A Genetic Rule-Based Data Clustering Toolkit

I Sarafis, AMS Zalzala and P W Trinder
Department of Computing and Electrical Engineering, Heriot-Watt University, Edinburgh, EH14 4AS, UK
i.sarafis@hw.ac.uk, a.zalzala@hw.ac.uk, trinder@cee.hw.ac.uk

Abstract- Clustering is a hard combinatorial problem and is defined as the unsupervised classification of patterns. The formation of clusters is based on the principle of maximizing the similarity between objects of the same cluster while simultaneously minimizing the similarity between objects belonging to distinct clusters. This paper presents a tool for database clustering using a rule-based genetic algorithm (RBCGA). RBCGA evolves individuals consisting of a fixed set of clustering rules, where each rule includes *d* non-binary intervals, one for each feature. The investigations attempt to alleviate certain drawbacks related to the classical minimization of square-error criterion by suggesting a flexible fitness function which takes into consideration, cluster asymmetry, density, coverage and homogeny.

I. INTRODUCTION

The tremendous volume and diversity of real-world data embedded in huge databases clearly overwhelm traditional manual methods of data analysis, such as spreadsheets and ad-hoc queries. An urgent need exists for a new generation of techniques and tools with the ability to intelligently and automatically assists users in analyzing mountains of stored data for nuggets of useful knowledge. These techniques and tools are the subject of the field of Knowledge Discovery on Databases (KDD), which is considered to be the "extraction of interesting (non-trivial, implicit, previously unknown and potentially useful) information or patterns from data in large databases [1]. Data Mining (DM) is a step in the process of KDD consisting of applying algorithms that, under acceptable computational efficiency limitations, produce a particular enumeration of patterns over the data [2].

Clustering is a common data mining task and refers to the application of algorithms for discovering interesting data distributions in the underlying data space. Given a large dataset consisting of multi-dimensional data points or patterns, the data space is usually not uniformly occupied. The aim of clustering procedures is to partition a heterogeneous multi-dimensional data set into groups of more homogenous characteristics [3]. The formation of clusters is based on the principle of maximizing similarity between patterns of the same cluster and simultaneously minimizing the similarity between patterns belonging to distinct clusters. Similarity or proximity is usually defined as a distance function on pairs of patterns and based on the values of the features of these patterns [4].

II. RELATED WORK

There are four basic types of clustering algorithms: *partitioning algorithms*, *hierarchical algorithms, density-based algorithms* and *grid-based algorithms.* Partitioning algorithms construct a partition of N objects into a set of *k* clusters [5]. Hierarchical algorithms create a hierarchical decomposition of the database that can be presented as *dendrogram* [13]. Density-based algorithms search for regions in the data space that are denser than a threshold and form clusters from these dense regions [14]. Grid-based algorithms quantize the search space into a finite number of cells and then operate on the quantized space [15]. Genetic algorithms (GA) have been proposed for clustering, because they avoid local optima and are insensitive to the initialization [7, 16, 17]. The individuals encode a fixed number (k) of clusters, using a binary representation for the center of clusters. The minimization of the squared error discussed in section III, is the fitness function used to guide the search.

III. DRAWBACKS OF THE K-MEANS ALGORITHM

The classical *k-means* clustering algorithm – and its variation the *k-medoids* – are representatives of partitioning techniques and have widely been used in clustering applications [1]. The reason behind the popularity of the k-means algorithm has to do with the simplicity and the speed of the algorithm. Broadly speaking, given the number of desired clusters, k-means algorithm attempts to determine k partitions that optimize a criterion function. The square-error criterion *E* is the most commonly used and is defined as the sum of the squared Euclidean distances between each multidimensional pattern *p* belonging to C_i cluster and the center m_i of this cluster.

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} \left\| p - m_i \right\|^2 \quad (1)$$

In k-means algorithm, each cluster is represented by a vector corresponding to the center of gravity for this cluster (section IV). In k-medoids, each cluster is described by one of the patterns, which are closely located to the center of gravity of this cluster. Both k-means and k-medoids assign pattern to clusters trying to minimize the square-error function in order to obtain k partitions that are as compact and separated as possible. However, there are some well-known drawbacks, such as sensitivity to the initialization process, which can lead to local optima, sensitivity to the presence of noise,

discovery of clusters with similar sizes and densities, and discovery of hyperspherical clusters [3]. The k-means method works well for clusters that are compact clouds (i.e. hypersherical in shape) and are well separated from each other. However, when there are large differences in the sizes or geometries or densities of different clusters the square error method could split large clusters to minimize equation (1) [11].

