Mining Comprehensive Clustering Rules with an Evolutionary Algorithm

Ioannis Sarafis[1], Phil Trinder[1], and Ali Zalzala[2]

[1] School of Mathematics and Computer Science, Heriot-Watt University
Riccarton Campus, Edinburgh, EH14 4AS Scotland, United Kingdom
{I.Sarafis, P.W.Trinder}@hw.ac.uk
[2] School of Engineering & Physical Sciences, Heriot-Watt University
Riccarton Campus, Edinburgh, EH14 4AS Scotland, United Kingdom
A.Zalzala@hw.ac.uk

Abstract. In this paper, we present a novel evolutionary algorithm, called NOCEA, which is suitable for Data Mining (DM) clustering applications. NOCEA evolves individuals that consist of a variable number of non-overlapping clustering rules, where each rule includes d intervals, one for each feature. The encoding scheme is non-binary as the values for the boundaries of the intervals are drawn from discrete domains, which reflect the automatic quantization of the feature space. NOCEA uses a simple fitness function, which is radically different from any distance-based criterion function suggested so far. A density-based merging operator combines adjacent rules forming the genuine clusters in data. NOCEA has been evaluated on challenging datasets and we present results showing that it meets many of the requirements for DM clustering, such as ability to discover clusters of different shapes, sizes, and densities. Moreover, NOCEA is independent of the order of input data and insensitive to the presence of outliers, and to initialization phase. Finally, the discovered knowledge is presented as a set of non-overlapping clustering rules, contributing to the interpretability of the results.

1 Introduction

Clustering is a common data analysis task that aims to partition a collection of objects into homogeneous groups, called clusters [8]. Objects assigned to the same cluster exhibits high similarity among themselves and are dissimilar to objects belonging to other clusters. Although, the idea of representing a group of similar object using its cluster descriptor loses fine details, it is very useful especially for large and high dimensional datasets to provide a simplification of the underlying distribution of objects. The challenging field of DM clustering led to the emergence of various clustering algorithms [6]. Evolutionary Algorithms (EAs) are optimization techniques that have been inspired from the biological evolution of species [3], [10]. NOCEA (*Non-Overlapping Clustering with an Evolutionary Algorithm*) employs the powerful search mechanism of EAs to meet some of the requirements for DM clustering such as ability to discover clusters of different shapes, sizes, and densities, independency of the order of input data, insensitivity to the presence of outliers and to initialization phase and interpretability of the results [6].

2 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 [8]. Hierarchical algorithms create a hierarchical decomposition of the database that can be presented as dendrogram [12]. Density-based algorithms search for regions in the data space that are denser than a threshold and form clusters from these dense regions [7]. Grid-based algorithms quantize the search space into a finite number of cells and then operate on the quantized space [1]. EAs have been proposed for clustering, because they avoid local optima and are insensitive to the initialization [5], [11].

3 NOCEA Clustering Algorithm

3.1 Bin Quantization

Let $A = \{A_1, ..., A_d\}$ be a set of bounded domains and $S = A_1 \times ... \times A_d$ is a d-dimensional numerical space. The input consists of a set of d-dimensional *patterns* $P = \{p_1, ..., p_k\}$, where each p_i is a vector containing d numerical values, $p_i = [\alpha_1, ..., \alpha_d]$. The *j*th component of vector p_i is drawn from domain A_j. NOCEA's clustering mechanism based on a statistical decomposition of the feature space into a multi-dimensional grid. Each dimension is divided into a finite number of intervals, called *bins*. The number of bins m_j and the bin width h_j for *j*th dimension is dynamically computed by projecting the patterns in this dimension and then applying the statistical analysis described below. Initially, each dimension is divided into four segments, namely A, B, C and D. Segments A, B, C and D represent the intervals [a, Q1), [Q1, median), [median, Q3) and [Q3, b], respectively. Note that, *a*, *b* are the left and right bounds of A_j, while *median*, *Q1* and *Q3* are the median, first and third quartiles of patterns in jth dimension, respectively. The bin width h for each segment is then computed using the following formula [2]:

$$h = 3.729 * \sigma * n^{-\frac{1}{3}} \quad (1)$$

where, n is the number of patterns inside this segment, while σ denotes the standard deviation. The number of bins for each segment is derived by dividing the length of the segment over the corresponding bin width h. The total number of bins m_j for the jth dimension is computed by forming the sum of bins from the four segments. Finally, h_j is obtained by dividing the length of A_j domain over m_j.

3.2 Individual Encoding

The proposed encoding scheme is a non-binary, rule-based representation, containing a variable number of non overlapping rules. Each rule comprises d genes, where each gene corresponds to an interval involving one feature. Each ith gene, where i=1,..,d of a rule, is subdivided into two fields: *left boundary* (lb_i) and *right boundary* (rb_i), where lb_i and rb_i denotes the lower and upper value of the ith feature in this rule. The boundaries are drawn from discrete domains, reflecting the grid-based decomposition of the feature search. Note that two rules are non-overlapping if there exists at least one dimension where there is no intersection between the corresponding genes.

3.3 Fitness Function

In our clustering context, the fitness function is greedy with respect to the number of patterns that are covered by the rules of individuals. In particular, NOCEA aims to maximize the *coverage* C, which is defined to be the fraction of *total patterns* N_{total} that are covered by the rules of the individuals.

$$C = max\left(\frac{\sum_{i=1}^{k} N_i}{N_{total}}\right) \tag{2}$$

where, k denotes the number of rules and N_i the number of patterns in ith rule. The above fitness function is suitable for comparing individuals that have different number of rules. Two individuals can have exactly the same performance with radically different genetic material (e.g. number, shape and relative position of rules). This fact contributes to high diversity among the population members. Another important characteristic of the above fitness function is the fact that its theoretical lowest and highest values are always known, that is, the fitness of an individual can be between zero and one. Theoretically, as the level of noise (fraction of total patterns that are not really belong to any particular cluster) increases the distance between the performance of best individuals and 1.0 increases.

3.4 Recombination Operator

In a typical EA the recombination operator is used to exploit known solutions by exchanging genetic material between good individuals. Taking into consideration the requirement for evolving individuals without overlapping rules elaborate one and two point crossover operators have been developed. The crossover operator is applied to two parents generating two offsprings, from which only one can survive. Firstly, a number j is randomly drawn from the domain [1..d]. Recall, that d denotes the dimensionality of the feature space. If m_j is the number of bins corresponding to jth dimension, the crossover points *cp1* and *cp2* are determined by randomly picking one bin from the interval [0, $(m_j - 1)$]. Two offsprings are generated by exchanging the rules lying in the region from cp1 to cp2, between the parents. An example of two-point crossover is depicted in figure 1. Note that if the number of crossover points is one then the above mentioned crossover operation reduces to one-point recombination. The proposed crossover operator

