Learning and Nonlinear Models (L&NLM) – Journal of the Brazilian Computational Intelligence Society, Vol. 13, Iss. 2, pp. 73-91, 2015

A HYBRID METHOD USING EVOLUTIONARY ALGORITHM AND A LINEAR INTEGER MODEL TO SOLVE THE AUTOMATIC CLUSTERING PROBLEM

Marcelo Dib Cruz[1]
Luiz Satoru Ochi[2]

[1]Departamento de Matemática – Universidade Federal Rural do Rio de Janeiro (UFRRJ)
[2] Instituto de Computação – Universidade Federal Fluminense (UFF)
Niteroi – RJ – Brasil
{madibcruz@gmail.com, satoru@ic.uff.br}

Abstract- Clustering is the process by which elements of a database are assigned for clusters of similar elements. In clustering algorithms, it is usually assumed that the number of clusters is known. Unfortunately, the optimal number of clusters is unknown for many applications. These problems are known as *Automatic Clustering Problems (ACP).* In this work, we propose a hybrid method that uses an Evolutionary Algorithm with Local Search and a linear and integer model to solve the ACP. Computational results on a set of instances illustrate the effectiveness and the robustness of the proposed method.

Keywords- Automatic Clustering, Evolutionary Algorithms, Linear and Integer Model, Hybrid Method.

1 Introduction

Clustering is a generic term for a process which joins similar objects in a cluster. When the number of clusters is known a priori, the problem is called K-Clustering Problem or Clustering Problem (CP). If not, it is called Automatic Clustering Problem (ACP). These problems are classified as NP-Hard (Welch, 1983), but the fact that the value of k is not known makes the problem much more complex, and increases substantially the number of possible solutions. The ACP is also known as either problem of finding the "true" number of clusters (Hardy, 1996), or problem of determination of the natural number of clusters (Everitt, 2001).

There are several applications related to clustering, including: Graphs Partitioning, Manufacturing Flexible Problem, Recognition of Pattern with Image Processing, Computational Biology, Market Research, Classification of Documents, Data Mining, among others.

The heuristics are powerful tools able to solve large problems that use any kind of functions, linear or nonlinear. The heuristics can solve combinatorial problems with any function in reasonable time.

The function used in this work, called Silhouette Index (Kaufman & Rousseeum, 1990), is a nonlinear function. We tried to model this function like Integer Programming, but it was not possible. We also tried to adapt the Integer Models to CP, called K-median Model (Vinod,1969) and another one proposed by (Rao,1971) , but it was not possible too. However, we realized that the integer model for the CP can be used to improve the solutions found by the ACP, using a hybrid method of heuristic and Integer Programming.

This paper proposes a hybrid method that uses an Evolutionary Algorithm with Local Search and Adaptive Memory (AECBL), and a linear and integer model, called k-median for the ACP. In this context, the AECBL is run initially to find the clusters. Then, the k-median model tries to improve the clusters generated by the AECBL.

This paper is organized as follows: In section 2, a partial literature review is presented; In section 3, the ACP is described. In section 3, the AECBL is described; in section 4, the hybrid method is described; in section 5 the results and analysis of computational tests are shown and, finally, in section 6 some conclusions are presented.

2 Related Work

The earliest papers on algorithms for Clustering Problems date from the 60's. However, most of the work done so far is for CP, where the number of clusters is known. One of the most known algorithm for CP is the k-Means (MacQueen,1967), which uses the centroid concepts (equation 4) to represent the clusters.

However, the ACP is not as exploited as the CP. The X-means (Pelleg & Moore, 2000) adapts the k-means for the ACP. The work of (Zalik, 2008) is more recent and also adapts the k-means to solve the ACP.

Some heuristics uses Genetic Algorithms (Tseng & Yang,2001) (Soares et al.,2006) (Saha & Bandyopadhyay,2008) (Liu et al., 2011) (He & Tan, 2012) (Chang et al.,2012)) (Liu et al., 2012), Simulated Annealing (Saha & Bandyopadhyay, 2010) (Saha & Bandyopadhyay, 2013), Ant Colony (Chowdhury & Das, (2012) , Neural Network (Tsenga et al., 2004) (Ai-sheng & Qi, 2011), Particle Swarm Optimization (Das et al., 2008) and GRASP (Nascimento et al., 2010).

Some heuristics uses Hierarquical Algorithms (Almeida et al.,2007) (Roldan, 2008) (Zhong et al, 2012).

The methods proposed in (Derya & Alp, 2007), (Duan et al., 2007) and (Seeman et al,20013) consists in dividing the space (database) that contains the items in a number of subspaces or cells, and once there are a relatively large number of points, they are potential candidates to form a cluster. The result of the method depends on the size of the cells, which is usually an input parameter.

A method developed by (Yujian, 2006) uses the concept of spanning tree. And in (Semaan et al.,2010) (Mok et al., 2012) they uses concept of Graph Partitioning .

In (Agraval et al., 2005), (Wang et al., 2007) and (Nosovskiy et al., 2008), density functions are used to define connectivity. They are based on the idea that points forming a dense region can be previously grouped in one cluster.

3 The Automatic Clustering Problem

The Automatic Clustering Problem (ACP) is presented as follows: given a set X of n objects $X=\{x_1, x_2, x_3....x_n\}$, where each object x_i is a tuple $(x_{i1},x_{i2},..., x_{ip})$, and each coordinate x_{ij} is related to an attribute of the object (each object may be a point in $\Re^p$ space.). The objective is to find a partition $X=\{C_1,C_2,...C_k\}$ of clusters, where k is not known, so that the similarity among the objects of the same cluster is maximized and, the similarity among objects of different clusters is minimized, under the following additional conditions:

$$C_i \neq \phi \text{, } for \quad i = 1,..,k \tag{1}$$

$$C_i \bigcap C_j = \phi \text{, } for\ i, j = 1,..,k\ e\ i \neq j \tag{2}$$

$$\bigcup_{i=1}^{k} C_i = X \tag{3}$$

Another definition needed in this paper is the concept of centroid of a cluster C_i. The replacement of a set C_i with similar t points (objects) by a single point $v_i = (v_{i1}, v_{i2}, ...v_{ip})$, that represent them, may be based on considering v_i as the centroid of C_i , and is given by the equation

$$v_{ij} = \frac{1}{t}\sum_{k=1}^{t} x_{ik} \quad , \quad j = 1,..,p \tag{4}$$

The Automatic Clustering Problem (ACP) can be considered an optimization problem, and defined as below:

$$\underset{C}{Maximize} \quad F(C) = \frac{1}{n}\sum_{i=1}^{n} s(x_i) \tag{5}$$

$$S.A. \qquad C \subset \underline{C} \tag{6}$$

where $C =\{C_1,C_2,...C_k\}$ is a particular partition of clusters and $\underline{C}$ is a set of all possible partition of the set X, with $k=2,...,n-1$ and $s(x_i)$ is a silhouette value of each point $x_i \in X$. The function (equation 5), called Silhouette Index, was proposed by

(Kaufman & Rousseeum, 1990). It returns values within the interval [-1,1] and is not necessary to define the value of k, because the most appropriate value of k is achieved by maximizing this function.