IV. RULE-BASED GENETIC ALGORITHM

In this paper, we suggest a non-binary, rule-based representation for the individuals and a flexible evaluation function, which can be used to alleviate the certain drawbacks of the k-means methods.

A. Individual Encoding

Let $A=\{A_1, A_2, \ldots, A_d\}$ be a set of domains and $S = A_1 \times A_2, \times \ldots \times A_d$ a d-dimensional numerical space. The input consists of a set of d-dimensional patterns $V=\{p_1, p_2, \ldots, p_k\}$, where each p_i is a vector containing d numerical values, $p_i = [\alpha_1, \alpha_2, \ldots, \alpha_d]$. The jth component (p_{ij}) of vector p_i is drawn from domain A_j.

Each individual consists of a set of k clustering rules. The number of rules is fixed and corresponds to the number of desired clusters. Each rule is constituted from d genes, where each gene corresponds to an interval involving one feature. Each ith gene, $i=1,..,d,$ of the of a rule is subdivided into two field: *lower boundary* (lb_i) and *upper boundary* (ub_i), where *lb* and *ub* denotes the lower and upper value of the ith feature in this rule. The conditional part of a rule is formed by the conjunction (logical *AND* operator) of d intervals corresponding to this rule. It should be pointed out that our approach use real-coded chromosome representation.

For example, consider a string corresponding to the clustering problem shown in Fig. 1. The two-dimensional feature space shows $k=2$ clusters and two features, namely *salary* and *tax* with domains $A_{salary}=\{0,1000\}$ and $A_{tax}=\{0,400\}$, respectively.

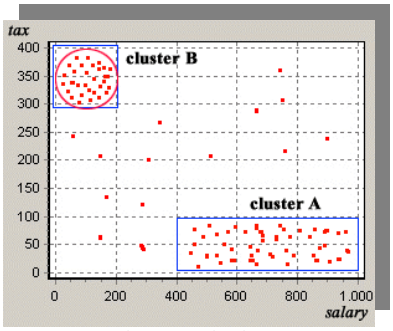


Fig. 1. Distribution of patterns for features salary and tax.

Rule A: [[400 ≤salary ≤1000] *AND* [0 ≤ tax ≤100]]
Rule B: [[0 ≤salary ≤200] *AND* [300 ≤ tax ≤400]]

The entire chromosome, which corresponds to a complete solution to the above clustering problem, is illustrated in Fig. 2.

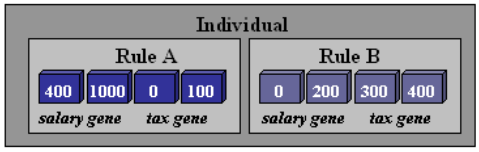


Fig. 2. The structure of the individuals.

B. Fitness Function

In order for a GA-based system to effectively search for the optimal solutions, an appropriate fitness function must be carefully implemented. Generally speaking the choice of the fitness function is closely related to the problem at hand. In our case, we focus on optimizing a set of clustering criteria that when they are simultaneously combined can ensure a) *high interclass dissimilarity* and b) *high intraclass similarity*.

Interclass dissimilarity: The distinctness of two rule-described clusters is defined in terms of the differences in their descriptions. Obviously, more distinct descriptions for the clusters produce better problem space decompositions. It is essential to avoid the generation of overlapping cluster descriptions.

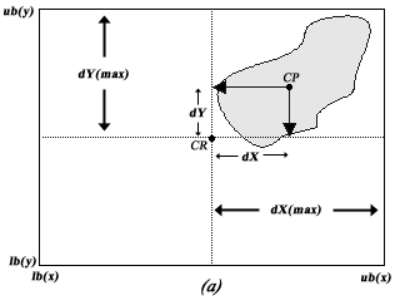
Intraclass similarity: This refers to the degree of cohesion of patterns within a specific cluster. The more coherent a cluster is, the more uniformly distributed (in the d-dimensional subspace defined by the cluster description) the patterns are.

