Finding Median Partitions Using Information-Theoretical-Based Genetic Algorithms[1]

Dana Cristofor
Department of Computer Science, University of Massachusetts at Boston, Boston, Massachusetts, 02125
Email: dana@cs.umb.edu.

Dan Simovici
Department of Computer Science, University of Massachusetts at Boston, Boston, Massachusetts, 02125
Email: dsim@cs.umb.edu.

Abstract: In a database with categorical attributes, each attribute defines a partition whose classes can be regarded as natural clusters of rows. In this paper we focus on finding a partition of the rows of a given database, that is as close as possible to the partitions associated to each attribute. We evaluate the closeness of two partitions by using a generalization of the classical conditional entropy. From this perspective, we wish to construct a partition (referred to as the median partition) such that the sum of the dissimilarities between this partition and all the partitions determined by the attributes of the database is minimal. Then, the problem of finding the median partition is an optimization problem, over the space of all partitions of the rows of the database, for which we give an approximative solution. To search more efficiently the large space of possible partitions we use a genetic algorithm where the partitions are represented by chromosomes. Our genetic algorithm obtains better clustering results than the classical k-means algorithm.

Key Words: categorical attributes, clustering, genetic algorithm, median partition, partitioning, Shannon entropy, Gini index
Category: H2,E4

1 Introduction

Clustering is an important problem in many computer science domains and is particularly important in the context of data mining.

The clustering problem can be described as follows: given a set of N objects, characterized by n attributes, we want to group these objects such that objects which are similar to be placed in the same group and objects which are dissimilar to be placed in different groups.

Research in clustering has been focus more on data characterized by numerical or quantitative attributes, for which natural distances (like Euclidean or Manhattan distance) between the objects to be clustered are available. Clustering categorical or qualitative attributes presents an special challenge, since a natural ordering of the attributes values is not available. Previous research in this direction include algorithms introduced in [VJR99, SRK99, Hua97].

[1] C. S. Calude, K. Salomaa, S. Yu (eds.). *Advances and Trends in Automata and Formal Languages. A Collection of Papers in Honour of the 60th Birthday of Helmut Jürgensen.*

In this paper we focus on the case of categorical attributes databases and we investigate the application of information-theoretical methods to the clustering problem. The objects that we cluster are represented by tuples in a table and the goal of the genetic algorithm is to construct a partition of the set of rows whose classes are regarded as clusters of the rows. We require that this partition satisfy the following two conditions:

1. it must summarize the partitions generated by the attributes of the table.
2. the number of classes of the partition should not exceed a prescribed upper limit.

The first condition ensures that the resulting partition represents a clustering of the rows, while the second condition represents a restriction of the number of classes in this clustering.

In literature, the partition satisfying the first condition is called *median* partition or *consensus* partition and was analyzed in [BL95, Rég83a, Rég83b]. Research concerning the application of genetic algorithms to grouping problems was presented in [Hol92, Mic99, Fal99, Mit97, JB91].

Finding the median partition is an NP-complete problem as it was shown by [BL95]. That is why we focus in this paper on finding an approximative solution using a genetic algorithm approach. To determine how well a partition summarizes the attribute partitions of a table, we use a measure based on a generalization of the conditional entropy, that was introduced in [SCC00c, SCC00a].

The rest of the paper is organized as follows: Section 2 introduces definitions and notation, and presents a generalization of the classical entropy measure and important properties of this generalization. Section 3 describes the genetic algorithm that we use to search for the median partition. Section 4 presents the results of our experiments as well as an analysis of the implications of the different techniques used. Section 5 concludes the paper with a summarization of our results.

2 Definitions

2.1 Generalized entropy

The set of partitions of a set R is denoted by $\mathsf{PART}(R)$. For $\pi, \sigma \in \mathsf{PART}(R)$ we write $\pi \leq \sigma$ if every class of σ is a union of classes of π. We denote by $\omega_R \in \mathsf{PART}(R)$ the one-class partition, and by $\alpha_R \in \mathsf{PART}(R)$ the partition with one-element classes. We denote by $|\pi|$ the number of classes of the partition π.

Let $\mathcal{U}$ be a set whose elements are referred to as *attributes*. We assume that for every element A of $\mathcal{U}$ there is a set denoted by $\mathtt{Dom}(A)$ referred to as the *domain of* A such that $|\mathtt{Dom}(A)| \geq 2$.

A *table* is a function $T : R \times H \longrightarrow \bigcup D_H$, where R, a finite set, is referred to as the *set of rows of* T, H is a finite subset of $\mathcal{U}$ (called the *heading of* T), $D_H = \{\mathtt{Dom}(A) \mid A \in H\}$, and $T(r, A) \in \mathtt{Dom}(A)$ for every $r \in R$ and $A \in H$.

The *projection of the table* $T : R \times H \longrightarrow \bigcup D_H$ *on the set of attributes* $L \subseteq H$ is the table $T[L] : R \times L \longrightarrow \bigcup D_L$ given by $T[L](r, A) = T(r, A)$ for every $r \in R$ and $A \in L$. If r is a row of the table T, then the set $\{T(r, A) \mid A \in H\}$ is the *content* of the row r. Note that this definition of a table allows for the existence of multiple rows having the same content. The *projection of the row* r *of the table* T

on L is the set $r[L] = \{T(r, A) \mid A \in L\}$. For $A \in H$, $Q_T[A] = \{T(r, A) \mid r \in R\}$ represents the content of the column associated with the attribute A in the table T.

The *active domain of a set of attributes* L *in the table* T is the set $\mathtt{aDom}_T(L) = r[L]$. The cardinality of the active domain of L in the table T is denoted by a_L^T, or, when there is no risk for confusion, just by a_L.

Let $T : R \times H \longrightarrow \bigcup D_H$ be a table with n attributes and $H = A_1 A_2 \ldots A_n$. Every attribute set $L \subseteq H$ determines a partition $\pi_L = \{D_1, \ldots, D_{a_L}\}$ of the set R of rows in table T, where $\mathtt{aDom}_T(L) = \{v_1, \ldots, v_{a_L}\}$ and $D_i = \{r \in R \mid T(r, L) = v_i\}$ for $1 \leq i \leq a_L$. Clearly, if $L = \{A_{i_1}, \ldots, A_{i_\ell}\}$, then $\pi_L = \bigcap\{\pi_{A_{i_j}} \mid 1 \leq j \leq \ell\}$. When $L = H$, we denote by $\pi_H = \bigcap\{\pi_{A_i} \mid 1 \leq i \leq n\}$ the partition determined by the intersection of all n attribute partitions. We call this partition the *intersection* partition.

Let $\mathbb{R}$ be the set of reals. The k-dimensional simplex is the set $\mathsf{SIMPLEX}_{k-1} = \{(p_1, \ldots, p_k) \in \mathbb{R}^k \mid p_i \geq 0 \text{ and } p_1 + \cdots + p_k = 1\}$.

A function $f : \mathbb{R} \longrightarrow \mathbb{R}$ is *concave on a set* $S \subseteq \mathbb{R}$ if $f(\alpha x + (1-\alpha)y) \geq \alpha f(x) + (1-\alpha) f(y)$ for $\alpha \in [0,1]$ and $x, y \in S$. The function f is *sub-additive* (*supra-additive*) on S if $f(x+y) \leq f(x) + f(y)$ ($f(x+y) \geq f(x) + f(y)$) for $x, y \in S$.

Definition 1 *A* generator *is a concave, subadditive function* $f : [0,1] \longrightarrow \mathbb{R}$ *such that* $f(0) = f(1) = 0$ *and* $f(\theta a_1) + \cdots + f(\theta a_n) \leq \theta(f(a_1) + \cdots + f(a_n)) + f(\theta)$ *for every* $(a_1, \ldots, a_n) \in \mathit{SIMPLEX}_{n-1}$ *and* $\theta \in [0,1]$.

The impurity measure *induced by* f *is* $i(p_1, \ldots, p_k) = f(p_1) + \cdots + f(p_k)$*, for every* $(p_1, \ldots, p_k) \in \mathit{SIMPLEX}_{k-1}$. $\square$

It is easy to verify that such functions as $f_{\mathtt{gini}}(p) = p - p^2$ (the Gini index), $f_{\mathtt{ent}}(p) = -p \log p$ (the Shannon entropy), or $f_{\mathtt{peak}}$ (given by $f_{\mathtt{peak}}(p) = p$ for $0 \leq p \leq 0.5$ and $f_{\mathtt{peak}}(p) = 1 - p$ for $0.5 < p \leq 1$) are generators.