Let x_i be a point that belongs to the cluster $C_w \in C$, with $|C_w| = M > 1$. The distance between the points x_i e x_j is defined by $d_{i,j}$. The average similarity of x_i for all points $x_j \in C_w$ is given by $a(x_i)$ where

$$a(x_i) = \frac{1}{M-1} \sum_{\forall x_j \neq x_i,\, x_j \in C_w} d_{i,j} \tag{7}$$

When C_w has only one element, then $a(x_i) = 0$. It is assumed $C_t \in C$ with $t \neq w$ and $|C_t| = T > 1$. The average similarity of x_i for all points of C_t is

$$d(x_i, C_t) = \frac{1}{T} \sum_{\forall x_j \in C_t} d_{i,j} \quad . \tag{8}$$

Let $b(x_i)$ *be* the lowest among all $d(x_i,C_t)$. Thus

$$b(x_i) = Min \;\; d(x_i, C_t),\, C_t \neq C_w, C_t \in C \tag{9}$$

The Silhouette of the point $x_i \in X$ is

$$s(x_i) = \frac{b(x_i) - a(x_i)}{\max(a(x_i), b(x_i))} \tag{10}$$

and the Silhouette Index(SI) is

$$SI = \frac{1}{n} \sum_{x_i \in X} s(x_i) \tag{11}$$

4 The Evolutionary Algorithm

The proposed Evolutionary Algorithm with Local Search and Adaptive Memory (AECBL) consists of two phases. The first phase includes a pre-processing step to reduce the dimensions of the input data of the problem and a generation of good initial solutions. The second phase of the AECBL is a Genetic Algorithm (GA) with local search and concepts of adaptive memory.

The presentation of the AECBL is divided in six (6) parts: the Construction phase, the Individual Representation, the Fitness, The Adaptive Memory, The Local Search and Evolutionary Phase.

4.1 The Construction Phase

This phase is based on the criterion of density proposed in (Garai & Chaudhuri, 2003) and (Tseng & Yang, 2001). The idea is to reduce the problem keeping together in the same cluster objects belonging to a dense region. In this work, the objects are points in $\Re^p$ space and it is used the Euclidean metric as a measure of similarity for a distance between two points. This phase contains two procedures: Generate Partial Clusters (GPC) and Merge Partial Cluster (MPC).

1. ***Procedure GPC (X, u)***
2. ***FOR*** $i = 1$ ***TO*** n ***DO***
3. $d_{\min}(x_i) = \min \| x_i - x_j \|, i \neq j,\, j = 1,...n$
4. ***END FOR***

5. $d_{mean} = \frac{1}{n}\sum_{i=1}^{n} d_{\min}(x_i)$
6. $r = u * d_{mean}$
7. **FOR** $i = 1$ **to** n **DO**
8. $N_i = circulo\ (x_i, r)$
9. $T = T \bigcup N_i$
10. **END FOR**
11. *Sort T in descending order*
12. $i = 1$
13. **WHILE** $(T \neq \emptyset)$ **DO**
14. $C_i = next\ (N_j \in T)$
15. $T = T - \{N_j\}$
16. $i = i + 1$
17. **END WHILE**
18. *Let* $C = \{C_1, C_2, ...C_t\}$ *, the partial clusters*
19. **END Procedure**

Figure 1: The GPC procedure

The GPC procedure is presented in Figure 1. Initially, in lines 2, 3 and 4, for each point, is defined the least distance to any other point. Line 5 presents an average of distances, called d_{mean}. Then, each point $x \in X$ is considered the center of a circular region whose radius is $r = u \,.\, d_{mean}$, where u is a input parameter. In line 8, the number of points in the circle of center x and *radius r* ($N_i = circle\ (x_i, r)$) is calculated. These values are placed in a T list which is arranged in descending order.

1. **Procedure** *MPC* **(X, u, v,** $\mathbf{d_{mean}}$**, min)**
2. $C = GPC\ (X,u)$
3. $d_{adj} = v \,.\, d_{mean}$
4. **FOR** $i = 1$ **to** t **DO**
5. **IF** $(cardinality(C_i) <= min)$ **THEN**
6. $C_k = shortest\_distance\_centroid(C_i)$
7. **IF** $(\exists x_r \in C_i \ e \ \exists x_s \in C_k \ such\ that \ || x_r - x_s || < d_{adj})$ **THEN**
8. $C_k = C_k \ U \ C_i$
9. **END IF**
10. **END IF**
11. **END FOR**
12. *Let* $C = \{C_1, C_2, ...C_m\}$ *the initial clusters, where* $m <= t$
13. **END Procedure**

Figure 2: The MPC procedure

The elements of T are considered the partial clusters $C=\{C_1,C_2,..C_t\}$. In lines 12-17, each time a circle is selected, the points belonging to this circle are excluded from the other circles. With this procedure, the densest regions (within p-dimensional spheres) are selected.

After this initial procedure, a refinement is done reducing the number of partial clusters. Figure 2 shows the MPC procedure, which uses clusters generated by GPC (line 2). Then, in line 3, d_{adj} is calculated where $d_{adj} = v \cdot d_{medio}$ (v is a input parameter). Then, in lines 4, 5, 6, 7, 8, 9, 10 and 11 there is an aggregation of clusters of small cardinality ($\leq$ min points, where min is a input parameter), wich are very close to some other cluster of higher cardinality (using the shortest distance between their centroids, as can be seen in line 6). This is done checking if the smallest cluster is at a distance d_{adj} from another cluster. The clusters generated after GPC and MPC are called initial clusters.

4.2 The Individual Representation

Consider the initial clusters as $C = \{C_1,...,C_m\}$. Let $B= (B_1,B_2,...B_m)$ be a binary string of m positions that represents an individual to the GA. There is a closed relationship between the sets B and C. Each element of B corresponds to only one element of C. Each B_i has value 0 or 1. If $B_i = 1$, the initial cluster C_i will be part of the solution as parent cluster. If $B_i = 0$, C_i will be part of the solution as a child cluster. Each child cluster (one at a time) is joined to the parent cluster (only one) using the criterion of shortest distance between their centroids (equation 4). On each union, a new cluster is generated and a new value of the centroid is calculated. At the end, all children clusters will be linked to parent clusters to generate a solution. The clusters generated after this process are called final clusters $C = \{C_1,C_2,,...,C_k\}$, where $k \leq m$. The number of final clusters is the same than the number of elements with value 1 in the individual.