To evaluate individual's fitness we consider five different concepts, namely, *rule asymmetry*, *rule density*, *rule coverage*, *rule homogeny* and *degree of rule overlapping*. Each one of the above criteria plays an important role in maximizing interclass dissimilarity and intraclass similarity.

1) Rule asymmetry

Rule asymmetry is a key factor that is used to ensure uniform distribution of patterns regarding to the patterns's center of gravity. Consider the distribution of a set $S=\{p_1, p_2, \ldots, p_k\}$ of d-dimensional patterns with a center of patterns $CP=\{cp_1, cp_2, \ldots, cp_d\}$, where cp_i denotes the mean value of patterns in ith dimension

$$cp_i = \frac{1}{k}\sum_{j=1}^{k} p_{ij}$$ and



a center of rule $CR=\{cr_1, cr_2, \ldots, cr_d\}$, where cr_i denotes the mean value of the interval corresponding to the i-th dimension $cr_i = lb_i + (ub_i - lb_i)/2$. The maximum distance $dpr_{i(max)}$ between cr_i and cp_i ($i=1,\ldots, d$) is $drp_{i(\max)} = (ub_i - lb_i)/2$, and for each dimension the coefficient of asymmetry a_i is given by equation 2.

$$a_i = \frac{dpr_i}{dpr_{i(\max)}} \quad (2)$$

Hence, the coefficient of asymmetry $a_{(R)}$ for the R rule is given by

$$a_{(R)} = 1 - \frac{1}{d}\sum_{i=1}^{d} a_i \quad (3)$$

Obviously, the closer CR and CP are, the more uniformly distributed around the center of gravity the patterns are.

2) Rule density and rule coverage

Broadly speaking, if A is a region in a d-dimensional feature space defined by the rule R, then it can be represented as the intersection of the intervals: $(lb_1 \leq \mathbf{x_1} \leq ub_1)$, … , $(lb_d \leq \mathbf{x_d} \leq ub_d)$, where lb_i and ub_i are the lower and upper boundaries of feature x_i. The d-dimensional volume V_R is defined as the product of the d sides $V_R = l_1 * l_2 * ... * l_d$, where $l_i = ub_i - lb_i$. Assuming that the region A which, is defined by the rule R contains n_R patterns, then the density of rule R is as follows:

$$d_{(R)} = \frac{n_R}{V_R} \quad (4)$$

Each rule R is assigned with another metric called rule coverage $cov_{(R)}$

$$\mathrm{cov}_{(R)} = \frac{n_R}{n_{total}} \quad (5)$$

where, n_{total} denotes the total number of patterns. All the rule-related concepts are combined into a single function to evaluate the weight f_R of rule R when calculating the fitness value of the entire chromosome

$$f_R = a_{(R)} * \mathrm{cov}_{(R)} * d_{(R)} \quad (6)$$

3) Rule homogeny

In real-world datasets, highly compact and closely located clusters can be covered by the same rule. The GA must be able to identify and quantify discontinuities in the distribution of patterns for all dimensions and then to combine the derived information in order to generate for each rule R a homogeny coefficient $h_{(R)}$. In order to assess rule's homogeny, for each dimension d the interval defined by the rule is subdivided into a number of bins $Bins_{(d)}$. The optimal upper bound for the width $W_{(d)}$ of the bins is given by the following equation [9] :

$$W_{(d)} \leq 3.729 * \sigma_d * n_R^{-1/3} \quad (7)$$

where, σ_d is the standard deviation of the patterns belonging to rule R in d dimension. For each bin the algorithm calculates its coverage, which is simply the number of patterns within the bin divided by the n_R. Bins with coverage below a threshold t_{sparse} are considered as sparse and therefore the homogeny coefficient h_d for the d dimension of rule R is calculated as follows:

$$h_d = \frac{Number\_dense\_bins}{Total\_number\_bins} \quad (8)$$

In our experiments the value of t_{sparse} is the mean value of the interval defined between the mean and median value of the coverage metrics for all bins.

