i)$ like follows:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{7}$$

When cluster A contains only one object we consider that $s(i) = 0$, which is the most neutral choice [10]. In addition, it is easy to see that: $-1 \leqslant s(i) \leqslant 1$. Thus, the employed objective function is the average of $s(i)$ for all the objects $i = 1, 2, \ldots, N$, i.e.:

$$\overline{s} = \sum_{i=1}^{N} s(i) \tag{8}$$

Thus, the $\overline{s}$ value will be high when the objects have small $a(i)$ values and high $b(i)$ values, what implies in high internal homogeneities and high external heterogeneities among the obtained clusters. It

implies that the best value of "k" happens when the objective function value is as high as possible. In order to provide a mutual comparison among the chromosomes, a linear normalization [30,31] is applied to the objective function values.

2.3. Selection

The individuals that make part of each generation are selected according to the roulette wheel selection strategy. As this strategy does not allow negative objective function values, a constant number equal to one is summed up to each objective function value before the selection process takes place. Doing so, one observes that the average silhouette range changes to $0 \leqslant s(i) \leqslant 2$. After this process, the objective function values are summed up with a constant equals to -1, which makes them return to the original range $[-1, +1]$. Besides, an elitist strategy is used, that is, the best individual of each generation is always copied into the succeeding generation.

2.4. Operators

Classic genetic operators are not suitable for clustering problems [13] because they usually work with a set of unrelated genes. In clustering problems the genes are related, i.e. equal allele genes mean that the objects they represent belong to the same cluster. In other words, the groups are the meaningful building blocks of a solution. Thus, it is necessary to use operators that work with groups of genes, instead of with the genes themselves. These operators are referred to as group oriented [13].

The CGA was developed as an attempt to improve the method developed by Falkenauer [13], called the Grouping Genetic Algorithm (GGA). This algorithm is a paradigm, that is, a recipe to be adapted to a given grouping problem. The paradigm supposes the use of genotypes of variable length, where the first part of the genotypes is equal to the genotypes employed by the CGA and there is an additional part, formed by the cluster labels. We have adapted the paradigm developed in [13] to the partitioning problem. Besides, our method simplifies that paradigm. Basically, the crossover and mutation operators employed by the CGA perform the same task, in conceptual terms, as those described in [13]. The main advantage of our method is that the developed operators work with genotypes of fixed length, notably simpler. Besides, Falkenauer [3] proposes the use of an inversion operator. This operator intends to change the positions of some genes on the chromosome in order to assist the crossover in transmitting good schemata to the offspring. However, the inversion operator success basically depends on the randomness involved in the process, what makes us disregard its application.

2.4.1. Crossover

The crossover operator combines pieces of information coming from different genotypes and it works in the following way: First, two genotypes (A and B) are selected; Second, considering that A represents $k1$ clusters, the CGA chooses randomly $n(1 \leqslant n \leqslant k1)$ clusters to copy in B. The unchanged groups of B are maintained and the changed ones have all their objects allocated to the cluster that has the nearest centroid. In this way the child C is obtained. This same process is employed to get child D, but now considering that the changed groups of B are copied in A. In order to illustrate this procedure, let us consider the following two genotypes where the last gene, which corresponds to the number of clusters, is not represented:

$A - 1123245125432533424$

$B - 1212332124423221321$

For example, we can select randomly the groups 2 and 3 (which are in bold type) of A:

$A - 1123245125432533424$

$B - 1212332124423221321$

These groups modify the groups 1, 2 and 3 in B, whereas the group 4 is not modified:

$A - 1123245125432533424$

$B - 1212332124423221321$

The underlined positions, which correspond to the genes not directly affected by clusters °2– and °3– of A, are now changed to °zero–. Thus the following genotype is found:

$C - 0023200020032033020$

The genes completed with zeros indicate positions with potential to receive either unchanged groups of B or they are objects that will be placed to the nearest cluster of C. For example, the group 4 is maintained because it was not changed. Therefore, the following genotype is found:

$C - 0023200024432033020$

The other objects (whose alleles are zeros) are placed to the nearest cluster $\{2,3,4\}$. Considering the child D, we observe that it will be initially formed by the groups $\{1,2,3\}$ of B and by those unchanged groups of A, that is, the changed groups of B (when child C is formed) are copied in D:

$D - 1212332120023221321$

Doing so, one observes that all the groups in A are changed, then the remaining objects are placed to the nearest cluster. Besides, one has to mention that the developed crossover operator provides children formed by a number of clusters that belongs to the range $[k1, k2]$, where $k1$ is the number of clusters represented by one parent and $k2$ is the number of clusters represented by the other parent.

2.4.2. Mutation

There are two operators for mutation. The operator 1 (that only works in clusterings formed by more than two groups) eliminates a randomly chosen group and places all their objects to the remaining cluster that has the nearest centroid. The operator 2 divides a randomly selected group into two new ones. The objects closer to the centroid form the first group, whereas those objects closer to the farthest object to the centroid form the other group. These operators are suitable for grouping problems because they just change the genotypes in the smallest possible way, i.e. dividing groups and eliminating others in the hope of finding better solutions to the problem being solved.

2.4.3. Parameters

The CGA does not employ crossover and mutation probabilities, that is, after the roulette wheel selection process, the designed operators are applied in some selected genotypes. This strategy considers that 50% of the selected genotypes are crossed-over, 25% are mutated by operator 1 and 25% are mutated by operator 2. Let us consider that the CGA employs a population formed by G genotypes. Thus, the crossover is applied in the following way: the first selected genotype is recombined with the $(G/2)$th selected genotype, the second selected genotype is recombined with the $(G/2 - 1)$th selected genotype and so on. The first mutation operator is applied from the $(G/2 + 1)$th genotype until the $(3G/4)$th genotype and the second mutation operator is applied to the remaining ones, which belong to the range $[3G/4 + 1, G]$.

Table 1
Number of generations until the highest objective function value was first met

Simulation	1	2	3	4	5	6	7	8	9	10	Average	Standard deviation
Ruspini	13	24	61	28	73	83	45	54	33	30	44	22.8
Randomly generated	14	11	12	16	14	13	4	12	8	16	12	3.7
Breast cancer	30	27	17	25	26	21	33	25	13	39	26	7.5
Iris plant	38	16	8	27	19	17	22	27	32	13	22	9.1

2.5. Initial population

Random initial populations were used on the experiments here described. Basically, one considers a population formed by 20 different genotypes. The first genotype represents two clusters, the second genotype represents three clusters, the third genotype represents four clusters, ..., and the last one represents 21 clusters, i.e. the maximum number of clusters was set equals to 21. In this way, each gene of a genotype has an equal chance to have any possible value belonging to the interval $[1, k1]$ where $k1$ is the number of clusters represented by the genotype.

3. Simulations

The efficacy of the proposed method is illustrated by presenting the results obtained on four different data sets: Ruspini Data, Randomly Generated Data, Breast Cancer Data and Iris Plant Data. In each described simulation, we considered a population formed by 20 genotypes that, in turn, implies in using 21 clusters at most. Besides, we set the maximum number of generations to 200. The Euclidean distance was used as the metric to calculate the dissimilarities between objects. We simulated 10 experiments on each dataset and Table 1 shows the obtained results in terms of the number of generations until the best objective function value was first found.

3.1. Example 1 – Ruspini data

The first simulation deals with the well-known Ruspini data [10]. There are 75 objects described by means of two attributes $\{x, y\}$. Four groups form the right clustering, as it can be seen in Fig. 1. The CGA found the right clustering on average after 44 generations (see Table 1), with an objective function value equals to 0.74, which is also the correct objective function value (8) to this problem. This is a very simple problem, but it shows that the CGA quickly found the right solution from a randomly generated initial solution.