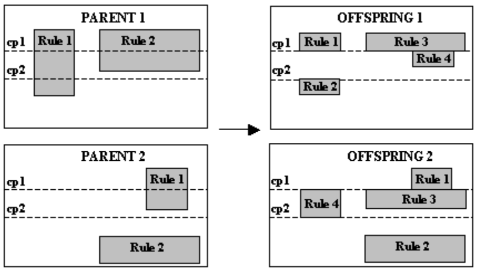


Fig. 1. An example of two-point crossover operation

can lead to the generation of offsprings that are not of the same length as their parents. This is because the offsprings must contain no overlapping rules, which in turn requires the splitting of any rule that intersects with the d-dimensional region defined by the two crossover points. Clearly, for very low dimensional datasets and high recombination rates, two or even one point crossover can be disruptive in terms of breaking a relatively large rule into a number of smaller rules. Obviously, the greater the number of rules the greater the computational complexity as far as the application of genetic operators is concerned. Moreover, for the gain of simplicity in describing the clusters, fewer rules are always preferable. On the other hand, the length-changing characteristic of the crossover operator can contribute to increased diversity among the population members and this can be useful especially in cases with arbitrary-shaped clusters. To merely cope with the disruptive effect of crossover operator, we apply the following heuristic in the reported experiments. Instead of arbitrarly selecting the crossover points in *j*th dimension from the interval $[0, (m_j - 1)]$, these points are now fixed and they correspond to the bins containing Q1, Q3 or median in *j*th dimension. Theoretically, the disruptive effect of crossover is reduced as the dimensionality of the dataset increases, because the probability of intersection between the crossover points and rules decreases. As a consequence, for very low dimensional datasets (2D or 3D) it is reasonable to set the ratio of crossover in relatively small values or even to deactivate the recombination operator.

3.5 Mutation Operator

The evolutionary search in our system is mainly based on an elaborate mutation operator, which alters the genetic material randomly, but ensures the generation of individuals without overlapping rules. The mutation operator has two functionalities: a) to grow and b) to shrink an existing rule.

Growing-Mutation The growing-mutation aims to increase the size of existing rules in an attempt to cover as many patterns as possible but with a minimum number of rules. It is particularly useful in cases where there are large and convex clusters that can be easily captured by a relatively small number of large rules. These large rules can be generated starting from smaller ones and by applying the growing-mutation operator over the generations. It is reasonable to focus on the discovery of as few and as large rules as possible, because these kind of rules contribute to the high interpretability of the clustering output. Additionally, the combined functionalities of the growing-mutation and the repair operator (see section 3.6) can lead to the discovery of new interesting regions (isolated clusters) within the d-dimensional feature space. Lets assume that the jth dimension of a d-dimensional rule R undergoes growing-mutation. We can compute the maximum value $rb_{j(max)}$ that rb_j can be extended to the right without causing overlapping as follows:

1. Sort in ascending order all the rules such that their left bound is greater than the right bound of rule R in jth dimension.
2. If the sorted list is empty, set $rb_{j(max)}$ to $(m_j - 1)$ and exit. Otherwise, proceed to step 3.
3. Pick the next element RN of the list. If there is intersection in at least one dimension (excluding the jth) between the rules R and RN, set $rb_{j(max)}$ equal to the left bound of RN in jth dimension minus one and terminate the loop. If there is no intersection, proceed to the next element.

The derivation of the minimum value $lb_{j(min)}$ that rb_j can be extended to the left without causing overlapping is the dual of computing $rb_{j(max)}$. Obviously, the allowable range of modification in the boundaries of rules due to growing mutation is solely determined by the relative position of rules within the d-dimensional feature space. The left and right bounds of each rule in jth dimension are randomly mutated within the intervals $[lb_{j(min)}, lb_j)$ and $(rb_j, rb_{j(max)}]$, respectively.
Shrinking-Mutation This type of mutation as its name implies shrinks an existing rule. Each time, the bound of a rule undergoes shrinking-mutation, the corresponding bound shrinks by one bin. The intuition behind this small modification is that for high dimensional datasets where the existence of isolated rules is increased, an arbitrary shrinking-mutation operation can cause the elimination of the rule and as a consequence additional generations may be required to re-discover this region of the feature space. The shrinking-mutation operator is particularly useful to perform local fine-tuning and to facilitate the precise capturing of non-convex clusters. This is done by allowing adjacent rules to be re-arranged within the d-dimensional grid in order to cover as many patterns as possible.
Balancing shrinking and growing The ratio of shrinking to growing operations r_{sg} should be set to relatively small values (e.g. 0.05). This is mainly done to bias the evolutionary search to discover new interesting regions in the large d-dimensional feature space, rather than trying to perform fine local tunning. Furthermore, due to the replacement strategy (see section 3.8) that allows the surviving of individuals for only a certain number of generations, high values for r_{sg} may be an obstacle for NOCEA to converge in an optimal solution.