4.3 The Fitness

The solution is evaluated using the fitness defined in equation (11).

4.4 Adaptive Memory

The use of adaptive memory in metaheuristics is very promising as seen by (Glover & Kochenberger,2003), (Pailla et al., 2010) and (Silva & Ochi, 2010). The path to become a self-adaptive heuristic algorithm may be related to the process of calibration of its parameters and to use relevant information from past iterations in a search process. In this work, to make a good use of adaptive memory, it is used information contained in a pool of the best solutions generated. Adaptive memory uses a set of the best solutions generated by the algorithm, which are updated throughout the iterations. In this work it is used a set called ES (ELITE Set), which stores the best solution for each iteration. The ES's solutions are different. The ES is used at two different moments in the proposed algorithm: in the middle of the algorithm to perform the local search Path Relinking, whose goal is to look for the best solutions found by the algorithm and to store in the ES, and to perform local search at the end of the algorithm, Exchange pairs, which is an exhaustive search to try to improve the best solutions found along the iterations of the algorithm and stored in the ES.

4.5 The Local Search

The purpose of employing local search combined with the GA is to refine the individuals obtained by the GA. In empirical tests performed, it was observed that local searches can greatly improve the individuals found by the GA.

The first local search used in this work, called Single Inversion, tries to find individuals closed to that generated by the AG, exchanging each element of the individual. The second local search is called Path Relinking, which tries to find better individuals between two individuals of good quality. It was observed, empirically, with these two local searches within the genetic algorithm, the quality of the individual is greatly improved, and the number of iterations required for convergence is reduced. The best solutions generated by the AECBL are stored in the ES, which can be improved through an exhaustive search, called Exchange Pair. These local searches are described below.

4.5.1 The Local Search Single Inversion

The basic idea of this first local search, called Single Inversion, is to improve the individual by analyzing individuals close to it. For that, this search exchanges value of each element of the individual (1 to 0, or 0 to 1), one at a time, generates a new individual and calculates the new value of Silhouette Index. But, the algorithm only accepts the change of the individual, if the new value is better than the previous one. This procedure is shown in Figure 3.

Procedure *Single Inversion* (C,B^0)
1. *Let* $C= \{C_1,C_2,...,C_m\}$ *a set of m initial clusters*
2. *Let* $B^0= (B_1,B_2,...B_m)$ *an individual of m positions*
3. s_0 = *Generated_solution*(C,B^0)
4. f^* = *Calculate_Silhouette_Index*(s_0)
5. ***FOR*** $i = 1$ *to* m ***DO***
6. B^1 = *Exchange_Value*(i,B^0)
7. s_1 = *Generated_solution*(C,B^1)
5. f^1 = *Calculate_Silhouette_Index*(s_1)
6. ***IF*** $(f^1>f^*)$ ***THEN***
7. $B^0=B^1$
8. $f^*=f^1$
9. ***END IF***
10. ***END FOR***
11. *Return* B^0
12. ***END*** *procedure*

Figure 3: The local search Single Inversion

For example, imagine that the individual is (0101101). Initially, the first element is exchanged. Then the new individual is (1101101). So, the Silhouette Index of the new individual is calculated. If this solution has the Silhouette Index value higher than the previous, then it will be the new current individual. So, the second element is exchanged. The new individual is now (1001101). If this individual has a Silhouette Index value higher than the previous one, then change is accepted. Otherwise, the previous individual is maintained. The search ends when all elements of the individual are tested.

The local search Single Inversion is justified, once the optimal number of clusters is one of the objectives of the ACP problem, and the inclusion or removal of an element with value 1 in the individual can improve the new solution.

4.5.2 The Path Relinking

The second local search proposed, called Path Relinking (PR), was first proposed by (Glover & Kochenberger, 2003) for the metaheuristic Tabu Search and Scatter Search. The basic principle of the PR is that between two solutions of good quality, there may be a third better than the others. The PR consists of tracing the path between a base solution (SB) and a target solution (SA), that have good quality and evaluate the intermediate solutions obtained along the path. The objective of this search is to find better solutions than the SB and SA.

In this work, a Path Relinking uses a path from the individual of better quality (B^0) for lower quality (B^1). The goal of the PR is to insert, at each iteration, an element of a target solution (B^1) in the base solution (B^0). In this context, the first intermediate solution is obtained as follows: take the elements until the i-th element of the B^1 and exchange by the elements of the B^0, if they are distinct, and repeat this procedure for all the elements that make a solution. With each permutation, a new intermediate solution is generated. Each intermediate solution is evaluated, and the best (which returns the best value of Silhouette Index) is chosen. The process is repeated until the B^0 is equal to the B^1. This procedure is shown in Figure 4.

Procedure *Path Relinking* (C,B^0,B^1)

1. *Let* $C= \{C_1,C_2,...,C_m\}$ *a set of m initial clusters*
2. *Let* $B^0 = (B^0_1,B^0_2,...,B^0_m)$ *an individual of m positions*

```
3.  Let B^1 = (B^1_1,B^1_2,...,B^1_m ) an individual  of m positions
4.  s_0 = Generated_solution(C,B^0)
5.  s_1 = Generated_solution(C,B^1)
6.  f_0= Calculate_Silhouette_Index(s_0)
7.  f_1= Calculate_Silhouette_Index(s_1)
8.  f_* = f_0
9.   FOR i = 1 to  m  DO
10.          B^2 = Exchange_Value(i,B^0,B^1)
11.          s_2 = Generated_solution(C,B^2)
12           f_2 = Calculate_Silhouette_Index(s_2)
13           IF (f_2>f^*) THEN
14.                  B^0=B^2
15.                  f^*=f^2
16.          END IF
17.  END FOR
18.   Return  B^0
19. END procedure
```

Figure 4 : The local search Path Relinking

Additionally, another justification for the use of PR in the ACP is that finding the optimal number of clusters is an objective of the problem, and to analyze two individuals with different numbers of clusters allows to obtain intermediate individuals with different number of clusters.