3.2. Example 2 – 200 randomly generated objects

The second simulation deals with 200 objects described by two variables randomly generated [10]. Three groups of objects are constructed and in each group the points are generated according to a spherical bivariate normal distribution with given mean vector $\mu = (\mu_x, \mu_y)$ and standard deviation σ for both x and y according to:

Group1: 120 objects $\mu_x = 0$ $\mu_y = 10$ $\sigma = 1.7$

Group2: 60 objects $\mu_x = 20$ $\mu_y = 12$ $\sigma = 0.7$

Group3: 20 objects $\mu_x = 10$ $\mu_y = 20$ $\sigma = 1.0$

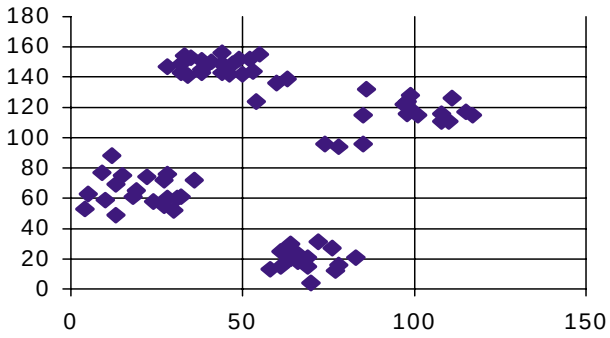

Fig. 2. Ruspini data.

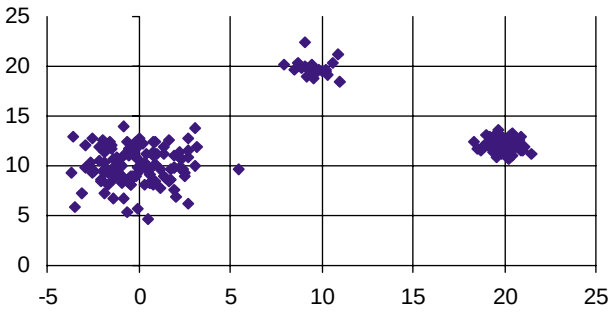

Fig. 3. Randomly generated data.

The three groups depicted in Fig. 3 form the right clustering. The CGA found the right clustering on average after 12 generations, with an objective function value (8) equals to 0.82, which also corresponds to the right solution to this problem.

3.3. Example 3 – Wisconsin breast cancer data

This database is available at the UCI Machine Learning Repository [32]. Each object has 9 attributes and an associated class label (benign or malignant). The two classes are known to be linearly inseparable. The total number of objects is 699 (458 benign and 241 malignant), of which 16 have a single missing feature. We removed those 16 objects and used the remaining ones, without the class labels, as inputs to the clustering genetic algorithm. The CGA found two clusters in all simulations (in the average after 26 generations), but the clusterings were not always the same ones. Thus, the average classification rate was not the same one in each experiment, i.e. it varied from 95.02% (fitness function value $=$ 0.59) to 95.75% (fitness function value $=$ 0.60), which is slightly better than the results described in [19]. Besides, one observes that the right clustering to this problem provides a fitness function value which is equal to 0.57 and which is smaller than those obtained by the CGA. It happens because some objects have negative $s(i)$ values when these are calculated according to the groups formed by means of the class labels. In other words, some objects are more similar to the ones of the opposite class, what prevents the CGA from finding the right solution.

3.4. Example 4 – Iris plant database

This database consists of 3 classes (Setosa, Versicolour and Virginica), each one formed by 50 examples of plants. There are 4 attributes (sepal and petal length and width). The class Setosa is linearly separable from the others, whereas the classes Versicolour and Virginica are not linearly separable from each other. The CGA found only two clusters after 22 generations with an objective function value equals to 0.68, which indicates that a reasonable structure would have been found [10]. However, the CGA did not find the right clustering. Actually, the CGA has just separated the objects of the first class – Setosa – from the others. The algorithm proposed by Kothari and Pitts [19] also found two clusters to this problem. Besides, Cole [9] reported that a Grouping Genetic Algorithm as well as the Ward's Method have also found two clusters. In fact, the right solution provides an objective function value which is equal to 0.50 and which is smaller than the value obtained by the CGA (0.68). It happens because many objects that belong to the classes virginica and versicolour are very similar.