3.6 Repair Operator

Often the search space S consists of two disjoint subsets of feasible and infeasible solutions, F and U, respectively [10]. Infeasible solutions are those that violate at least one constraint of the problem and we have to deal with them very carefully, because their presence in the population influences other parts of the evolutionary search [10]. In the following we introduce some basic definitions in order to formalize the notion of feasibility in our clustering context. The *selectivity* s_{ij} of ith bin in jth dimension is defined to be the number of patterns lying inside this bin over the total number of patterns covered by this rule. We define as *Selectivity Control Chart* (*SCC*) in jth dimension a diagram that has a centerline *CL*, and an upper (*UCL*) and a lower (*LCL*) control limits that are symmetric about the centerline. Additionally, measurements corresponding to selectivity of bins in jth dimension are plotted on the chart. CL, UCL and LCL are given by equation 3.

$$\begin{aligned} \text{UCL} &= CL * (1+t) \\ \text{CL} &= \frac{1}{b_j} \\ \text{LCL} &= CL * (1-t) \end{aligned} \tag{3}$$

where, b_j denotes the number of bins in jth dimension and t is a tolerance threshold that controls the sensitivity of SCC chart in detecting shifts in the selectivity of bins.

Consider the two dimensional feature space shown in figure 2(a), where each dimension has been partitioned into a number of bins. Figures 2(c) and 2(d) illustrate the SCC charts for dimensions x and y of a rule R1 shown in figure 2(a), respectively. If the patterns covered by R1 are projected to x-axis, there are two distinct and well separated regions. These regions can be easily detected by examing the corresponding SCC chart shown in figure 2(c), where the selectivity of the third is below LCL. In contrast, the selectivity for each bin in the y-axis is within the control limits and as a consequence NOCEA detects a single region in y-axis. In our approach, for each dimension of a rule R we construct a SCC diagram. If there exist points below LCL, then the corresponding bins are relatively sparse with respect to the rest and can be discarded. On the other hand, if there are no points below LCL but there exists at least one bin with selectivity over UCL, then NOCEA reduces t gradually by subtracting a small constant number, until at least one point falls below LCL. If neither of the above conditions are true, the distribution of patterns across this dimension can be considered as uniform and as a consequence all bins are kept. The removal of sparse bins described above creates a set of intervals for each dimension. The original rule is replaced by a set of new rules, which are formed by combining one interval from each dimension. A rule R is said to be *solid*, if the selectivity of all bins for each dimension is between the corresponding LCL and UCL. An individual that contains only solid rules is a feasible solution in our clustering context. Finally, if the selectivity of a rule, that is the fraction of total patterns covered by this rule, is below a user-defined threshold t_{sparse}, then this rule is said to be *sparse* and is eliminated. Theoretically, for relatively large values of t, regardless the underlying distribution of patterns, each rule will be solid. In such

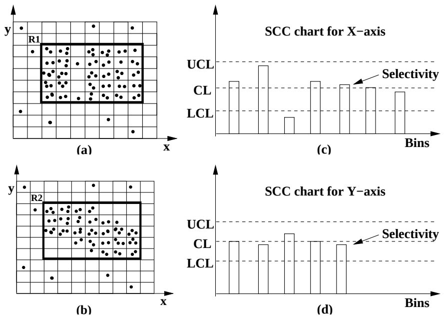


Fig. 2. Visualization of SCC charts

a case, NOCEA can not produce homogeneous rules because in essence there is no mechanism to stop rules from growing arbitrarily. On the other hand, relatively small values for t can cause over-triggering of the repair operator resulting in the generation of many small rules, which in turn add extra complexity to the application of the genetic operators. To merely cope with the problem of fine tunning the parameter t, we suggest a step-size adaptation mechanism that increases t over the generations according to:

$$t_i = t_{min} + (t_{max} - t_{min}) * \left(1 - \left(\frac{i}{T}\right)^b\right) \tag{4}$$

where, b is a user-defined parameter, t_i denotes the value of t in ith generation, T is the total number of generations, while t_{min} and t_{max} are input parameters corresponding to minimum and maximum values of t, respectively.
A drawback of SCC charts is the fact that consider each dimension independently and a consequence if a rule R contains partially sparse bins, these discontinuities in the distribution of patterns can not be detected. For instance, consider the rule R2 that is shown in figure 2(b). Assuming relatively large values for t the corresponding SCC charts, which are not shown here, can not detect the very sparse regions located in the bottom-left and top-right of rule R2. To merely cope with partially sparse bins we apply the following heuristic to each rule, periodically (e.g. every 40 generations). Initially, a dimension that contains at least six bins is randomly selected. Then the parent rule R is split in that dimension in the middle and forms two touching rules R' and R''. Both these rules undergo the repairing procedure described above. If these rules remain intact as far as their size is concerned, another dimension which has not previously been examined and which contains at least six bins is randomly chosen. However, if the repair operator modifies at least one of these rules, the parent rule R is replaced with the set of rules that has been produced after repairing both rules, R' and R''. Note that we only consider dimensions containing at least six bins to avoid local abnormalities in the derived rules R' and R''. For the same reason the repair operator is invoked using t_{max} as the value for the tolerance threshold t.

3.7 Merging Rules

As soon as NOCEA converges, a merging procedure that combines adjacent rules in order to form the genuine clusters in data, is activated. The merging procedure based on the concept of density and it is different from other distance-based merging approaches [4], [9]. Two rules are adjacent if they have a common face, i.e there are d-1 dimensions where there is an intersection in at least one bin between the corresponding genes and there is one dimension where the left bound of one rule is adjacent to the right bound of the other rule. For instance, rules R1 and R2 shown in figure 3 are adjacent because there is an intersection between their corresponding genes in y-axis, and the right and left bound of rules R1 and R2, respectively, are touching. For each pair of adjacent rules R1 and R2 we

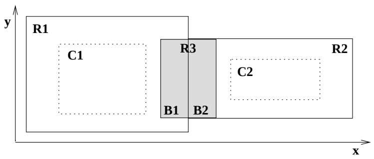


Fig. 3. An example of adjacent rules (R1 and R2).

compute five density metrics, namely, d_{C1}, d_{C2}, d_{B1}, d_{B2} and d_{R3}. These density metrics correspond to the density of patterns within the d-dimensional regions $C1$, $C2$, $B1$, $B2$ and $R3$, respectively, which are shown in figure 3. R3 (shadowed region) represents the interface layer between the two rules R1 and R2, while C1 and C2 denote the central regions within the rules R1 and R2, respectively. NOCEA assigns a pair of adjacent rules R1 and R2 to the same cluster as long as the following conditions are satisfied:

$$\begin{array}{ll} \left(\frac{\min(d_{C1},d_{B1})}{\max(d_{C1},d_{B1})}\right) \geq k \ \ (a), & \left(\frac{\min(d_{C1},d_{C2})}{\max(d_{C1},d_{C2})}\right) \geq k \ \ (d) \\ \left(\frac{\min(d_{C2},d_{B2})}{\max(d_{C2},d_{B2})}\right) \geq k \ \ (b), & \left(\frac{\min(d_{C1},d_{R3})}{\max(d_{C1},d_{R3})}\right) \geq k \ \ (e) \\ \left(\frac{\min(d_{B1},d_{B2})}{\max(d_{B1},d_{B2})}\right) \geq k \ \ (c), & \left(\frac{\min(d_{C2},d_{R3})}{\max(d_{C2},d_{R3})}\right) \geq k \ \ (f) \end{array} \qquad (5)$$

where k is a user-defined parameter. Conditions (a) and (b) ensure that the border (across the interface between R1 and R2) and central regions of rules R1 and R2 have similar densities. Condition (c) ensures that the two equi-sized regions B1 and B2 that comprise R3 have similar densities. Finally, conditions (d), (e) and (f) simply ensure that the central regions of R1, R2 and the interface rule R3 have similar densities.