Definition 2 *Let* f *be a generator,* R *be the set of rows of a table* T*, and let* $\pi = \{B_1, \ldots, B_n\}$, $\sigma = \{C_1, \ldots, C_m\}$ *be two partitions of* R.

The f-impurity of partition π *(called also the* f-entropy *of* π*) is* $\mathcal{H}^f(\pi) = \sum_{i=1}^n f\left(\frac{|B_i|}{|R|}\right)$.

The specific f-impurity of a subset L of R relative to a partition π *is the quantity*

$$\mathit{imp}_\pi^f(L) = f\left(\frac{|L \cap B_1|}{|L|}\right) + \cdots + f\left(\frac{|L \cap B_n|}{|L|}\right).$$

The f-impurity of π relative to σ *is*

$$\mathcal{H}^f(\pi|\sigma) = \sum_{j=1}^m \frac{|C_j|}{|R|} \mathit{imp}_\pi^f(C_j) = \frac{1}{|R|} \sum_{j=1}^m |C_j| \sum_{i=1}^n f\left(\frac{|B_i \cap C_j|}{|C_j|}\right).$$

This quantity is also called the f-conditional entropy of π relative to σ. $\square$

In other words, the conditional entropy $\mathcal{H}^f(\pi|\sigma)$ is the average value of the specific impurity of the classes of the partition σ relative to the partition π. Note that $\mathcal{H}^f(\pi|\omega_R) = \mathcal{H}^f(\pi)$ for every $\pi \in \mathsf{PART}(R)$. We have $\sigma \leq \pi$, if and

only if $\mathcal{H}^f(\pi|\sigma) = 0$. Thus, for any partition of the set R we have $\mathcal{H}^f(\pi|\alpha_R) = 0$ and $\mathcal{H}^f(\omega_R|\pi) = 0$.
Several properties of entropy (see [SCC00b]) are given next.

(i) Let f be a generator and let π, σ be two partitions of a set R. We have: $\mathcal{H}^f(\pi) \geq \mathcal{H}^f(\pi|\sigma) \geq \mathcal{H}^f(\pi \wedge \sigma) - \mathcal{H}^f(\sigma)$.
(ii) If $\pi_1, \pi_2, \sigma \in \mathsf{PART}(M)$ are such that $\pi_1 \leq \pi_2$, then $\mathcal{H}^f(\pi_1|\sigma) \geq \mathcal{H}^f(\pi_2|\sigma)$.
(iii) If $\pi, \sigma_1, \sigma_2 \in \mathsf{PART}(R)$ are such that $\sigma_1 \leq \sigma_2$, then $\mathcal{H}^f(\pi|\sigma_1) \leq \mathcal{H}^f(\pi|\sigma_2)$.
(iv) If $\pi_1, \pi_2, \sigma \in \mathsf{PART}(M)$ are such that $\pi_1 \leq \pi_2$, then $\mathcal{H}^f(\pi_1) \geq \mathcal{H}^f(\pi_2)$.
(v) Let f be a generator and let π, σ be two partitions of a set R. We have

$$\frac{1}{2}\left[\mathcal{H}^f(\pi) + \mathcal{H}^f(\sigma)\right] \leq \mathcal{H}^f(\pi \wedge \sigma) \leq \mathcal{H}^f(\pi) + \mathcal{H}^f(\sigma).$$

(vi) Let f be a generator and let π, σ be two partitions of a set R. We have

$$\frac{1}{2}\mathcal{H}^f(\pi) - \frac{1}{2}\mathcal{H}^f(\sigma) \leq \mathcal{H}^f(\pi|\sigma) \leq \mathcal{H}^f(\pi)$$

and

$$\frac{1}{2}\mathcal{H}^f(\sigma) - \frac{1}{2}\mathcal{H}^f(\pi) \leq \mathcal{H}^f(\sigma|\pi) \leq \mathcal{H}^f(\sigma).$$

(vii) Let f be a generator and let $\pi_1, \pi_2, \ldots \pi_n$ partitions of a set R. We have

$$\frac{1}{n}\sum_{i=1}^{n}\mathcal{H}^f(\pi_i) \leq \mathcal{H}^f(\pi_1 \wedge \pi_2 \ldots \wedge \pi_n) \leq \sum_{i=1}^{n}\mathcal{H}^f(\pi_i).$$

2.2 Properties of Partitions with k classes

Let $\pi_k = \{B_1, B_2 \ldots B_k\} \in PART(R)$ with $|\pi_k| = k$ classes and let π_{k-1} be the partition with $k-1$ classes, obtained from π_k by fusing the classes B_i and B_j, and keeping the rest of the classes unmodified. The difference between the entropies of the two partitions is given by $\mathcal{H}^f(\pi_k) - \mathcal{H}^f(\pi_{k-1}) = f\left(\frac{|B_i|}{|R|}\right) + f\left(\frac{|B_j|}{|R|}\right) - f\left(\frac{|B_i \cap B_j|}{|R|}\right) \geq 0$ (since the generator f is sub-additive). Thus, by joining two classes of a partition we obtained another partition with a smaller value of its entropy. Similarly, by splitting a class of a partition we obtained another partition with a greater value of the entropy. The quantity $\mathcal{H}^{f_{gini}}(\pi_k) - \mathcal{H}^{f_{gini}}(\pi_{k-1}) = 2\frac{|B_i||B_j|}{|R|^2}$. Thus for the Gini index, the maximum decrease in the entropy is obtained when the largest classes of the partition π_k are united. For the Shannon entropy, the decrease in entropy is $\mathcal{H}^{f_{ent}}(\pi_k) - \mathcal{H}^{f_{ent}}(\pi_{k-1}) = \frac{1}{|R|}(1+\frac{1}{r})^{r|B_i|}$, where $r = \frac{|B_j|}{|B_i|}$. If we choose B_i as the largest class of π_k, then the maximum value of the expression $(1+\frac{1}{r})^r$ is obtained for the largest possible value of r, thus for the largest value of $|B_j|$. Again, to obtain the greatest decrease in entropy we have to fuse the largest classes of the partition π_k.

Given a set R, α_R has the largest value of the entropy

$$\mathcal{H}^f(\alpha_R) = |R| f\left(\frac{1}{|R|}\right)$$

and the largest number of classes $|\alpha_R| = |R|$ and ω_R has the smaller value of the entropy $\mathcal{H}^f(\alpha_R) = 0$ and the smallest number of classes $|\omega_R| = 1$. We can obtain any partition π with k classes, by starting from α_R and successively fusing two classes. We obtain in this way a series of partitions $\pi^0 = \alpha_R, \pi^1, \pi^2 \dots \pi^k = \pi$ such that each π^i has the same classes as π^{i-1} with the exception of two classes which are united in π^i. If we want the partition π to have the smallest possible value of the entropy among all partitions with k classes, in the sequence of partitions obtained above, we must always to unite the largest possible classes, to ensure at each step the largest decrease in the value of the entropy. Thus, in this process, we always add one more element to the largest class obtained so far. Therefore, the partition π having $|\pi| = k$ classes and the smallest possible value of $\mathcal{H}^f(\pi)$ has $k-1$ classes of cardinality 1 and one class of cardinality $|R| - k + 1$.

The partition π having $|\pi| = k$ classes and the largest possible value of the entropy, has classes of equal cardinality $\frac{|R|}{k}$.

We conclude that the entropy of a partition π such that $|\pi| = k$ is bounded by

$$kf\left(\frac{1}{|R|}\right) + f\left(\frac{|R-k+1|}{|R|}\right) \leq \mathcal{H}^f(\pi) \leq kf\left(\frac{1}{k}\right).$$

Given an entropy value $\mathcal{H}^f_{\text{target}} \leq |R| f(\frac{1}{|R|})$, we can estimate an upper and lower bound for the number of classes $k = |\pi|$ of a partition π which has entropy $\mathcal{H}^f(\pi) = \mathcal{H}^f_{\text{target}}$.

For the Gini index as generator, the inequality $\mathcal{H}^f_{\text{target}} = \mathcal{H}^f(\pi) \leq kf\left(\frac{1}{k}\right)$ implies

$$k \geq \left\lceil \frac{1}{1 - \mathcal{H}^f_{\text{target}}} \right\rceil,$$

and the inequality $\mathcal{H}^f_{\text{target}} = \mathcal{H}^f(\pi) \geq kf\left(\frac{1}{|R|}\right) + f\left(\frac{|R-k+1|}{|R|}\right)$, yields

$$k \leq \left\lceil |R| \sqrt{\mathcal{H}^f_{\text{target}}} + 1 \right\rceil.$$

For the Shannon entropy as generator, we obtain the boundaries $2^{\mathcal{H}^f_{\text{target}}} \leq k \leq |R| + 1 - \lceil e^{\mathcal{H}^f_{\text{target}}} \rceil$, where k_0 is the first integer that makes the inequality $\frac{|R|}{e^{\mathcal{H}^f_{\text{target}}}} \leq (|R| - k + 1)^{(|R|-k+1)}$ true.