4.5.3 The Local Search Exchange Pair

The third local search proposed, called Exchange Pair, is an intensive search, that changes the elements with two different values of an individual. For example, suppose that the individual is B = (10111010). Firstly, the first and the second element of the individual are exchanged. So, the new individual is (01111010). If the new individual improves the value of the Silhouette Index from the old, then, it will be accepted and the process continues. The next exchange is made between the first and third element of B. The local search ends when all changes between two elements of B with different values are tested. This procedure is shown in Figure 5.

```
1.  Procedure Exchange Pair (C,B^0)
2.  Let  C= {C_1,C_2,...,C_m}  a  set of m initial clusters
3. Let B^0 = (B_1,B_2,...,B_m) an individual of m positions
4. s_0 = Generated_solution(C,B^0)
5. f^*= Calculate_Silhouette_Index(s_0)
6.    FOR i = 1 to  m-1  DO
7.           FOR j =i+1 to  m  DO
8.                   B^1= Exchange_position(i,j,B^0)
9.                   s_1 = Generated_solution(C,B^1)
6.                   f^1= Calculate_Silhouette_Index(s_1)
7.                   IF (f^1>f^*) THEN
8.                          B^0=B^1
9.                          f^*=f^1
10.                  END IF
11.          END FOR
12.  END FOR
13.   Return  B^0
14. END procedure
```

Figure 5: The local search Exchange Pair

This local search tries to find different individuals without changing the number of elements with value 1 in each individual. Due to high computational time required, this search is done only at the end of heuristic in the best individuals of ES.

4.6 The Evolutionary Phase

This phase consists of a traditional Genetic Algorithm (GA) including Adaptive Memory and three local searches.

The traditional GA has not always worked successfully, especially in combinatorial optimization problems where there are already very efficient heuristics. This work proposes the use of local search Inversion Individual, Path Relinking and Exchange Pair.
The initial population is constructed from the initial clusters generated during the construction phase. To build the initial population of size n, the algorithm generates randomly a greater number of individuals (10n) and chooses those with the highest fitness. This allows the AG to start with better quality population. The reason of using GA`s is generate any number of values 1 in the individual. As the ideal number of Clusters is one of the goals of the problem, the GA can generate different numbers of clusters.

For selection of individuals for crossover, two operators are used alternately. The first operator selects randomly two individuals among the 60% from the population with the highest fitness. The second operator selects randomly an individual among the 60% from the population with the highest fitness and the other from the remaining 40%. The operator of crossover works as follows: pairs of positions are obtained randomly and the elements between these positions are changed. Individuals are subject to crossover with probability p_c. The mutation operator performs random exchange of some values of individuals with probability p_m, to find new areas of research, from selected individuals in the crossover. After applying the crossover and mutation operators, the descendants who obtain the fitness values better than the values of the current population are inserted into the new population.

At each *w* iterations (w is an input parameter) and q iterations (q is an input parameter), the local search Single Inversion are applied in the best individuals of the population and the local search Path Relinking are applied with the best element of the ES and the best element of population. At each generation, the best solution is stored in the ES and in the end, the local search Exchange Pair will be used in ES, whose goal is to investigate different possible solutions with the same number of parent clusters. This AECBL procedure is shown in Figure 6.

```
1. Procedure AECBL (X,Tpop, Gmax , Pc , Pm, ES,u,v,w,q)
2.   G = Procedure GPC(X,u)
3.   G´ = Procedure MPC (X, u, v, dmean, min)
4.   P = Generate_Initial_Population (G´, Tpop)
5.   FOR k = 1 to Gmax DO
6.   i = 1
7.       WHILE i < 40% Tpop DO
8.           Select (p1, p2)
9.           q = random (100)
10.          IF ( q >= Pc * 100) THEN
11.           i = i + 1
12.              IF Crossover (p1,p2,f1,f2) THEN
13.                              Mutation ( p1,p2,Pm)
14.              END IF
15.              IF (Evaluate_Solution (f1,f2) ) THEN
16.                              Update_population (p1, p2, P)
17.              END IF
18.          END IF
19.     END WHILE
20.  IF ( i mod w = 0 ) THEN
21.                          Single Inversion (P)
22.   END IF
23.  IF ( i mod q = 0 ) THEN
24.                          Path Relinking (P)
```

```
25.  END IF
26.  Update_Elite Set(ES)
27.  END FOR
28.  Exchange Pair (ES)
29.  Return (Best_solution)
30.  END AECBL
```

Figure 6: The AECBL procedure

5 A Hybrid Method Using AECBL and the K-Median Model (AECBL+KM)

In the description of the k-median is necessary to define the cluster median, as one object which belongs to the cluster and is closed to all other objects in the cluster.

The goal of this model is to select k cluster median from the set X, a cluster median for each cluster, where the sum of the distances of the other objects of the cluster to the cluster median is minimal. Each object can belong to only one cluster and the number of clusters k is given.

The K-median model is showed below. The restriction (13) ensures that each object i will be associated with only one cluster median j. Restriction (14) ensures that exactly k points are chosen as cluster median and, finally, the restriction (15) ensures that the object i is associated with object j only if the object j is a cluster median.

$$Min \quad \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} x_{ij} \tag{12}$$

$$S.A. \quad \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i, \quad i = 1,...,n \tag{13}$$

$$\sum_{j=1}^{n} x_{jj} = k \tag{14}$$

$$x_{ij} \le x_{jj} \quad \forall i, j, \quad i = 1,...,n \quad j = 1,...,n \tag{15}$$

$$x_{ij} \in \{0,1\} \quad \forall i, j, \quad i = 1,...,n, \quad j = 1,...,n \tag{16}$$

$$x_{ij} = \begin{cases} 1, \textit{if the object } i \textit{ is associate to cluster median } j \\ 0, \quad \textit{other case} \end{cases}$$

$$x_{jj} = 1, \textit{the object } j \textit{ is the cluster median.}$$

$$d_{ij} = \textit{dis tance between objects } i \textit{ e } j.$$

The goal of the method *AECBL+KM* is to improve the solutions found by the AECBL with a linear integer model *k-median*. The hybrid method works as follows: firstly the heuristic is executed using the silhouette index and find a value *k* for the number of clusters. Then, the Integer model is executed using the k previously generated and some information about the clusters found, and a new solution is found. Then, the silhouette index is calculated to this new solution. This procedure is shown in Figure 7.

Procedure *AECBL+KM*
1. Run AECBL procedure
2. Let k the number of clusters found in AECBL
3. so = K-median (k)
4. Calculate_Silhouette_Index(s_0)
12. ***END*** *procedure*

Figure 7: The AECBL+KM procedure

There is a relationship between the functions k-median and index silhouette. They are different, but with a common part of the criterion of evaluation. This common part is the sum of distances within the cluster. The k-median is defined as the sum of all objects of each cluster to an object near to the center of that cluster, called cluster median. Therefore, if the clusters are cohesive and separated, this sum is smallest. The silhouette Index uses two criteria to evaluate the quality of the solution. It tries to maximize the distance between different clusters and minimizes the sum of distances within the cluster. That is, if the clusters are more cohesive and separated, the sum of distances within the cluster is the least, and so it is better to the silhouette index.

6 Computational Results

To analyze the performance of the hybrid method *AECBL+KM* proposed here, tests were carried out on computer. The *AECBL+KM* was compared with two algorithms from the literature, known as *CLUES (*Wang et al., 2007) and MRDBSCAN (Seeman et al,20013). The *CLUES* was implemented using the statistical software R and its source code was available and used to perform the proposed instances. The source code of CLUES was executed without any changes.