The homogeny coefficient $h_{(R)}$ for the entire rule is the mean value of the homogeny coefficient for all dimensions

$$h_{(R)} = \frac{1}{d}\sum_{i=1}^{d} h_i \quad (9)$$

Equation 6 can now be enchanced by taking into consideration the factor $h_{(R)}$. Hence, the weight of a rule R can now be listed as

$$f_R = a_{(R)} * \mathrm{cov}_{(R)} * d_{(R)} * h_{(R)} \quad (10)$$

In order to prevent the generation of offsprings, which cover the entire search space equation, some kind of penalty for "very" large rules should be imposed. This can easily be done by simply multiplying equation (10) with the factor:

$$\left(1 - V_R / V_{Domain}\right) \quad (11)$$

where V_{Domain} denotes the maximum volume of the entire d-dimensional domain. Hence, each rule is assigned a weight, which is given by the following equation:

$$f_R = a_{(R)} * \mathrm{cov}_{(R)} * d_{(R)} * h_{(R)} * \left(1 - V_R / V_{Domain}\right) \quad (12)$$

4) Overlapping rules

During the evolution of individuals overlapping rules may occur, causing confusion about the most appropriate cluster to which it should be assigned. In an attempt to penalize individuals containing overlapping rules, a weighted pattern coefficient $N_{w(R)} = N_{1(R)} + \frac{N_{2(R)}}{2} + ... + \frac{N_{k(R)}}{k}$ replaces the total number of patterns n_R for each rule, where $N_{k(R)}$ represent the number of patterns covered by this rule and (k-1) additional rules. This appears in all the above equations. The factor introduced in equation 11 favours "small" rules and can cause a premature convergence in a suboptimal solution, because in relatively few generations individuals containing small rules dominate the population. A way to avoid this problem is by assigning the entire individual a pattern coverage factor P_{cov} defined as follows:

$$P_{COV} = \frac{N_1 + N_2/2 + ... + N_k/k}{n_{total}} \quad (13)$$

where N_k denotes the total number of patterns covered by k rules.

5) Final form of the fitness function

In our investigation, rules that contain fewer patterns than a user-defined threshold a (e.g. 5% of the total number of patterns), are considered as sparse and are excluded from the evaluation procedure described above. Furthermore in order to avoid the generation of empty or sparse rules, a penalty based on the number of sparse clusters is imposed. So the

total fitness function used in our experiments takes the following form

$$F_{eval} = \frac{P_{COV}}{NDR} * \sum_{i=1}^{NDR} f_{(i)} - \left(\left(\frac{NSR}{TNR} \right) * \left(\frac{P_{COV}}{NDR} * \sum_{i=1}^{NDR} f_{(i)} \right) \right) \quad (14)$$

where *NDR* is the number of dense rules included in the individual, *NSR* is the total number of sparse rules and *TNR* is the total number of rules.

C. Crossover Operator

The crossover operator used in our experiments is an extension of a standard two-point crossover [10]. Recall that each individual consists of a set of *k* clustering rules. The number of rules is fixed and corresponds to the number of desired clusters. Each rule is constituted from *d* genes, where each gene corresponds to an interval involving one feature. Each *i*-th gene, *i=1,..,d,* of the of a rule is subdivided into two field: *lower boundary* (lb_i) and *upper boundary* (ub_i), where *lb* and *ub* denotes the lower and upper value of the *i*-th feature in this rule. The conditional part of a rule is formed by the conjunction (logical *AND*). Two-point crossover is applied to each of the *k* rules of the mating parents generating two offsprings having fixed size. The idea of performing *k* crossovers is to enable each rule to move independently of the others. By adjusting independently each rule every generation the number of generations needed for the algorithm to converge is expected to be significantly reduced.

D. Mutation Operator

When a binary representation is used, the mutation operator flips the value (bits) of each gene. Our elaborate representation requires more complex mutation operator, which can be able to cope with non-binary genes.