For a generator f, the upper bound of the value k, obtained from the inequality $\mathcal{H}^f_{\text{target}} \leq kf\left(\frac{1}{k}\right)$ represents the number of classes of a partition π with equal cardinality classes having the value of the entropy $\mathcal{H}^f(\pi) = \mathcal{H}^f_{\text{target}}$. The lower bound represents the number of classes of a partition with one class containing the majority of the elements. Since the last situation will be very unlikely to occur in practical clustering problems, more attention should be given to the upper bound estimate.

2.3 Dissimilarity measure

A generalization of the notion of metric on a set is the notion of *dissimilarity* (see[KR90] and [JMF99]) which is useful in clustering.

A *dissimilarity on a set* S is a function $d : S \times S \longrightarrow \mathbb{R}$ such that $d(x, y) \geq 0$, $d(x, x) = 0$ and $d(x, y) = d(y, x)$ for every $x, y \in S$.

The dissimilarity d is said to be *definite* if for all $x, y \in S$, $d(x, y) = 0$ implies $x = y$. If a dissimilarity satisfies the triangular inequality $d(x, y) + d(y, z) \geq d(x, z)$ for $x, y, z \in S$, then we obtain the familiar notion of metric on S.

If f is a generator, then the mapping $d^f : \mathsf{PART}(R) \times \mathsf{PART}(R) \longrightarrow \mathbb{R}$ given by $d^f(\pi, \sigma) = \mathcal{H}^f(\pi|\sigma) + \mathcal{H}^f(\sigma|\pi)$ for $\sigma, \pi \in \mathsf{PART}(R)$, is clearly a definite dissimilarity on $\mathsf{PART}(R)$. When π is close to σ, meaning that their classes have many elements in common, then both $\mathcal{H}^f(\pi|\sigma)$ and $\mathcal{H}^f(\sigma|\pi)$ are close to 0, so $d^f(\pi, \sigma)$ is close to 0.

3 Clustering using Genetic Algorithms

3.1 Genetic Algorithm

Let $T : R \times H \longrightarrow \bigcup D_H$ be a table. We are searching for the median partition, that is a partition π such that $|\pi| \leq k$, and π minimizes the sum $\sum_{A \in H} d^f(\pi_A, \pi)$.

A *k-chromosome* on a table $T : R \times H \longrightarrow \bigcup D_H$ is a function $K : R \longrightarrow \{1, \ldots, k\}$. An element of the set $\{1, \ldots, k\}$ is called a *class identifier.*

The partition π_K of the set of rows R determined by the k-chromosome K is $\pi_K = \{C_1, \ldots, C_k\}$, where $C_j = \{r \in R \mid K(r) = j\}$ for $1 \leq j \leq k$.

The input parameters of the genetic algorithm are summarized in Figure 1.

M	cardinality of the chromosomial population
N	number of rows in the table T
n	number of columns in the table T
k	number of classes in the median partition
r	percent of chromosomes that are used for crossover
m	percent of chromosomes that undergo a mutation
N_{max}	maximum number of consecutive iterations without improvement
ϵ	margin error for the fitness function

Figure 1: The input parameters of the genetic algorithm

The chromosomial population consists of k-chromosomes, $K_1, K_2 \ldots K_M$, where each k-chromosome K_i can be regarded as a sequence of length $N = |R|$ representing a possible assignment of the rows of the table T to the k classes of the partition π_{K_i}. Initially, the chromosomes $K_1, K_2 \ldots K_M$ are generated using random values between 1 and k.

The idea of the genetic evolution is to modify the chromosomes in the current population by using mutation and crossover as genetic operators such that in the new population we have chromosomes that will be increasingly closer to the median partition, that is, they will summarize better and better the columns of the table T.

We use the classical single point crossover operator. Starting from two chromosomes, a random crossing point (called a *crossover site*) is selected as a number l between 1 and N. The offspring will contain the first 1 to l positions from the first parent and the last $l+1$ to N positions from the second parent and vice versa. We use the classical mutation operator which involves changing randomly a number of $\max\{1, 0.1N\}$ positions in the chromosomes. The new value for each chromosome position is chosen randomly from 1 to k.

The pseudocode of the algorithm is the following:

```
initialize the population of genetic algorithm
while (true)
      compute the fitness of chromosomes in the population;
      if (there has been no relative
        improvement in best fitness value for $N_{max}$ iterations)
        then
            output the partition of $K_{\text{best}}$;
            exit;
      copy fittest $(1 - r - m)M$ chromosomes to new population;
      select probabilistically $\max\{2, rM\}$ chromosomes to cross over;
      apply crossover operator to the selected chromosomes
        and copy the offspring to the new population;
      select with uniform probability $\max\{1, mM\}$ chromosomes to mutate;
      apply mutation operator to the selected chromosomes
        and copy the modified chromosomes to the new population;
      replace old population by the new one;
```

For each chromosome K_i we compute the value fitness(K_i). We denote by K_{best} the chromosome which has the largest fitness(K_i) value; its fitness is denoted by fitness$_{\text{best}}$.

If in N_{max} consecutive iterations there is no improvement in the best fitness, then the algorithm will halt and the median partition will correspond to the chromosome K_{best}. Otherwise, a new population is computed using the elite selection (see [Mic99]). The fittest $\max\{M - 3, (1 - r - m)M\}$ chromosomes are copied directly in the new population, ensuring that we will never lose the best chromosome from the old population.

Next, a number of $\max\{2, rM\}$ chromosomes from the old generation are selected probabilistically to be used in the generation of the new offspring by crossover. The selection method used is a roulette wheel strategy (or fitness-proportionate selection) in which the chromosomes having greater values of their fitness have also a greater chance of being selected. The newly generated chromosomes are copied into the new population.

Finally, a number of $\max\{1, mM\}$ chromosomes are selected with uniform probability and are subjected to mutations. The mutation operator is not biased towards the fittest chromosomes and therefore the chromosomes that will suffer a mutation are selected with uniform probability.

3.2 Genetic Algorithm Fitness Measures

Let $\pi \in PART(R)$, $\mathcal{K}^f(\pi) = \sum_{A \in H} \mathcal{H}^f(\pi_A|\pi)$ and $\mathcal{L}^f(\pi) = \sum_{A \in H} \mathcal{H}^f(\pi|\pi_A)$. In the median partition problem, we are searching for a partition π which minimizes the distance $d^f(\pi) = \sum_A \left[\mathcal{H}^f(\pi_A|\pi) + \mathcal{H}^f(\pi|\pi_A)\right] = \mathcal{K}^f(\pi) + \mathcal{L}^f(\pi)$.

We have $d^f(\omega_R) = \mathcal{K}^f(\omega_R)$, since $\pi_A \leq \omega_R$ and $\mathcal{H}^f(\omega_R|\pi_A) = 0$ for all partitions π_A. Also $\mathcal{K}^f(\omega_R) = \sum_A \mathcal{H}^f(\pi_A)$, so $d^f(\omega_R) = \sum_A \mathcal{H}^f(\pi_A)$.

Similarly, we have $d^f(\alpha_R) = \mathcal{L}^f(\alpha_R)$, since $\pi_A \geq \alpha_R$ and $\mathcal{H}^f(\pi_A|\alpha_R) = 0$ for all partitions π_A. Given that $\pi_A = \{B_1^A, \ldots B_l^A\}$ and $\alpha_R = \{D_1, \ldots D_{|R|}\}$ we have $\mathcal{H}^f(\alpha_R|\pi_A) = \sum_{i=1}^{l} \frac{|B_i^A|}{|R|} \sum_{j=1}^{|R|} f\left(\frac{|B_i^A \cap D_j|}{|B_i|}\right)$. There are $|B_i^A|$ non-empty intersections in the inner sum, so

$$\mathcal{H}^f(\alpha_R|\pi_A) = \sum_{i=1}^{l} \frac{|B_i^A|^2}{|R|} f\left(\frac{1}{|B_i|}\right).$$

Thus, $d^f(\alpha_R) = \sum_{i=1}^{l} \frac{|B_i^A|^2}{|R|} f\left(\frac{1}{|B_i|}\right)$.