The *AECBL+KM* was implemented in C++ compiler using Ubuntu Linux 7.5 environment. To execute the k-median model was used the library from the software XPRESS. It was used a computer with processor Intel Xeon Quad-core 3.0 Ghz with 16G of RAM to count the processing time.

The algorithms were tested with some instances. Five of them are known in the literature as RuspiniDataSet (Ruspinil, 2006), Iris Plants Database (Fisher, 1936), MaronnaDataSet (Maronna & Jacovkis, 1974), 200DATA (Fisher, 1936) and VowelDataSet (Hastie et al., 2001).

Other instances were built using a graphical tool called Dots, developed by (Soares et al.,2006) and presented in (Cruz,2010). The main feature of this tool is to generate instances where the optimal solution can be viewed. Each instance has a number of points and the ideal number of clusters (the ideal number of clusters is not send to the algorithms, since one of its objectives is to find this value). For example, the instance 100p5c has 100 points and 5 clusters. All instances are in space R^2. The method was tested with two kinds of instance: the well-defined, where the clusters are well defined and separated, and not defined, where there is a lot of points among the clusters and in some cases, the optimal number of clusters is not well defined.. The instances not defined are characterized by a number "1" in the end of the name, as in 300p4c1, and the well defined by a "c" in the end of their names, as in 300p4c. Some instances can be viewed in (Cruz,2010). To access the instances, just contact us.

The parameters used in *AECBL* were defined from preliminary tests. The value of *u* was used between *1.5 and 4.5* and the value of *v* was *equal to 2* (only for instances with more than 200 points. Otherwise, the value of v is 0). The value of parameter *min* was 4. The crossover rate of each generation was set at 80% of population size and the mutation rate was set at 10% of the population size. Therefore, the probability of crossover P_c and mutation P_m were fixed at 0.20 and 0.90. The population size chosen was 1/3 of the size of the individual with a maximum of 20 (twenty) individuals. The size of the Elite Set (ES) was set to 5 (five) elements. The stopping criterion of the algorithm was the maximum number of iterations set to 50. The value of t, which indicates the frequency of local search Inversion Individual iterations was set at 5. This search was performed only in the three best elements of the population. The Local search Path Relinking was performed 4 times in iterations 18, 28, 38 and 48. The *AECBL+KM* uses the same parameters than *AECBL*.

Table1: *Comparison between AECBL and AECBL+KM*

	AECBL			AECBL+KM			
Instance	IS	t(s)	NC	IS	t(s)	NC	%
RuspiniDataSet	0,738	0,8	4	0,738	3,3	4	0,00
Iris Plants Database	0,686	2,7	3	0,686	6,1	3	0,00
MaronnaDataSet	0,575	2,8	4	0,575	9,5	4	0,00
200DATA	0,823	4,3	3	0,823	9,8	3	0,00
VowelDataset	0,425	15,4	3	0,425	82,6	3	0,00
100p3c	0,786	2,2	3	0,786	5,6	3	0,00
100p3c1	0,58	2,4	3	0,58	5,8	3	0,00
100p5c1	0,696	1,6	7	0,696	4,3	6	0,00
100p7c	0,834	1,8	7	0,834	4,7	7	0,00
100p7c1	0,491	1,9	7	0,495	4,9	7	**0,81**
100p10c	0,834	1,8	10	0,834	6,3	10	0,00
200p2c1	0,764	2,3	2	0,764	7,1	2	0,00
200p3c1	0,680	3,1	3	0,68	8,7	3	0,00
200p4c	0,773	3,1	4	0,773	8,5	4	0,00
200p4c1	0,745	3,2	4	0,745	7,7	4	0,00
200p7c1	0,576	2,7	13	0,578	7,2	13	**0,35**
200p12c1	0,575	2,8	13	0,575	10,5	12	0,00
300p2c1	0,776	5,1	2	0,777	18,3	2	**0,13**
300p3c	0,766	7,2	3	0,766	19,2	3	0,00
300p3c1	0,667	6,4	3	0,677	23,4	2	**1,48**
300p4c1	0,591	5,7	2	0,591	21,8	6	0,00
300p6c1	0,664	5,4	8	0,664	20,1	8	0,00
400p3c	0,799	9,1	3	0,799	31,2	3	0,00
400p4c1	0,599	6,2	4	0,607	32,3	4	**1,32**
400p17c1	0,514	10,6	2	0,514	42,4	2	0,00
500p3c	0,825	9,5	3	0,825	63,7	3	0,00
500p4c1	0,658	8,1	5	0,659	62,2	4	**0,15**
500p6c1	0,629	8,5	6	0,632	67,3	6	**0,47**
600p3c1	0,721	18,3	3	0,721	100,2	3	0,00
600p15c	0,781	32,7	15	0,781	87,9	15	0,00
700p4c	0,797	31,5	4	0,797	186,5	4	0,00
700p15c1	0,68	23,4	15	0,692	145,3	15	**1,73**
800p4c1	0,702	38,6	4	0,705	266,3	4	**0,43**
800p10c1	0,468	34,7	2	0,478	246,8	10	**2,09**
800p18c1	0,691	24,9	19	0,691	210,2	19	0,00
800p23c	0,787	55,4	23	0,787	230,7	23	0,00
900p5c	0,716	71,2	5	0,716	353,3	5	0,00

900p12c	0,841	70,8	12	0,841	317,8	12	0,00
1000p5c1	0,639	71,5	5	0,643	522,6	5	**0,62**
1000p6c	0,736	76,7	6	0,736	445,5	6	0,00
1000p14c	0,831	84,7	14	0,831	440,3	14	0,00
1000p27c1	0,516	112,3	25	0,516	495,7	25	0,00

Firstly, the algorithms AECBL e AECBL+KM was compared to verify the efficiency of application of the exact method k-median in the algorithm AECBL. The results are shown in Table 1. The algorithms were run 5 (five) times for each instance. In this Table, the column Instance shows the name of the instance. The column IS show the average value of Silhouette Index found by each algorithm. The Columns t(s) and NC shows the average time in seconds and the number of clusters of the best solution found by each algorithm. The column % contains the deviation of the best solution, according to the following definition: deviation = (Best - IS) / 100 * Best. The best results are highlighted in bold.

It was observed that in 17 instances well-defined (RuspiniDataSet, MaronnaDataSet, 200DATA, 100p3c, 100p7c, 100p10c, 200p4c, 300p3c, 400p3c, 500p3c, 600p15c, 700p4c, 800p23c, 900p5c, 900p12c, 1000p6c, 1000p14c) the solutions found by the algorithms AECBL and AECBL+KM are the same and have the same Silhouette Index value. And, with the instances not-defined, the *AECBL+KM* improves the solutions found by the *AECBL* in 11 instances. But, the runtime of the *AECBL+KM* is much greater than the *AECBL* because the exact model is very slow.