Our mutation operator extends the step-size control mechanism for mutating real-valued vectors, suggested by Michalewicz [10]. The intuitive idea behind Michalewicz's mutation operator is to perform uniform mutation initially and very local mutation at later stages. Recall that each rule contains *d* intervals of the form *[lb, ub]*. The application of the mutation operator in such kind of genes with domain [a, b] is a two-step process: a) mutation of the *lb* and b) mutation of *ub.*

1) Mutation of the left boundary lb

If the operator is applied at generation t then the new *lb* of the gene is given by the following equation:

$$lb_{(t)} = \begin{cases} lb_{(t-1)} + \Delta\left(t, b - lb_{(t-1)}\right) & if \quad \tau = 0 \\ lb_{(t-1)} + \Delta\left(t, lb_{(t-1)} - a\right) & if \quad \tau = 1 \end{cases} \quad (15)$$

where *lb(t-1)* is the value of the left boundary in generation *(t-1)* and τ is a random number that may take value 0 or 1. Function Δ provides the mutation operator the capability of performing a uniform search initially, and very local search at later stages. More precisely,

$$\Delta(t, y) = y * \left(1 - r^{\left(1 - \frac{t}{T}\right)^b} \right) \quad (16)$$

where *r* is a random number from the interval [0,1] and *b* is a user-defined parameter, which determines the degree of dependency on the number of generations *T*. The function Δ returns a value in the range [0, y] such that the probability of returning a number close to 0 increases as the search progresses.

2) Mutation of the right boundary ub

The right boundary *ub* of the gene is mutated by applying the same method as in case of *lb*. The only difference is that $a = lb_{(t)}$, in order to ensure that always

$$lb_{(t)} < ub_{(t)} \quad (17)$$

E. Setting Parameters

The crossover operator is applied with a probability 90%. The mutation rate is set to 0.9, and the probability of mutating the value of a gene is 0.1. Although, Michalewicz suggested an optimal value for the parameter b=5.0, we found that the value of b=3.0 ensures population diversity.

Finally, to ensure population diversity at the later stages of the search (when the factor b impose local search) a random mutation in the value of the each gene is introduced with a very small probability (e.q 0.005). The selection strategy that is used in our experiments is a k-fold stochastic tournament selection, with tournament size k=2. We also used an elitism reproduction strategy where the best individual of each generation was passed unaltered to the next generation. The maximum number of generations was 200, which is the only stopping criterion used in our experimentations. Finally, the population size is 50 individuals. Knowing the mean value m_d and the standard deviation σ_d of the patterns for the *d* dimension the corresponding gene is randomly initialised within the interval $[(m_d - 4 * \sigma_d), (m_d + 4 * \sigma_d)]$ (18)

V. EXPERIMENTAL RESULTS

We report the results of experiments with two data sets, namely DS1 and DS2, which contain patterns in two dimensions (Fig. 3). The number of patterns and the shape of clusters in each data set are described in table I. Both data sets are synthetically generated based on, but not identical to published data sets: DS1 is from [11], and DS2 is from [12].

TABLE I
DATA SETS

	Number of Patterns	Shape of Clusters	Number of clusters
DS1	1120 noise=0%	One big and two small Circles with the same density	3
DS2	1650 noise=18%	Various, including ellipsoid, triangular, rectangular	4

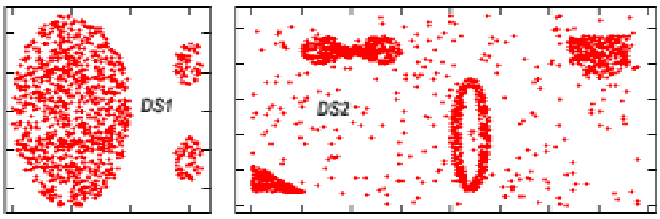


Fig. 3. Data sets 1 and 2.

DS1 contains one big and two small circles with the same density. We applied an implementation of the standard k-means algorithm in DS1 and the clustering results, given in Fig 4 (a), are similar to the result reported in [11].

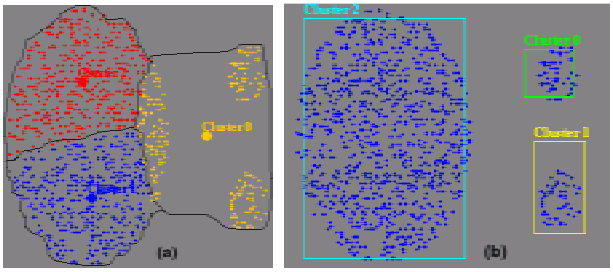


Fig. 4. Clustering results for DS1 using k-means (a) and RBCGA (b)