$\alpha_R \leq \pi \leq \omega_R$, $\mathcal{H}^f(\pi_A|\pi) \leq \mathcal{H}^f(\pi_A|\omega_R)$ and $\mathcal{H}^f(\pi|\pi_A) \leq \mathcal{H}^f(\alpha_R|\pi_A)$. Thus, we have an upper bound for the value $d^f(\pi)$, namely $d^f(\pi) \leq d^f(\omega_R) + d^f(\alpha_R)$.

We propose the quantities summarized in Figure 3.2 as measures of the closeness between the partitions represented by the chromosomes and the attribute partitions.

$$\boxed{\begin{array}{l}
\mathcal{K}^f(\pi) = \sum_{A \in H} \mathcal{H}^f(\pi_A|\pi) \\
\mathcal{M}^f(\pi) = \mathcal{K}^f(\pi) + \mathcal{L}^f(\pi) \\
\mathcal{N}^f(\pi) = \sum_{A \in H} \frac{\mathcal{H}^f(\pi|\pi_A)}{1+\mathcal{H}^f(\pi_A)} + \sum_{A \in H} \frac{\mathcal{H}^f(\pi|\pi_A)}{1+\mathcal{H}^f(\pi)} \\
\mathcal{O}^f(\pi) = \begin{cases} \mathcal{K}^f(\pi) \text{ if } \mathcal{H}^f(\pi) \leq \mathcal{H}^f_{\text{avg}} \\ \mathcal{L}^f(\pi) \text{ otherwise} \end{cases} \\
\mathcal{P}^f(\pi) = \begin{cases} \mathcal{K}^f(\pi) \text{ if } \mathcal{K}^f(\pi) \geq \mathcal{L}^f(\pi) \\ \mathcal{L}^f(\pi) \text{ otherwise} \end{cases} \\
\mathcal{Q}^f(\pi) = |\mathcal{K}^f(\pi) + \mathcal{L}^f(\pi)| + |\mathcal{K}^f(\pi) - \mathcal{L}^f(\pi)|
\end{array}}$$

Figure 2: Measures for assessing the quality of the partition π

Based on these measures, we define the fitness measure associated with each chromosome K whose partition is π_K, as in Figure 3.2.

Note that if a chromosome has a small value for $\mathcal{K}^f(\pi_{K_i})$, then its associated fitness value $\text{fitness}^f_{\mathcal{K}}(K_i)$ will be large. This property holds for all proposed fitness measures.

The measure $\mathcal{K}^f(\pi)$ leads the convergence process toward a partition π whose classes fit as well as possible inside the classes of all attribute partitions.

The measure $\mathcal{M}^f(\pi)$ requires the partition π to have classes that fit as well as possible inside the classes of all attribute partitions and conversely the classes

$\text{fitness}^f_{\mathcal{K}}(K) = \frac{\max\{\mathcal{K}^f(\pi_{K_i}) \mid 1 \leq i \leq n\}}{\mathcal{K}^f(\pi_K)}$	$\text{fitness}^f_{\mathcal{M}}(K) = \frac{\max\{\mathcal{M}^f(\pi_{K_i}) \mid 1 \leq i \leq n\}}{\mathcal{M}^f(\pi_K)}$
$\text{fitness}^f_{\mathcal{N}}(K) = \frac{\max\{\mathcal{N}^f(\pi_{K_i}) \mid 1 \leq i \leq n\}}{\mathcal{N}^f(\pi_K)}$	$\text{fitness}^f_{\mathcal{O}}(K) = \frac{\max\{\mathcal{O}^f(\pi_{K_i}) \mid 1 \leq i \leq n\}}{\mathcal{O}^f(\pi_K)}$
$\text{fitness}^f_{\mathcal{P}}(K) = \frac{\max\{\mathcal{P}^f(\pi_{K_i}) \mid 1 \leq i \leq n\}}{\mathcal{P}^f(\pi_K)}$	$\text{fitness}^f_{\mathcal{Q}}(K) = \frac{\max\{\mathcal{Q}^f(\pi_{K_i}) \mid 1 \leq i \leq n\}}{\mathcal{Q}^f(\pi_K)}$

Figure 3: Fitness Measures

of the attribute partitions should fit as well as possible inside the classes of the partition π.

For a given partition π and a given database, $\mathcal{K}^f(\pi)$ and $\mathcal{L}^f(\pi)$ do not have the same range of possible values, but $\frac{\mathcal{H}^f(\pi_A|\pi)}{1+\mathcal{H}^f(\pi_A)}$ and $\frac{\mathcal{H}^f(\pi|\pi_A)}{1+\mathcal{H}^f(\pi)}$ both range between 0 and 1. The measure $\mathcal{N}^f(\pi)$ uses the scaled versions of the conditional entropies.

The introduction of the measure $\mathcal{O}^f(\pi)$ is explained by the following results. For any partition π by summing over all partitions π_A we get the following inequalities $\frac{1}{2}\sum_A \mathcal{H}^f(\pi_A) - \frac{1}{2}n\mathcal{H}^f(\pi) \leq \mathcal{K}^f(\pi) \leq \sum_A \mathcal{H}^f(\pi_A)$ and $\frac{1}{2}n\mathcal{H}^f(\pi) - \frac{1}{2}\sum_A \mathcal{H}^f(\pi_A) \leq \mathcal{L}^f(\pi) \leq n\mathcal{H}^f(\pi)$. Let $\mathcal{H}^f_{\text{avg}}$ be the average entropy of the partitions generated by the attributes of the table, given by $\mathcal{H}^f_{\text{avg}} = \frac{\sum_A \mathcal{H}^f(\pi_A)}{n}$. The previous inequalities can be rewritten as $\mathcal{H}^f_{\text{avg}} - \mathcal{H}^f(\pi) \leq \frac{2\mathcal{K}^f(\pi)}{n} \leq 2\mathcal{H}^f_{\text{avg}}$ and $\mathcal{H}^f(\pi) - \mathcal{H}^f_{\text{avg}} \leq \frac{2\mathcal{L}^f(\pi)}{n} \leq 2\mathcal{H}^f(\pi)$. We have two cases depending on the relative values of $\mathcal{H}^f(\pi)$ and $\mathcal{H}^f_{\text{avg}}$:

1. If $\mathcal{H}^f_{\text{avg}} \geq \mathcal{H}^f(\pi)$, then we have $\max\left\{\frac{\mathcal{L}^f(\pi)}{n}, \mathcal{H}^f_{\text{avg}} - \frac{2\mathcal{K}^f(\pi)}{n}\right\} \leq \mathcal{H}^f(\pi) \leq \mathcal{H}^f_{\text{avg}}$, $\mathcal{K}^f(\pi) \leq n\mathcal{H}^f_{\text{avg}}$, and $\mathcal{L}^f(\pi) \leq n\mathcal{H}^f(\pi) \leq n\mathcal{H}^f_{\text{avg}}$.
   To have the blocks of the partition π fit as well as possible inside the blocks of the partitions π_A, we seek to minimize the average f-impurity of the blocks of π, relative to each of these partitions, that is, we seek to minimize $\mathcal{K}^f(\pi)$. The inequality $0 \leq \mathcal{H}^f_{\text{avg}} - \mathcal{H}^f(\pi) \leq \frac{2\mathcal{K}^f(\pi)}{n}$, guarantees that if $\mathcal{K}^f(\pi)$ is small, then $\mathcal{H}^f(\pi)$ is close to $\mathcal{H}^f_{\text{avg}}$.
   For the quantity $\mathcal{M}^f(\pi) = \mathcal{K}^f(\pi) + \mathcal{L}^f(\pi)$ we have also the following lower and upper bound $\frac{n}{2}\{\ \mathcal{H}^f_{\text{avg}} - \mathcal{K}^f(\pi)\ \} \leq \mathcal{K}^f(\pi) + \mathcal{L}^f(\pi) \leq 2n\mathcal{H}^f_{\text{avg}}$.
2. If $\mathcal{H}^f_{\text{avg}} < \mathcal{H}^f(\pi)$, then we have $\max\left\{\frac{\mathcal{L}^f(\pi)}{n}, \mathcal{H}^f_{\text{avg}}\right\} \leq \mathcal{H}^f(\pi) \leq \mathcal{H}^f_{\text{avg}} + \frac{2\mathcal{L}^f(\pi)}{n}$, $\mathcal{K}^f(\pi) \leq n\mathcal{H}^f_{\text{avg}}$, and $\mathcal{L}^f(\pi) \leq n\mathcal{H}^f(\pi)$.
   To have the blocks of the attribute partitions π_A, fit as well as possible inside the blocks of the partition π found by the genetic algorithm, we seek to minimize the average f-impurity of the blocks of attribute partitions π_A, relative to the partition obtained by the genetic algorithm, that is, we seek to minimize $\mathcal{L}^f(\pi)$. The inequality $0 \leq \mathcal{H}^f(\pi) - \mathcal{H}^f_{\text{avg}} \leq \frac{2\mathcal{L}^f(\pi)}{n}$, insures that the value of $\mathcal{H}^f(\pi)$ will get close to the value $\mathcal{H}^f_{\text{avg}}$.