4. Conclusions and future work

Determining the optimal clustering is a very difficult task. Most optimization clustering methods require the user to specify the number of clusters, and hierarchical methods typically produce a series of 1 to N clusters without specifying the most appropriate number of clusters. This paper describes a clustering genetic algorithm that allows both to find the best number of clusters and also to get the best clustering according to the average silhouette width criterion.

The efficacy of the proposed method is illustrated by presenting the simulation results on four different datasets: Ruspini, Randomly Generated, Breast Cancer and Iris Plants. The CGA found the right clustering both in the Ruspini and in the Randomly Generated Datasets. Besides, the CGA provided an average classification rate equals to 95.42% in the Wisconsin Breast Cancer and this result is comparable to the best ones described in the literature. Considering the Iris Plants Database, the CGA did not find the right clustering, but the obtained solution provides a higher objective function value than the right one.

Therefore, the results obtained by the application of the CGA in four examples show that the method is very promising, mainly considering that we have employed small initial populations randomly generated. Besides, we believe that it is possible to get better results by: using different dissimilarity measures, applying other genetic operators, defining a heuristic to set up the initial population and by studying the best way of applying the operators. We also think that the CGA can be useful in a variety of data mining tasks and we hope to describe useful applications soon.

References

[1] P. Arabie and L.J. Hubert, An Overview of Combinatorial Data Analysis (Chapter 1), in: *Clustering and Classification*, P. Arabie, L.J. Hubert and G. DeSoete, eds, World Scientific, 1999.

[2] S. Levine, I. Pilowski and D.M. Boulton, The classification of depression by numerical taxonomy, *The British Journal of Psychiatry* **115** (1969), 937–945.

[3] P.E. Green, R.E. Frank and P.J. Robinson, Cluster analysis in test market selection, *Management Science* **B13**(8), 387–400.

[4] F.R. Hodson, Numerical Typology and prehistoric archaeology, in: *Mathematics in the Archaeological and Historical Sciences*, F.R. Hodson, D.G. Kendall and P.A. Tautu, eds, University Press, Edimburg, 1971.

[5] E. Levrat, V. Bombardier, M. Lamotte and J. Bremont, Multi-level image segmentation using fuzzy clustering and local membership variations detection, *IEEE International Conference on Fuzzy Systems*, IEEE Press, NY, 1992, pp. 221–228.