The k-means method splits the large cluster in order to minimize the square-error [11]. RBCGA was tested against DS1 50 times to find k=3 partitions. The set of parameters used for the genetic operators were as defined in section IV (E. Setting Parameters). In every 50 repetition RBCGA generated clusters similar to those illustrated in Fig. 4 (b). In contrast to k-means, RBCGA never splits the big cluster into smaller ones. We increased the density of one of the small cluster by a factor 3 and the clustering results were similar. The mean fitness value derived from 50 repetitions is around 0.019. This is an indication that for DS1 our algorithm is not sensitive to the initialization phase and always finds a solution close to the global optimum. Finally, the convergence speed of RBCGA for the DS1 is around 30 generations. The relatively small number of generations needed for the RBCGA to find the best solution is probably due to the absence of noise. Recall that the fitness function reduces the weight for rules that contain bins, which are sparse, and the more the noise lower the probability of sparse bins.

DS2 contains clusters of various shapes, densities and sizes. In addition, DS2 has random outliers scattered in the entire space (table I). We evaluated the effectiveness of RBCGA in discovering different types of distributions of data. The four clusters illustrated in Fig. 3 are well separated each other and the k-means algorithm usually reveals the clusters shown in Fig. 5(a), but may produce different results depending on the initialization phase and the presence of outliers (Fig. 5 (b)).

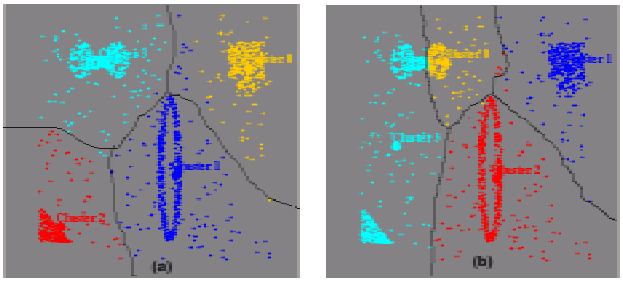


Fig. 5. Sensitivity of k-means to initialization and outliers.

We run RBCGA 50 times against the DS2 using the set of parameters given in section IV (E. Setting Parameters), and a typical result is illustrated in Fig. 6.

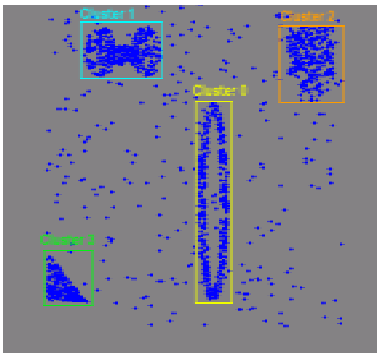


Fig. 6. Rules Discovered by RBCGA.

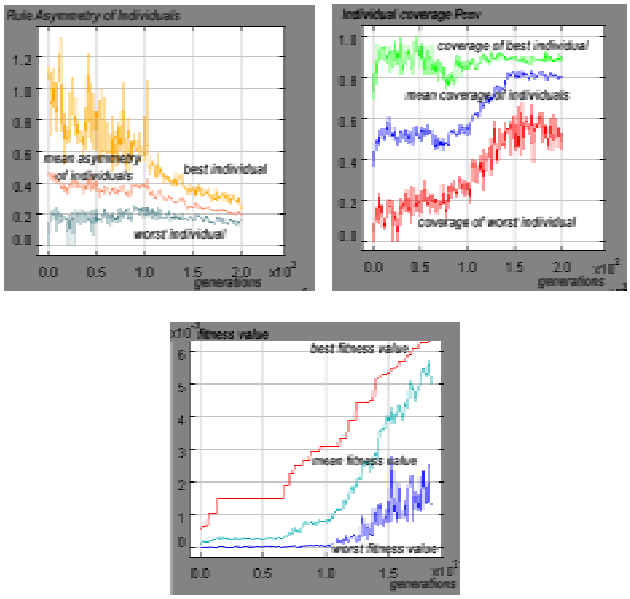


Fig. 7. Various statistics metrics related to individuals.