In this case the lower and upper bound of the quantity $\mathcal{M}^f(\pi) = \mathcal{K}^f(\pi) + \mathcal{L}^f(\pi)$ are $\frac{n}{2}\{\ \mathcal{H}^f(\pi) - \mathcal{H}^f_{\text{avg}}\ \} \leq \mathcal{K}^f(\pi) + \mathcal{L}^f(\pi) \leq n\mathcal{H}^f_{\text{avg}} + n\mathcal{H}^f(\pi)$.

To conclude, by minimizing $\mathcal{K}^f(\pi)$ and $\mathcal{L}^f(\pi)$ we obtain a partition π which has the value $\mathcal{H}^f(\pi)$ as close as possible to the value $\mathcal{H}^f_{\text{avg}}$. This observation suggest the following convergence criteria for the genetic algorithm:

If $\mathcal{H}^f_{\text{avg}} \geq \mathcal{H}^f(\pi)$, the genetic algorithm will minimize $\mathcal{K}^f(\pi)$, favoring partitions with many blocks of smaller sizes. Otherwise ($\mathcal{H}^f_{\text{avg}} < \mathcal{H}^f(\pi)$), the genetic algorithm will minimize $\mathcal{L}^f(\pi)$, favoring partitions with fewer blocks of bigger sizes. This alternation between minimization of $\mathcal{K}^f(\pi)$ and $\mathcal{L}^f(\pi)$ corresponds to the measure $\mathcal{O}^f(\pi)$ from figure 3.2.

Similarly, the measure $\mathcal{P}^f(\pi)$ uses the idea of alternating between the minimization of $\mathcal{K}^f(\pi)$ and $\mathcal{L}^f(\pi)$, but the choice depends on the relative values of the two quantities. Namely, if $\mathcal{K}^f(\pi) < \mathcal{L}^f(\pi)$ the genetic algorithm will minimize $\mathcal{L}^f(\pi)$, otherwise will minimize $\mathcal{K}^f(\pi)$.

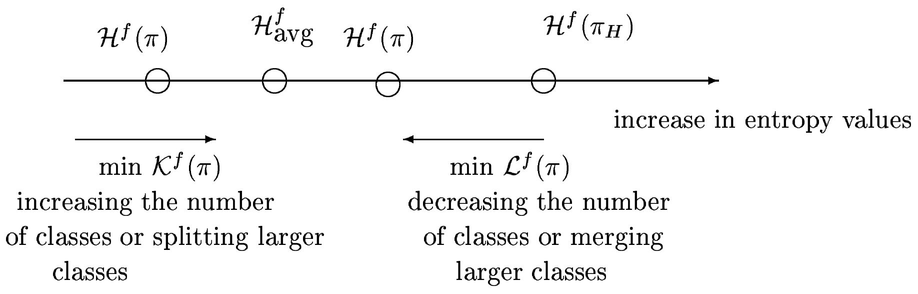
When the quantity $\mathcal{K}^f(\pi)$ is very small, the partition π has many classes with smaller sizes, that fit well inside the classes of attribute partitions and therefore the classes of attribute partitions might not fit that well inside the classes of π, leading to a greater value of the quantity $\mathcal{L}^f(\pi)$. Thus, we want to favor partitions for each the difference between the two quantities is small in absolute value. The measure $\mathcal{Q}^f(\pi)$ is based on this observation.

For any given database, we have $\mathcal{H}^f_{\text{avg}} \leq \mathcal{K}^f(\pi_H) \leq n\mathcal{H}^f_{\text{avg}}$. Thus, the average entropy of the attribute partitions has always a smaller value than the entropy of the *intersection* partition. We denote by $k_H = |\pi_H|$. The blocks of π_H fit perfectly inside the blocks of all attribute partitions π_A, so π_H is a candidate for the partition that we search. But, we are not interested in this special partition since it might have a large number of blocks (close to the total number of rows $|R|$) and thus, the corresponding clustering of the data is not interesting. We are interested to find a partition π that is as close as possible to all attribute partitions and has the maximum possible number of blocks k less than k_H. In this process of minimizing $\mathcal{K}^f(\pi)$ and $\mathcal{L}^f(\pi)$, the entropy $\mathcal{H}^f(\pi)$ of the partition π will get closer and closer to the value of the average entropy of the attribute partitions $\mathcal{H}^f_{\text{avg}}$.

The interpretation of the measure $\mathcal{O}^f(\pi)$ is illustrated in Figure 4.

If $\mathcal{H}^f(\pi) \geq \mathcal{H}^f_{\text{avg}}$, the partition π has too many blocks or very small blocks and the minimization of $\mathcal{L}^f(\pi)$ has the effect of decreasing the number of blocks or respectively uniting blocks of the partition π. Otherwise ($\mathcal{H}^f(\pi) \leq \mathcal{H}^f_{\text{avg}}$), π has too few blocks or blocks with large sizes and the minimization of $\mathcal{K}^f(\pi)$ has the effect of increasing the number of blocks or splitting the blocks of the partition π.

For databases for which the value $\mathcal{H}^f_{\text{avg}}$ represents a good approximation of the entropy of the partition that we are searching, then the measures $\mathcal{O}^f(\pi)$ or $\mathcal{P}^f(\pi)$ should be used in the convergence process of the genetic algorithm in preference to the other proposed measures. The partition π associated with the minimum possible value of the quantity $\mathcal{M}^f(\pi)$ has the entropy $\mathcal{H}^f(\pi)$ close to the value of the average entropy of all attribute partitions $\mathcal{H}^f_{\text{avg}}$. Thus, the



$\mathcal{H}^f(\pi)$
$\mathcal{H}^f_{\text{avg}}$
$\mathcal{H}^f(\pi)$
$\mathcal{H}^f(\pi_H)$
increase in entropy values
min $\mathcal{K}^f(\pi)$
increasing the number of classes or splitting larger classes
min $\mathcal{L}^f(\pi)$
decreasing the number of classes or merging larger classes

Figure 4: Explanation of the convergence process for measure $\mathcal{O}^f(\pi)$

number of blocks $k = |\pi|$ of the partition π can be estimated from the inequality $kf\left(\frac{1}{|R|}\right) + f\left(\frac{|R-k+1|}{|R|}\right) \leq \mathcal{H}^f_{\text{avg}} \leq kf\left(\frac{1}{k}\right)$ as explained in the previous section. For the Gini index generator we get the limits for the number of classes: $|R|\sqrt{\mathcal{H}^f_{\text{avg}}} + 1 \geq k \geq \frac{1}{1-\mathcal{H}^f_{\text{avg}}}$. If the partition π has well represented classes (with similar cardinalities) then the lower bound limit is a better estimation of k. On the other hand if π has almost all elements grouped into one class, then the upper bound limit is a better estimate. If we do not know in advance what to assume about the nature of the partition π that we searched, we have to apply the genetic clustering algorithm searching for successive values of k, starting from $\frac{1}{1-\mathcal{H}^f_{\text{avg}}}$ and increasing k by 1. The quantity $\mathcal{M}^f(\pi)$ will have a minimum for the value of k reflecting the natural number of classes. If the value $\mathcal{M}^f(\pi)$ always increases as we increment the number of classes, then the value $k = \frac{1}{1-\mathcal{H}^f_{\text{avg}}}$ represents the natural number of classes.

4 Experimental results

We studied the clustering found by three algorithms, our genetic algorithm denoted by `ALG-RAND`, the algorithm introduced by [Rég83a], denoted by `ALG-TP`, based on the idea of transferring elements from one class to another until no improvements has been done in a number of consecutive iterations, and the classical k-means algorithm denoted by `K-MEANS`.

To prove the better quality of the partition obtained by our method, we used synthetically generated databases for which we know in advance the partition embedded in the data, denoted by *reference* partition.