It was realized that in instances that have well-defined clusters and no objects between them, the solutions found by the AECBL and the hybrid method have the same value for the silhouette index. Given the number of clusters k, and suppose this number is the best possible value for the silhouette Index. So, if the objects of each cluster are in their best places, this implies that the function of k-median achieves the lowest possible value and silhouette index achieves the greatest possible value. Thus, the sum of distances among objects within the clusters is the minimum and the distance among different clusters is the maximum. That is, there is a relationship between the function of k-median and silhouette index.

Since the solutions have the same values for the silhouette index in well-defined instances, it was tested in instances that there are a lot of objects among the clusters. These instances make the clustering process more complex. However, it was observed that, since the number of k is fixed and is the best, the model tends to place the objects in the best position for each cluster. This is can be viewed with the instance 300p3c1, in figure 8. Thus, sometimes the k-median also improves the index silhouette from AECBL. But it is not guaranteed that this always happens, because the AECBL and k-median model uses different functions.

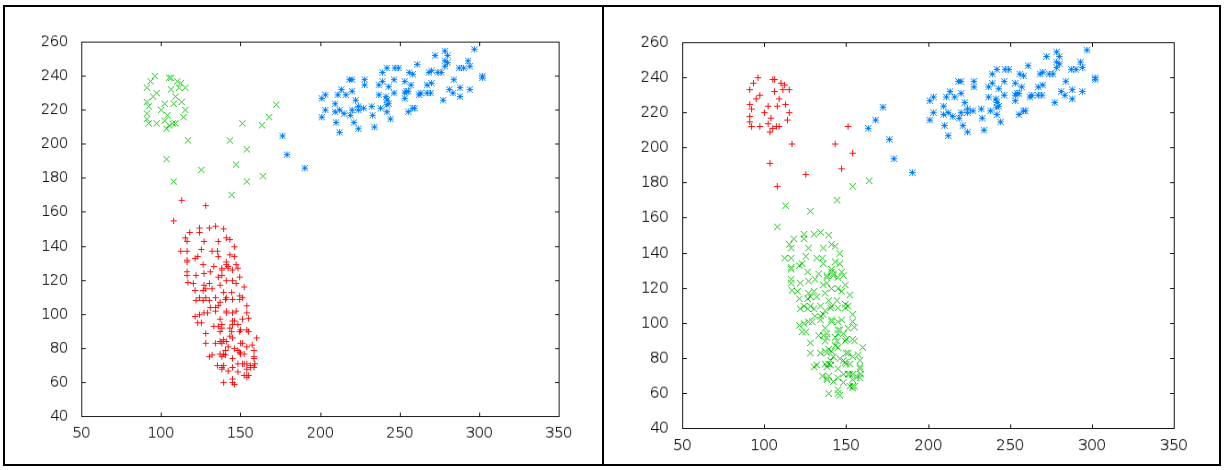

Figure 6: Instance 300p3c1 : Image from solution by AECBL and AECBL+KM

The second comparison is between the *AECBl+KM* , *CLUES* and MRDBSCAN. This comparison is shown in table2. The algorithms *AECBl+KM* and *CLUES* were run 5 (five) times for each instance. The columns have the same meanings as in Table 1. The column Best show the best value of Silhouette Index found by the tree algorithms. In this comparison, the difference between the average deviation is large, 1.30 to 5.49 and 31.06, meaning that the *AECBL+KM* is closer to the best solutions. Besides having an average deviation much better, AECBL+KM also achieves the best values in a larger number of instances. The AECBL+KM reaches the best value in 29 instances, in a total of 42. The CLUES reaches the best value in 21 instances and MRDBSCAN in 15 instance. In relation to the Runtime, the *AECBl+KM* is slower because the exact model K-median is very slow.

Table2: *Comparison between AECBL+KM* and *CLUES*

		AECBL+KM			**CLUES**			**MRDBSCAN**		
INSTANCE	**BEST**	**SI**	**NC**	**%**	**SI**	**NC**	**%**	**SI**	**NC**	**%**
RuspiniDataSet	0,738	0,738	4	**0,00**	0,738	4	**0,00**	0,738	**4**	**0,00**
Iris Plants Database	0,687	0,686	3	0,15	0,563	3	18,05	0,687	**2**	**0,00**
MaronnaDataSet	0,575	0,575	4	**0,00**	0,575	4	**0,00**	0,562	**2**	2,26
200DATA	0,823	0,823	3	**0,00**	0,462	3	43,86	0,823	**3**	**0,00**
VowelDataset	0,448	0,425	3	5,13	0,448	9	**0,00**	0,417	**2**	6,92
100p3c	0,786	0,786	3	**0,00**	0,786	3	**0,00**	0,786	3	**0,00**
100p3c1	0,597	0,580	3	2,85	0,597	3	**0,00**	0,104	5	82,58
100p5c1	0,703	0,696	7	1,00	0,703	6	**0,00**	0,424	2	39,69
100p7c	0,834	0,834	7	**0,00**	0,834	7	**0,00**	0,834	7	**0,00**
100p7c1	0,551	0,491	27	10,89	0,551	7	**0,00**	-0,012	2	102,18
100p10c	0,834	0,834	10	**0,00**	0,834	10	**0,00**	0,691	8	17,15
200p2c1	0,764	0,764	2	**0,00**	0,591	3	22,64	0,624	2	18,32
200p3c1	0,680	0,680	3	**0,00**	0,674	3	0,88	0,648	2	4,71
200p4c	0,773	0,773	4	**0,00**	0,773	4	**0,00**	0,773	4	**0,00**
200p4c1	0,754	0,754	4	**0,00**	0,754	4	**0,00**	0,622	3	17,51
200p7c1	0,576	0,576	8	**0,00**	0,555	9	3,65	0,392	3	31,94
200p12c1	0,570	0,570	8	**0,00**	0,562	9	1,40	0,403	3	29,30
300p2c1	0,776	0,776	2	**0,00**	0,490	5	36,86	0,620	4	20,10
300p3c	0,766	0,766	3	**0,00**	0,552	3	27,94	0,766	3	**0,00**
300p3c1	0,677	0,677	3	**0,00**	0,592	4	12,56	0,639	2	5,61
300p4c1	0,592	0,592	2	**0,00**	0,592	7	**0,00**	0,269	3	54,56
300p6c1	0,661	0,661	8	**0,00**	0,579	7	12,41	0,548	2	17,10
400p3c	0,799	0,799	3	**0,00**	0,799	3	0,06	0,799	4	**0,00**
400p4c1	0,620	0,607	4	2,10	0,620	4	**0,00**	0,379	2	38,87
400p17c1	0,552	0,514	2	6,88	0,552	15	**0,00**	0,183	14	66,85
500p3c	0,825	0,825	3	**0,00**	0,825	3	**0,00**	0,825	3	**0,00**
500p4c1	0,660	0,660	3	**0,00**	0,515	3	21,97	0,305	2	53,79
500p6c1	0,669	0,631	6	5,68	0,669	6	**0,00**	0,494	12	26,16
600p3c1	0,721	0,721	3	**0,00**	0,703	3	2,50	0,686	2	4,85
600p15c	0,781	0,781	15	**0,00**	0,753	16	3,59	0,781	15	**0,00**
700p4c	0,797	0,797	4	**0,00**	0,797	4	**0,00**	0,797	4	**0,00**
700p15c1	0,692	0,680	15	1,73	0,660	17	4,62	0,122	2	82,37
800p4c1	0,714	0,705	4	1,26	0,714	4	**0,00**	0,508	2	28,85