[6] Y. Reich and S. Fenves, The formation and use of abstract concepts in design, in: *Concept Formation: Knowledge and Experience in Unsupervised Learning*, D. Fisher, M. Pazzani and P. Langley, eds, Morgan Kaufman, San Mateo, 1991, pp. 323–354.
[7] M. Funk, R.D. Appel, D. Roch and D. Hochstrasser et al., Knowledge acquisition in expert system assisted diagnosis: a machine learning approach, *Proceedings of the AIME-87*, Springer Verlag, 1987, pp. 99–103.
[8] J.P. Bigus, *Data Mining with Neural Networks*, (First Edition), McGraw-Hill, USA, 1996.
[9] R.M. Cole, Clustering with Genetic Algorithms, Master of Science Thesis, Department of Computer Science, University of Western Australia, 1998.
[10] L. Kaufman and R.J. Rousseeuw, Finding Groups in Data – An Introduction to Cluster Analysis, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons Inc., 1990.
[11] G.W. Miligan, Clustering Validation: Results and Implications for Applied Analyses, in: *Clustering and Classification*, P. Araboe, L.J. Hubert and G. de Soete, eds, World Scientific, Publishing Co. Pte. Ltd., Singapore, 1996, pp. 341–373.
[12] G.L. Liu, *Introduction to combinatorial mathematics*, McGraw Hill, 1968.
[13] E. Falkenauer, *Genetic Algorithms and Grouping Problems*, John Wiley & Sons, 1998.
[14] Y. Park and M. Song, A Genetic Algorithm for Clustering Problems, *Proceedings of the Genetic Programming Conference*, University of Wisconsin, July 1998.
[15] E.R. Hruschka and N.F.F. Ebecken, Credit approval by a clustering genetic algorithm, in: *Data Mining II*, N. Ebecken and C.A. Brebbia, eds, Proceedings of the Data Mining 2000 Conference, Cambridge University, Cambridge, UK, July 2000, pp. 403–413.
[16] E.R. Hruschka and N.F.F. Ebecken, Applying a Clustering Genetic Algorithm for Extracting Rules from a Supervised Neural Network, *Proceedings of the IEEE International Joint Conference on Neural Networks* (*IJCNN'2000*), Como, Italy, July 2000.
[17] E.R. Hruschka and N.F.F. Ebecken, Using a Clustering Genetic Algorithm for Rule Extraction from Artificial Neural Networks, *Proceedings of The First IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks*, San Antonio, TX, USA, May 2000.
[18] E.R. Hruschka and N.F.F. Ebecken, A Clustering Genetic Algorithm for Extracting Rules from Supervised Neural Network Models in Data Mining Tasks, in: *International Journal of Computers, Systems and Signals, Special Issue: Knowledge Discovery from Structured and Unstructured Data*, (Vol. 1), (No. 1), A.P. Engelbrecht, ed., December 2000, pp. 17–29.
[19] R. Kothari and D. Pitts, On finding the number of clusters, *Pattern Recognition Letters* (No. 20), Elsevier, 1999, pp. 405–416.
[20] E. Nakamura and N. Kehtarnavaz, Determining number of clusters and prototype locations via multi-scale clustering, *Pattern Recognition Letters*, (No. 19), Elsevier, 1998, pp. 1265–1283.
[21] H. Bock, Probability Models and Hypotheses Testing in Partitioning Cluster Analysis, in: *Clustering and Classification*, P. Arabie, L.J. Hubert and G. de Soete, eds, World Scientific, Publishing Co. Pte. Ltd., Singapore, 1996, pp. 377–454.
[22] R. Maitra, Clustering Massive Datasets, Department of Statistics, Carnegie Mellon University, Pittsburgh, PA, www.math.umbc.edu/~maitra/publications.html, in preparation, 1998.
[23] D. Boley and V. Borst, Unsupervised Clustering: A Fast Scalable Method for Large Datasets, Technical Report IIS-9811229, Department of Computer Science and Engineering, University of Minnesota.
[24] C. Fraley and A.E. Raftery, How Many Clusters? Which Clustering Method? Answers via Model-Based Cluster Analysis, Technical Report No. 329, Department of Statistics, University of Washington, Seattle, USA, Feb. 1998.
[25] T. Calinski and J. Harabasz, A dendrite method for cluster analysis, *Communications in statistics* **3**(1) (1974), 1–27.
[26] D.L. Davies and D.W. Bouldin, A cluster separation measure, *IEEE Transactions on pattern analysis and machine intelligence* **PAMI-1**(**2**) (1979), 224–227.
[27] B.S. Everitt, *Cluster Analysis*, (third edition), Halsted Press, 1993.
[28] G.W. Milligan and M.C. Cooper, An examination of procedures for determining the number of clusters in a data set, *Psychometrika* **50**(2) (1985), 159–179.
[29] M.J. Symons, Clustering criteria and multivariate normal mixtures, *Biometrics* **37** (1995), 655–659.
[30] L. Davis, *Handbook of Genetic Algorithms*, International Thompson Computer Press, 1996.
[31] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley Longman, Inc., 1989.
[32] C.J. Merz and P.M. Murphy, UCI Repository of Machine Learning Databases, [http://www.ics.uci.edu]. Irvine, CA, University of California, Department of Information and Computer Science.