The generation of these databases follows the pattern: for each row number *rowid* $\in [1, N]$, a number $i \in [1, c]$ is randomly generated and saved at position *rowid* in the *reference* partition and in all attribute partitions, but one. The exception attribute $A_e \in H$, randomly chosen, receives at position *rowid* a different value $j \in [1, c], j \neq i$. To ensure that the *reference* and attribute partitions have exactly c classes, the first values for i are $1, 2, \ldots c$.

We generated a synthetic database with a fixed number of rows $N = 50$ and columns $n = 5$, and embedded a $k = 5$ class partition in the data. To see how the algorithm behave on average when the population of chromosomes increases relative to the size of the database, we ran the genetic algorithm with 10 different values for the random number generator seed and with the number of chromosomes equal to 1, 2, up to 10 times the number of rows. The other parameters for the genetic algorithm are: crossover rate of 0.8, mutation rate of 0.1, the Shannon entropy used as generator and 100 consecutive iterations without improvement. The results are shown in Figures 5, 6, and 7.

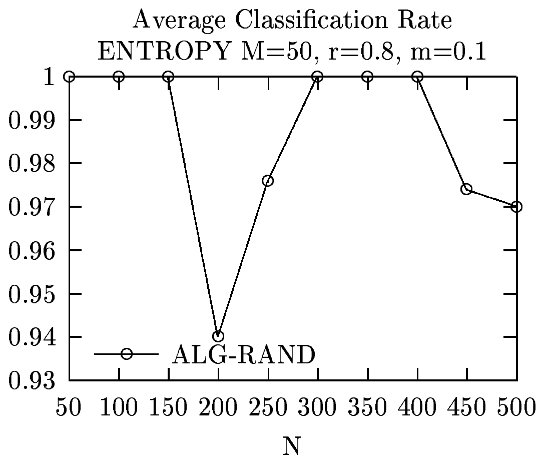


Figure 5: Classification rate / `ALG-RAND` with increasing number of chromosomes

`ALG-RAND` performed extremely well and found the *reference* partition or a close approximation of it, leading to classification rates (the number of rows classified in the partition found by the algorithm in the same way as in the *reference partition*) between 0.94 and 1. By increasing the size of the chromosomial population, the number of iterations required by the algorithm decreases significantly. The time required by the algorithm is linear in the number of chromosomes.

We fixed the number of chromosomes to $N = 50$ and we generated databases with $c = 5$ columns, $k = 5$ class partitions embedded in the data and a number of rows equal respectively to 1, 2 up to 10 times the number of chromosomes. The other parameters for the genetic algorithm remained as in the previous experiment. The results are shown in Figures 8, 9, and 10.

With the increase in the number of rows, the classification rate decreases since the search space becomes more complex, the number of iterations and the time increases. The increase in the number of iterations and the time is linear in the size of the database.

We generated a database with $N = 100$ rows, $n = 10$ columns, and $c = 5$ classes in the attribute and *reference* partitions. On this database, we ran our genetic algorithm `ALG-RAND`, the algorithm `ALG-TP` and the algorithm `K-MEANS` implemented in the WEKA package, searching for a partition with $k = 5$ classes.

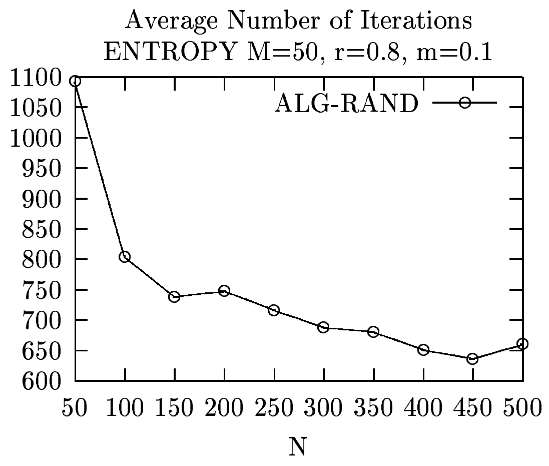


Figure 6: Number of iterations / `ALG-RAND` with increasing number of chromosomes

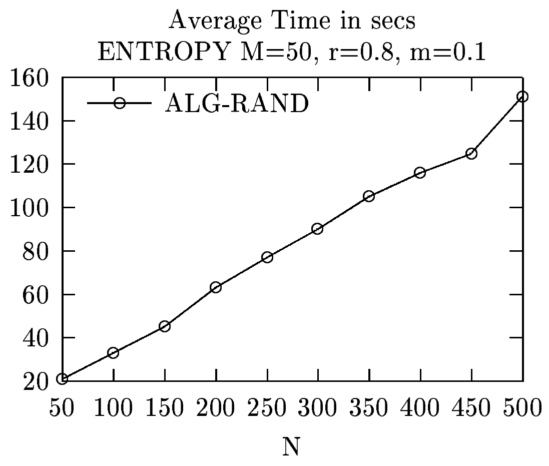


Figure 7: Time (secs) / `ALG-RAND` with increasing number of chromosomes

We executed 10 runs for each algorithm, corresponding to different values for the seed of the random number generator and we computed the average performance of the partitions found. For the genetic algorithm we used the fitness measures based on the quantities presented in Figure 3.2, 100 chromosomes, a crossover rate of 0.8, a mutation rate of 0.1, the Gini index and respectively the Shannon entropy as generators, and 100 consecutive iterations without improvement. The results are summarized in the Figure 11. The results table contains the entropy

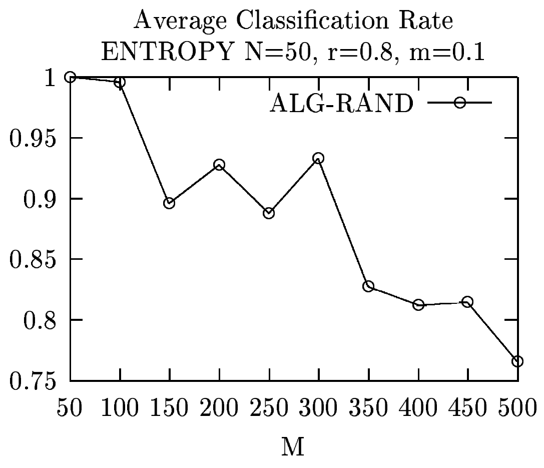


Figure 8: Classification rate / `ALG-RAND` with increasing number of chromosomes

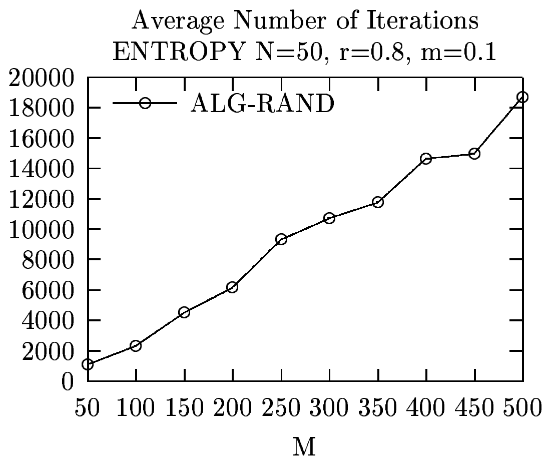


Figure 9: Number of iterations / `ALG-RAND` with increasing number of chromosomes

of the *intersection* partition $\mathcal{H}^f(\pi_H)$, the average value of the entropies of all attribute partitions $\mathcal{H}^f_{\mathrm{avg}}$, the entropy of the *reference* partition $\mathcal{H}^f(\pi_{\mathrm{ref}})$ and its associated value $\mathcal{M}^f(\pi_{\mathrm{ref}})$. For the genetic algorithms we present the average value of the following measures: $\mathcal{H}^f(\pi)$, $\mathcal{M}^f(\pi)$, $d(\pi, \pi_{\mathrm{ref}})$ (the distance between π and π_{ref}), the classification rate cr, and the number of iterations nitrs. For the `ALG-TP` and the `K-MEANS` algorithms we present only the average value of the classification rate.