800p10c1	0,507	0,478	2	5,72	0,507	8	**0,00**	0,079	2	84,42
800p18c1	0,694	0,691	19	0,43	0,694	19	**0,00**	0,265	24	61,82
800p23c	0,787	0,787	23	**0,00**	0,739	22	6,10	0,787	23	**0,00**
900p5c	0,716	0,716	5	**0,00**	0,613	7	14,39	0,716	5	**0,00**
900p12c	0,841	0,841	12	**0,00**	0,798	10	5,11	0,841	12	**0,00**
1000p5c1	0,643	0,643	5	**0,00**	0,558	7	13,22	0,164	2	74,49
1000p6c	0,736	0,736	6	**0,00**	0,736	6	**0,00**	0,736	6	**0,00**
1000p14c	0,831	0,831	14	**0,00**	0,767	11	7,70	0,808	15	2,77
1000p27c1	0,563	0,516	24	8,35	0,563	14	**0,00**	-0,293	3	152,04
Deviation Average				1,30			5,49			31,06
Number of Best				29			21			15

To ilustrade the difficulty of instances, a few of them are shown in Figures 7, 8, 9, 10, 11 and 12. We just have the images from the *AECBl+KM* and *CLUES*. The images from MRDBSCAN wasn't available. The instances have two main formats: clusters are separated and well defined (Figures 7 and 9); the clusters have many points among them, and in some cases, the clusters are not defined, because is not known where ends one cluster and begins the other (Figures 8, 10, 11and 12). The AECBL+KM is more efficient and find either good or optimal solutions. Deviation average confirm this, with value of 1.30.