3.8 Setting Parameters

The mutation ratio is set to 1.0, while the probability of mutating the boundaries of rules is 0.05. The shrinking to growing ratio r_{sg} is set to 0.05. We used one-point crossover with ratio 0.1. The selection strategy that is used in our experiments is a k-fold tournament selection with tournament size k=4. The replacement strategy implements best of all scheme by merging the current and offspring populations and selecting the best individuals. However, to avoid premature convergence, each individual is allowed to survive only for a certain number of generations (e.g. 5). Moreover, for every 40 generations all parents are replaced by the offsprings in order to eliminate all individuals containing partially sparse bins. Usually, the repairing of such rules cause a small reduction in the coverage of the entire individual, which can be easily observed from the fitness diagrams shown in figures 5(b), 5(d) and 5(f). The only stopping criterion used in the reported experiments is the maximum number of generations 200, while the population size is 50. Each population member is randomly initialized with a single d-dimensional rule. The parameters t_{min} and t_{max} and b, which control the adaptation of the tolerance threshold t are set to 0.4, 0.65 and 1.5, respectively. The threshold t_{sparse} for eliminating sparse rules is 0.05.

4 Experimental Results

In this section we report the experimental results derived by running NOCEA against three different datasets that were synthetically generated and containing patterns in 2D feature space as depicted in figure 4. These datasets are particularly challenging because they contain clusters of different sizes, shapes, densities and orientations. Furthermore, there are special artifacts such as streaks running across clusters and outliers that are randomly scattered in the features space. The first and second dataset, DS1, and DS2, respectively, contain 8000 patterns and were synthetically generated in order to evaluate the performance of a well-known clustering algorithm, called CHAMELEON [9]. The third dataset, DS3 has 100000 patterns and was used by another popular clustering algorithm called CURE [4]. Figure 5 illustrates the rules and clusters found by NOCEA for each one of the three datasets. Rules belonging to the same cluster are assigned the same number. Furthermore, for each dataset there is a fitness diagram where the coverage of the best and worst individuals as well as the mean coverage of the entire population in each generation are plotted. It can be observed from the figure 5 that NOCEA has the following desirable properties:

a) Discovery of non-convex clusters. NOCEA has the ability to discover arbitrary-shaped clusters, with different sizes and densities. The density-based merging procedure combines correctly adjacent rules forming the genuine clusters in data.

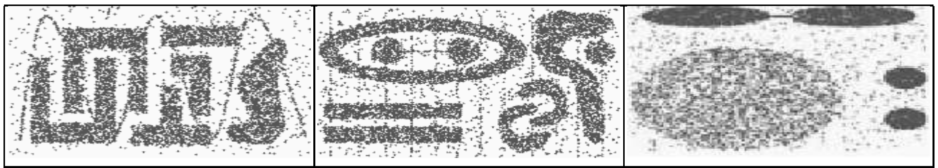
b) No a priori provision of the number of clusters. Unlike other clustering algorithms such as k-means [8], NOCEA does not require the provision of the number of clusters a-priori, because the merging operator discover the correct number of clusters on the fly.

c) Simplicity of fitness function. NOCEA has a simple fitness function which differs radically from distance-based criterion functions used in partitioning and hierarchical clustering algorithms[8]. Unlike other hierarchical and partitioning algorithms, NOCEA is not biased on splitting large clusters into smaller ones in order to minimize some distance criterion function [4].