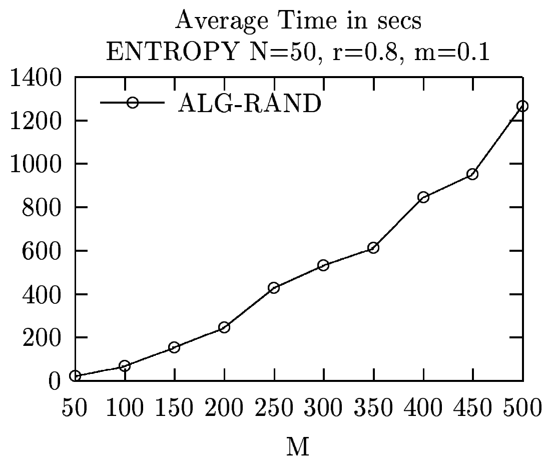


Figure 10: Time (secs) / `ALG-RAND` with increasing number of chromosomes

The genetic algorithms using fitness measures based on the quantities from Figure 3.2 lead in all cases to a partition close to the *reference* partition, in approximatively the same number of iterations. The genetic algorithm using the Gini index as generator obtained the best partition in the case of the measure $\mathcal{Q}^f(\pi)$, for which $\mathcal{M}^f(\pi) = 3.820$ is the closest value to $\mathcal{M}^f(\pi_{\text{ref}}) = 3.561$, and $\mathcal{H}^{f_{gini}}(\pi) = 0.793$ is the closest value to $\mathcal{H}^{f_{gini}}_{\text{avg}} = 0.793$. For this quality measure the average classification rate was 0.99, leading to a very good approximation of the searched partition, in which 99 elements have been classified in average as in the *reference* partition. When the Shannon entropy was used as generator, the best partition was found by using the measure $\mathcal{K}^f(\pi)$; in this case also the associated value $\mathcal{M}^f(\pi) = 11.102$ is the closest value to $\mathcal{M}^f(\pi_{\text{ref}}) = 10.480$, and $\mathcal{H}^{f_{entr}}(\pi) = 2.300$ is the closest value to $\mathcal{H}^{f_{entr}}_{\text{avg}} = 2.298$. For the measure $\mathcal{K}^f(\pi)$, the genetic algorithm has also a high value of the average classification rate equal to 0.992, meaning that 99 elements have been grouped as in the *reference* partition.

We are searching for a partition π having well represented classes, that is we do not have any reason to assume that some classes have significantly more elements than the others, and we can estimate the number of classes k of this partition using the value of the average entropy of the attribute partitions. When the Gini index is used as generator, the value of k is given by $\lceil \frac{1}{1-\mathcal{H}^f_{\text{avg}}} \rceil = \lceil 4.85 \rceil = 5$, and when the Shannon entropy is used, we have $k = \lceil 2^{\mathcal{H}^f_{\text{avg}}} \rceil = \lceil 2^{2.3} \rceil = 5$. This estimation corresponds to the real number of classes of the *reference* partition.

The algorithm `ALG-TP` converged toward a partition that is far from the *reference* partition, having in average a classification rate of 0.616. `ALG-TP` favors partitions with fewer than k classes and thus, with larger class cardinalities.

ALG-RAND with Gini index					
$\mathcal{H}^{f_{gini}}(\pi_H) = 0.99$			$\mathcal{H}^{f_{gini}}_{\text{avg}} = 0.793$		
$\mathcal{H}^{f_{gini}}(\pi_{\text{ref}}) = 0.793$			$\mathcal{M}^f(\pi_{\text{ref}}) = 3.561$		
fitness$^f_{\mathcal{K}}$	$\mathcal{H}^{f_{gini}}(\pi)$	$\mathcal{M}^f(\pi)$	$d(\pi, \pi_{\text{ref}})$	cr	nitrs
$\mathcal{K}^f(\pi)$	0.784	4.461	0.122	0.911	798
$\mathcal{M}^f(\pi)$	0.757	4.331	0.111	0.909	704
$\mathcal{N}^f(\pi)$	0.776	4.224	0.093	0.94	690
$\mathcal{O}^f(\pi)$	0.787	4.216	0.090	0.938	719
$\mathcal{P}^f(\pi)$	0.786	4.127	0.080	0.945	745
$\mathcal{Q}^f(\pi)$	0.793	3.820	0.036	0.99	706
ALG-RAND with Shannon entropy					
$\mathcal{H}^{f_{ent}}(\pi_H) = 6.643$			$\mathcal{H}^{f_{ent}}_{\text{avg}} = 2.300$		
$\mathcal{H}^{f_{ent}}(\pi_{\text{ref}}) = 2.298$			$\mathcal{M}^f(\pi_{\text{ref}}) = 10.480$		
fitness$^f_{\mathcal{K}}$	$\mathcal{H}^{f_{ent}}(\pi)$	$\mathcal{M}^f(\pi)$	$d(\pi, \pi_{\text{ref}})$	cr	nitrs
$\mathcal{K}^f(\pi)$	2.300	11.102	0.029	0.992	816
$\mathcal{M}^f(\pi)$	2.054	13.003	0.170	0.867	769
$\mathcal{N}^f(\pi)$	2.274	11.660	0.068	0.972	780
$\mathcal{O}^f(\pi)$	2.274	12.365	0.117	0.924	811
$\mathcal{P}^f(\pi)$	2.279	12.273	0.106	0.934	851
$\mathcal{Q}^f(\pi)$	2.273	12.337	0.120	0.905	960
ALG-TP Average Classification Rate 0.616					
K-MEANS Average Classification Rate 0.824					

Figure 11: `ALG-RAND`, `ALG-TP`, and `K-MEANS` on the first synthetically generated database

The `K-MEANS` algorithm from the WEKA package converged to better partitions than `ALG-TP`, having an average classification rate of 0.824, but still did not outperformed our genetic algorithms.

We can conclude that on this database, our genetic algorithms produce better clusterings than the classical "transfer points" and k-means algorithms.

A second type of synthetically generated databases are characterized by attribute partitions with a predominant class. The generation process differs from the one described previously, in the fact that after the first c positions have being filled with the sequence $1, 2, \ldots c$, the number i to be filled in the *reference* and all attribute partitions, is set to 1 for any two other rows. Thus, the *reference* and attribute partitions have class 1 with the largest cardinality value.

The parameters of the genetic algorithms and the setting of the experiments remain the same as in the previous ones. The results are summarized in Figure 12.

The partitions found by the genetic algorithms using both the Gini index and Shannon entropy as generators, are very close to the *reference* partition for the quality measures $\mathcal{M}^f(\pi)$ and $\mathcal{N}^f(\pi)$. The best partition is found in case of the Gini index for $\mathcal{M}^f(\pi)$, when $\mathcal{M}^f(\pi) = 4.380$ is the closest value to $\mathcal{M}^f(\pi_{\text{ref}}) = 3.353$, and in case of the Shannon entropy for $\mathcal{N}^f(\pi)$, when $\mathcal{M}^f(\pi) = 11.580$ is the closest value to $\mathcal{M}^f(\pi_{\text{ref}}) = 9.600$. The best classification rate for the genetic algorithms is in average 0.895 for the Gini index and 0.927

ALG-RAND with Gini index					
$\mathcal{H}^{f_{gini}}(\pi_H) = 0.99$			$\mathcal{H}^{f_{gini}}_{\text{avg}} = 0.660$		
$\mathcal{H}^{f_{gini}}(\pi_{\text{ref}}) = 0.623$			$\mathcal{M}^f(\pi_{\text{ref}}) = 3.353$		
$fitk$	$\mathcal{H}^{f_{gini}}(\pi)$	$\mathcal{M}^f(\pi)$	$d(\pi, \pi_{\text{ref}})$	cr	nitrs
$\mathcal{K}^f(\pi)$	0.767	6.921	0.460	0.61	792
$\mathcal{M}^f(\pi)$	0.614	4.380	0.141	0.895	721
$\mathcal{N}^f(\pi)$	0.599	4.478	0.159	0.876	789
$\mathcal{O}^f(\pi)$	0.660	6.986	0.486	0.722	727
$\mathcal{P}^f(\pi)$	0.650	4.861	0.198	0.874	835
$\mathcal{Q}^f(\pi)$	0.648	5.087	0.229	0.848	780
ALG-RAND with Shannon entropy					
$\mathcal{H}^{f_{ent}}(\pi_H) = 6.643$			$\mathcal{H}^{f_{ent}}_{\text{avg}} = 1.911$		
$\mathcal{H}^{f_{ent}}(\pi_{\text{ref}}) = 1.810$			$\mathcal{M}^f(\pi_{\text{ref}}) = 9.600$		
$fitk$	$\mathcal{H}^{f_{ent}}(\pi)$	$\mathcal{M}^f(\pi)$	$d(\pi, \pi_{\text{ref}})$	cr	nitrs
$\mathcal{K}^f(\pi)$	2.156	15.796	0.383	0.653	843
$\mathcal{M}^f(\pi)$	1.656	12.181	0.142	0.903	852
$\mathcal{N}^f(\pi)$	1.757	11.580	0.109	0.927	850
$\mathcal{O}^f(\pi)$	1.879	13.696	0.245	0.842	803
$\mathcal{P}^f(\pi)$	1.863	13.745	0.238	0.841	872
$\mathcal{Q}^f(\pi)$	1.896	15.285	0.340	0.775	813
ALG-TP Average Classification Rate 0.82					
K-MEANS Average Classification Rate 0.833					

Figure 12: `ALG-RAND`, `ALG-TP`, and `K-MEANS` on the second type of synthetically generated database

for the Shannon entropy.