It is worthwhile to report that RBCGA always converges to solutions where there are no overlapping rules. The asymmetry graph in Fig. 7 depicts the mean value of asymmetry for all rules regarding the best and worst individuals. Clearly, the fitness function biases the evolutionary search for solutions that contains rules with relatively low asymmetry. The coverage graph in Fig. 7 corresponding to individual coverage factor P_{cov} indicates that the best individual in each generation always has the

highest P_{cov}. Finally, the convergence speed of RBCGA for the DS2 is around 150 generations, which is considerably higher than for DS1. We suspect that the large number of generations in case of DS2 is due to the relatively high ratio of noise (18%).

VI. CONLCUSIONS AND FUTURE WORK

We have presented a genetic rule-based algorithm for data clustering. RBCGA evolves individuals consisting of a fixed set of clustering rules, where each rule includes *d* non-binary intervals, one for each feature. A flexible fitness function is presented which takes into consideration various factors in order to maximise interclass dissimilarity and intraclass similarity.

The preliminary experimental results reported in this paper show that RBCGA can discover clusters of various shapes, sizes and densities. Furthermore, it appears that RBCGA never splits a large cluster into smaller ones, which is not the case with the standard k-means algorithm. Another important characteristic of RBCGA is its insensitivity to the initialization phase: it always found solutions close to the global optima.

Unfortunately, RBCGA does not easily scale up is concerned. This is due to the form of the fitness function, which is computational, very expensive regarding the total number of patterns. Future work should target to improve the scalability of RBCGA. This can be achieved by adopting the idea of bins in order to replace the raw data with bins. Another possible extension of the current work might be the attempt to use multi-objective optimization approaches in order to handle all the rule-related factors discusses earlier with a different weight.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the provision of the EOS software by BT's Intelligent Systems Laboratory.

REFERENCES

[1] J. Han and M. Kamber, "Data Mining: Concepts and Techniques," Morgan Kaufman Publishers, 2000.

[2] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy, "Advances in Knowledge Discovery and Data Mining", AAAI Press/The MIT Press, 1996.

[3] R.O. Duda and P.E Hart, "Pattern Classification and Scene Analysis", John Wiley & Sons, NY, USA, 1973.

[4] A.K. Jain and R.C. Dubes, "Algorithms for Clustering Data", Prentice-Hall, Englewood Cliffs, NJ, 1988.

[5] L. Kaufman, "Finding groups in data: an introduction to cluster analysis", Wiley, New York, 1990.

[6] E. Bonsma, M. Shackleton and R. Shipman, "Eos - an evolutionary and ecosystem research platform", BT Technology Journal, 18(14):24-31, 2000.

[7] Estivill-Castro, "Hybrid genetic algorithms are better for spatial clustering", Technical Report 99-08, Department of Computer Science and Software Engineering, The University of Newcastle, Callaghan, 2308 NSW, Australia, 1998.

[8] David E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison-Wesley, Reading, Massachusetts, 1989.

[9] David W.Scott, "Multivariate Density Estimation: Theory", Practice and Visualization (John Wiley, New York, NY) 1992.

[10] Zbigniew Michalewicz, "Genetic Algorithms + Data Structures = Evolution Program", Third Edition, Springer-Verlag, 1996.

[11] S. Guha, R. Rastogi, and K. Shim, "CURE: An efficient clustering algorithm for large databases", In Proceedings of ACM SIGMOD International Conference on Management of Data, pages 73--84, New York, 1998.

[12] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling", IEEE Computer 32, pp. 68-75, August 1999.

[13] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases", In Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, pp. 103--114, Montreal, Canada, 1996.

[14] A. Hinneburg, D.A Keim, "An Efficient Approach to Clustering in Large Multimedia Databases with Noise", In Proceedings of the 4rd International Conference on Knowledge Discovery and Data Mining, AAAI Press, 1998.

[15] W. Wang, J. Yang, and R. R. Muntz, "STING: A Statistical Information Grid Approach to Spatial Data Mining", In Proceedings of the 23rd VLDB, pp. 186-195, Athens, Greece, 1997.

[16] K. Krishna and M. Murty, "Genetic K-means algorithm", IEEE Transactions on Systems, Man, and Cybernetics - PartB: Cybernetics, 29(3):433—439, 1999.

[17] L. O. Hall, I. B. Ozyurt, and J.C. Bezdek, "Clustering with a genetically optimized approach", *IEEE Transactions on evolutionary Computation*, 3(2):103--112, July 1999.