d) Handling Outliers. Although NOCEA is greedy with respect to the number of patterns that are covered by the rules of the individuals, the fitness diagrams shown in figure 5 indicate that the maximum theoretical value of coverage 1.0, was never reached (assuming an amount of noise). This is due to the combined functionalities of the repair operator which do not allow a sparse region to be a member of a rule and to the procedure that eliminates very sparse rules. Thus, NOCEA is relatively insensitive to the presence of noise.

e) Data order and initialization independency. The form of fitness function which is not distance-based and the randomized search of NOCEA ensures independency to the order of input data and to the initialization phase.

f) Interpretability of the results. Whereas other clustering techniques describe the derived partitions by labeling the patterns with an identifier corresponding to the cluster that they have been assigned [6], NOCEA presents the discovered knowledge in the form of non-overlapping IF-THEN clustering rules, which have the advantage of being high-level, symbolic knowledge representation.



DS1: 8000 patterns	DS2: 8000 patterns	DS3: 100000 patterns

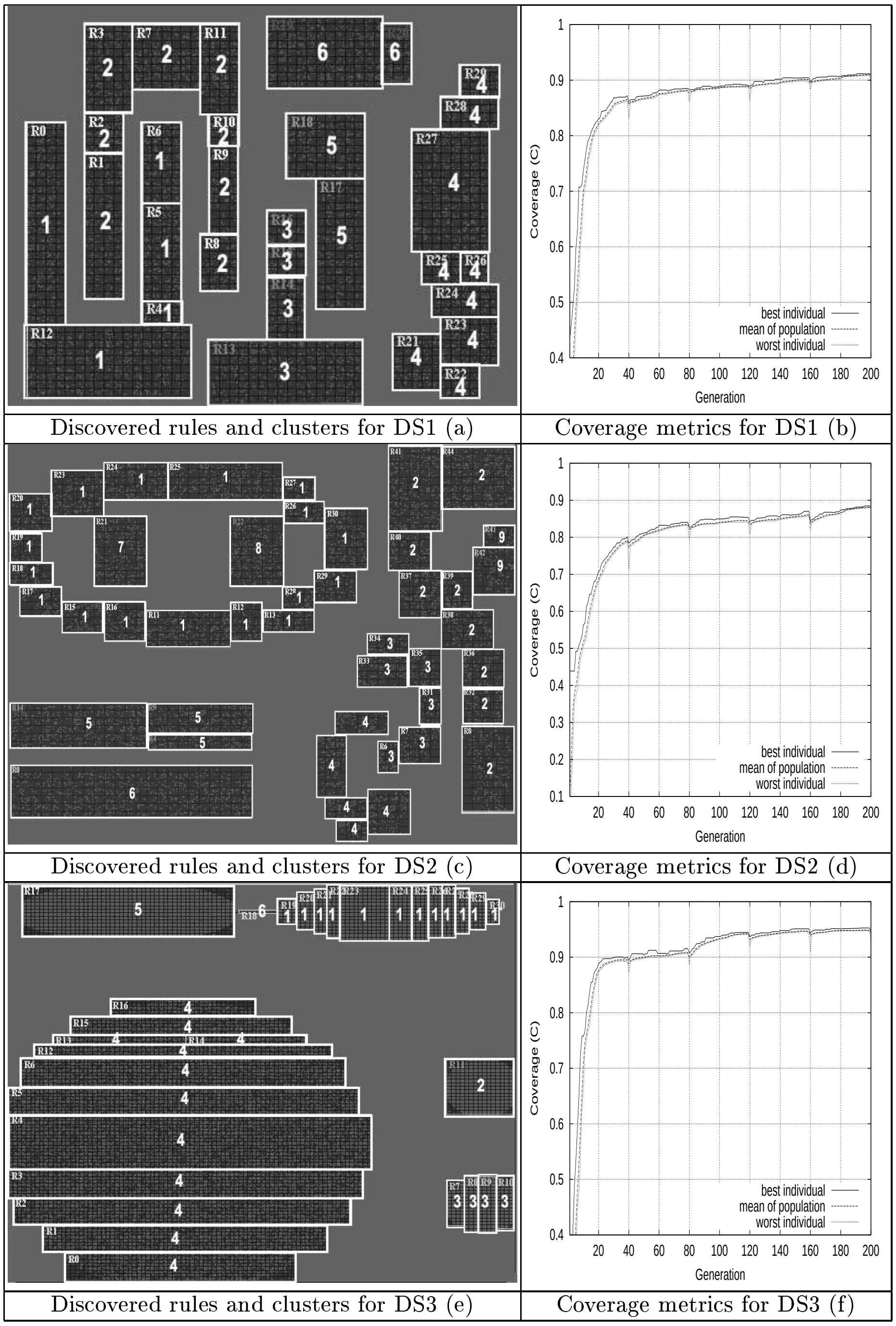
Fig. 4. The three datasets used in our experiments