For this database `ALG-TP` performed better than in the previous experiment, but still the average classification rate 0.82 is smaller than the performance of the genetic algorithm using $\mathcal{M}^f(\pi)$ and $\mathcal{N}^f(\pi)$ as quality measures. The better performance of `ALG-TP` for this database is explained by the tendency of the algorithm to converge toward partitions with fewer classes of larger cardinality.

Again the `K-MEANS` algorithm from the WEKA package has a better classification rate 0.833 than the one of `ALG-TP`, and is still not as good as the genetic algorithm using the measures $\mathcal{M}^f(\pi)$, and $\mathcal{N}^f(\pi)$.

Note that the quality measures $\mathcal{N}^f(\pi)$ and $\mathcal{P}^f(\pi)$ lead to good classification rates regardless of the type of database.

As we discussed in section 2.2, we can obtain an estimate of the natural number of classes in the input database, by using the value of $\mathcal{H}^f_{\text{avg}}$. In the second experiment we work with a database having $\mathcal{H}^{f_{entr}}_{\text{avg}} = 1.911$. For this value the lower bound estimate of the number of classes gives $k = \lceil 2^{1.911} \rceil = 4$. Using the same database we executed 10 runs of the algorithm `ALG-RAND`, searching for partitions with $k \in \{3, 4, 5, 6, 7, 8, 9, 10\}$ classes.

We observe from Figure 13 that the value of the quantity $\mathcal{M}^f(\pi)$ we search to minimize reaches a minimum for values of k around the real number of classes embedded in the data. A detailed description of the partitions found by `ALG-RAND`

ALG-RAND with $\mathcal{N}^f(\pi)$ and Shannon entropy					
$\mathcal{H}^{f_{ent}}(\pi_H) = 6.643$			$\mathcal{H}^{f_{ent}}_{\text{avg}} = 1.911$		
$\mathcal{H}^{f_{ent}}(\pi_{\text{ref}}) = 1.810$			$\mathcal{M}^f(\pi_{\text{ref}}) = 9.600$		
k	$\mathcal{H}^{f_{entr}}(\pi)$	$\mathcal{M}^f(\pi)$	$d(\pi, \pi_{\text{ref}})$	cr	nitrs
3	1.292	13.384	0.242	0.827	581
4	1.555	12.373	0.163	0.894	663
5	1.757	11.580	0.109	0.927	850
6	1.885	10.712	0.051	0.967	998
7	1.952	11.102	0.068	0.959	1178
8	2.051	11.632	0.093	0.946	1080
9	2.148	12.456	0.133	0.919	1349
10	2.295	13.615	0.192	0.876	1345

Figure 13: ALG-RAND with $\mathcal{N}^f(\pi)$, Shannon entropy and various k on the second type of synthetically generated database

in one of the 10 experiments is given in Figure 14.

π_{ref}				
k	class card	$\mathcal{H}^{f_{entr}}(\pi)$	$\mathcal{M}^f(\pi)$	cr
5	57 17 10 9 7	1.810	9.600	
ALG-RAND with $\mathcal{N}^f(\pi)$ and Shannon entropy				
k	class card	$\mathcal{H}^{f_{entr}}(\pi)$	$\mathcal{M}^f(\pi)$	cr
3	57 33 10	1.322	13.047	0.84
4	58 19 16 7	1.602	11.546	0.9
5	57 17 10 9 7	1.810	9.600	1
6	56 17 16 10 1	1.724	11.365	0.92
7	56 17 10 9 7 1	1.882	10.196	0.99
8	57 17 10 8 7 1	1.855	9.932	0.99
9	53 16 10 9 7 4 1	2.074	11.856	0.95
10	50 10 9 9 8 7 7	2.286	13.441	0.85

Figure 14: Characteristics of the partitions found by ALG-RAND with $\mathcal{N}^f(\pi)$ on the second type of synthetically generated database

The best partition found by ALG-RAND using the quality measure $\mathcal{N}^f(\pi)$ and the Shannon entropy is the one obtained for $k = 5$, characterized by the smallest value of $\mathcal{M}^f(\pi)$. In this case ALG-RAND obtained exactly the *reference* partition, as the classification rate of 1 indicates. Even if we started the algorithm by asking for more that the real number of classes, the partitions found by the algorithm have discovered the major classes of the *reference* partition and the additional classes have smaller cardinalities. This show that our algorithms are robust and they are able to discover the real clustering of the data even if they are asked for more than the necessary number of classes.

We conclude that whenever we do not know in advance the number of classes in the natural clustering of the input data, we have to run a couple of experiments with increasing values of $k = |\pi|$, and we have to consider both the shape of the partition π found by the algorithm and its associated value $\mathcal{M}^f(\pi)$. We can use as the best approximation of the median partition, the partition that has the smaller value of $\mathcal{M}^f(\pi)$ and the fewest number of small cardinality classes.

5 Conclusions

We demonstrated the use of the notion of generalized entropy for designing a genetic algorithm for finding clusters in a given data and experimented with different fitness measures for the evaluation of the chromosomial population. Our experiments showed that, with very few exceptions, the algorithms converged to the clustering that we embedded in the synthetically generated databases, and, when they didn't converge, then they found a clustering that was close to the "hidden" one.

The number of iterations required by our algorithms scales linearly with the size of the database.

Using the dissimilarity measure $\mathcal{K}^f(\pi) + \mathcal{L}^f(\pi)$, it is possible to determine the number of clusters embedded in the data in the absence of any apriori knowledge. As shown in this paper the information-theoretical measures proved successful in finding the natural clustering of the data.

References

[BL95] Jean-Pierre Barthélemy and Bruno Leclerc. The median procedure for partitions. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. AMS, 1995.

[Fal99] Emanuel Falkenauer. *Genetic algorithms and grouping problems*. Jon Wiley, New York, 1999.

[Hol92] John H. Holland. *Adaptation in Natural and Artificial Systems*. The MIT Press, Cambridge, Massachusetts, 1992.

[Hua97] Zhexue Huang. A fast clustering algorithm to cluster very large categorical data sets in data mining. In *Research Issues on Data Mining and Knowledge Discovery*, pages 0–, 1997.

[JB91] Donald R. Jones and Mark A. Beltramo. Solving partitioning problems with genetic algorithms. *Proceedings of the fourth International Conference on Genetic Algor hms*, pages 442–449, 1991.

[JMF99] A.K. Jian, M.N. Murty, and P.J. Flynn. Data clustering: A review. *ACM Computing Surveys, vol 31, no. 3*, September 1999.

[KR90] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data*. John Wiley, New York, 1990.

[Mic99] Zbigniew Michalewicz. *Genetic algorithms + Data Structures = Evolution Programs*. Springer, New York, 1999.

[Mit97] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[Rég83a] Régnier. Sur quelques aspects mathématiques des problèmes de classification automatique. *Math. Sci. Hum., no 82*, pages 13–29, 1983.

[Rég83b] Simon Régnier. Etudes sur le polyèdre des partitions. *Math. Sci. Hum., no 82*, pages 85–111, 1983.

[SCC00a] Dan Simovici, Dana Cristofor, and Laurentiu Cristofor. Generalized entropy and projection clustering of categorical data. Technical Report 00-3, University of Massachusetts, Boston, Massachusetts, 2000.

[SCC00b] Dan Simovici, Dana Cristofor, and Laurentiu Cristofor. Generalized entropy and projection clustering of categorical data. *Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 619–625, 2000.

[SCC00c] Dan Simovici, Dana Cristofor, and Laurentiu Cristofor. Mining for purity dependencies in databases. Technical Report 00-2, University of Massachusetts, Boston, Massachusetts, 2000.

[SRK99] Guha Sudipto, Rastogi Rajeev, and Shim Kyuseok. Rock: A robust clustering algorithm for categorical attributes. *Proceedings of the IEEE International Conference on Data Engineering, Sydney, March 1999*, March 1999.

[VJR99] Ganti Venkatesh, Gehrke Johannes, and Ramakrishnan Raghu. Cactus - clustering categorical data using summaries. *KDD-99 San Diego CA USA*, 1999.