5 Conclusions and Future Work

In this paper, we have presented a novel evolutionary algorithm called NOCEA, which is suitable for DM clustering applications. NOCEA evolves individuals that consist of a variable number of non-overlapping clustering rules, where each rule includes d intervals, one for each feature. The encoding scheme is non-binary as the values for the boundaries of the intervals are drawn from discrete domains. These discrete domains reflect the dynamic quantization of the feature space, which is based on information derived by analyzing the distribution of patterns in each dimension. We use a simple fitness function, which is radically different from any distance-based criterion function suggested so far. A density-based merging operator combines adjacent rules forming the correct number of clusters on the fly. Experimental results reported in section 4 indicate that the specific fitness function, together with, the elaborate genetic operators allow NOCEA to meet some of the requirements for DM clustering. NOCEA is an evolutionary algorithm and as a consequence does not easily scale up comparing to other hill-climbing clustering techniques [6]. However, EA are highly parallel procedures and ongoing work attempts to investigate the improvements in the efficiency of NOCEA using various schemes of parallelism. Additionally, ongoing work attempts to overcome the problem of splitting large rules into smaller ones when the crossover points intersect with rules, by extending the functionality of the recombination operator. Another possible extension can be the development of a generalization operator that can be used to reduce the complexity of the cluster descriptors. In particular this procedure will take as argument a set of clustering rules corresponding to a particular cluster and will produce a more general descriptor for it.

References

1. R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data*, pages 94–105, 1998.
2. Scott D. W. *Multivariate Density Estimation*. Wiley, New York, 1992.
3. D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.
4. S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD-98)*, volume 27,2 of *ACM SIGMOD Record*, pages 73–84. ACM Press, June 1–4 1998.
5. L. O. Hall, I. B. Özyurt, and J. C. Bezdek. Clustering with a genetically optimized approach. *IEEE Trans. on Evolutionary Computation*, 3(2):103–112, 1999.
6. M. Han, J.and Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, August 2000.
7. A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proc. 1998 International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pages 58–65. AAAI Press, 1998.
8. Jain, A.K., Dubes, R.C: . *Algorithms for Clustering Data*. Prentice-Hall, Englewood Cliffs, NJ, 1988.
9. G. Karypis, E-H. Han, and V. Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Computer*, 32(8):68–75, 1999.
10. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1996. Third Edition.
11. I. Sarafis, A. Zalzala, and P. Trinder. A genetic rule-based data clustering toolkit. In *Proc. Congress on Evolutionary Computation (CEC), Honolulu, USA*, 2002.
12. T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *Proceedings of theACM SIGMOD International Conference on Management of Data*, volume 25, 2, pages 103–114. ACM Press, 1996.



Discovered rules and clusters for DS1 (a) | Coverage metrics for DS1 (b)

Discovered rules and clusters for DS2 (c) | Coverage metrics for DS2 (d)

Discovered rules and clusters for DS3 (e) | Coverage metrics for DS3 (f)

Fig. 5. NOCEA's experimental results