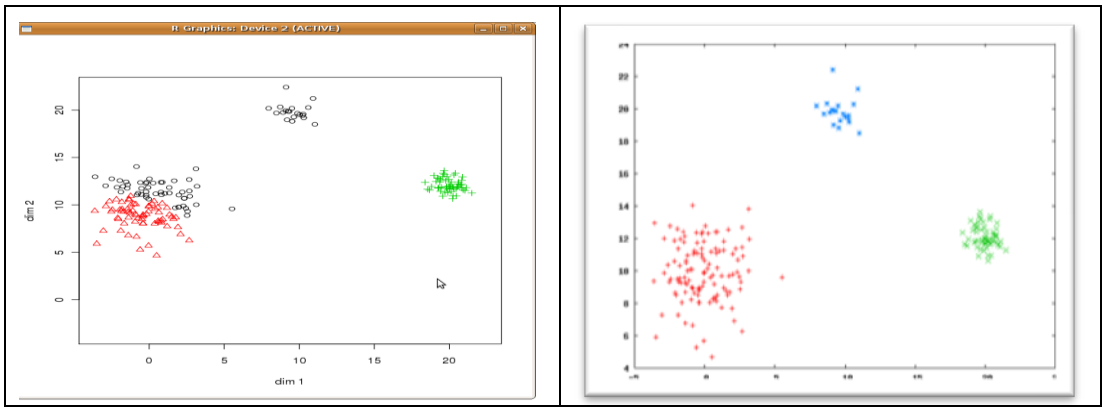

Figure 7 : Instance 200DATA : Image from solution by Clues and AECBL+KM

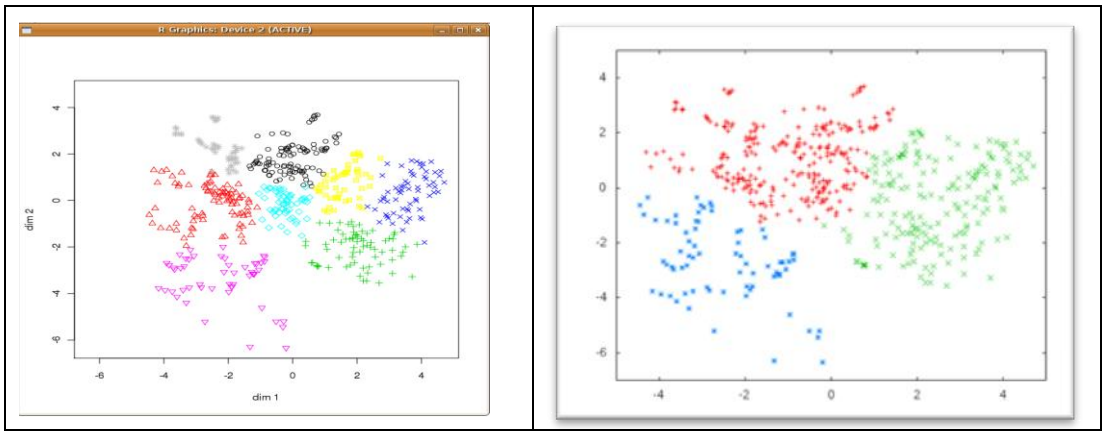

Figure 8 : Instance VowelDataSet : solution by Clues and AECBL+KM

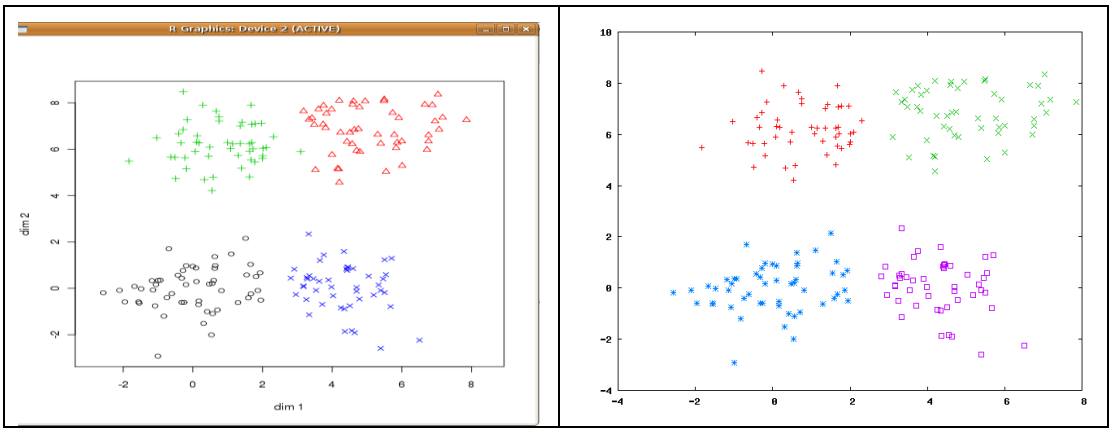

Figure 9 : Instance Maronna : solution by Clues and AECBL+KM

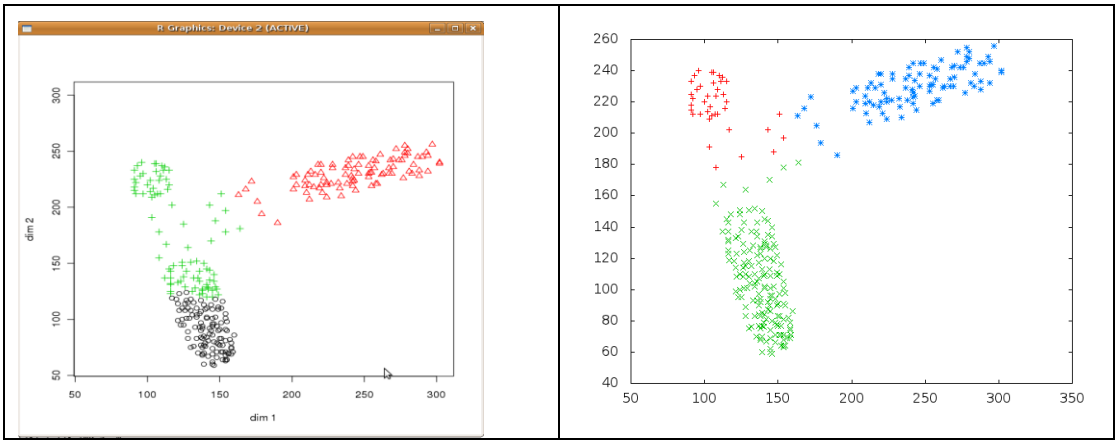

Figure 10 : Instance 300p3c1 : solution by Clues and AECBL+KM

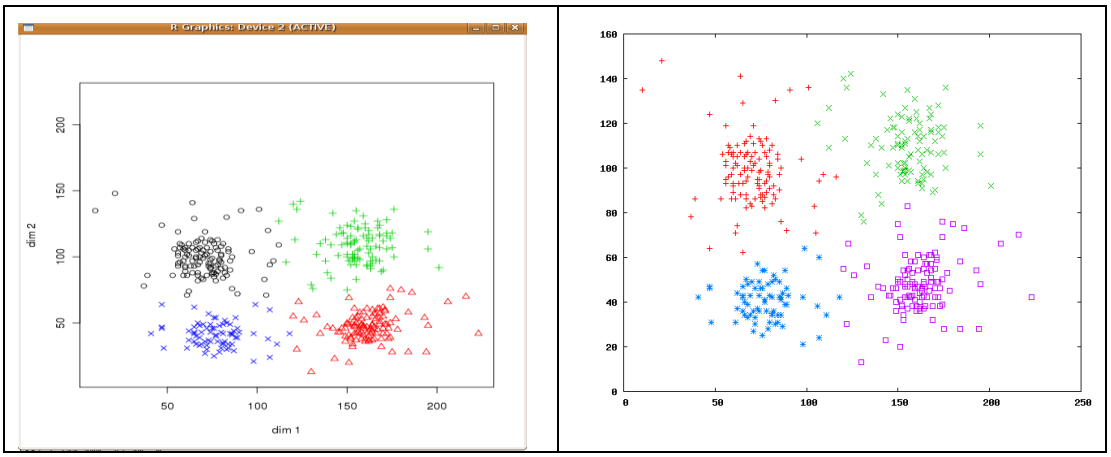

Figure 11 : Instance 400p4c1 : solution by Clues and AECBL+KM

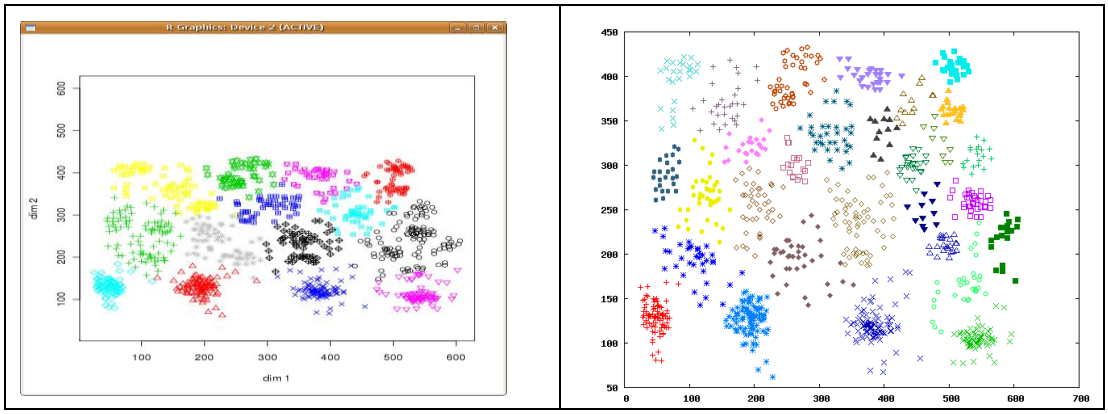

Figure 12 : Instance 1000p27c1 : solution by Clues and AECBL+ KM

7 Conclusions

It was presented a new hybrid method joining Evolutionary Algorithm with Integer Programming. The goal of this method is tries to improve the solutions found in Evolutionary Algorithm using Integer Programming.

The hybrid model *AECBL+KM* can improve the solutions found by the *AECBL* because, in some cases, it can find the best place of each object. When compared with the *CLUES and* MRDBSCAN , it was realized that it was very robust and reaches the best solutions. But this does not happen always. The mainly reasons are: the value of k passed to the hybrid model may not be the ideal; and the *AECBL+KM* is trying to improve the solution using a function different than the Silhouette Index. As a negative point, the processing time of the hybrid model is much higher.

8 References

Ai-sheng, L. & Qi, Z. , (2011) Automatic modulation classification based on the combination of clustering and neural network. The Journal of China Universities of Posts and Telecommunications 18(4). pp. 13–19 (neural network)

Almeida, J.A.S.& Barbosa, L.M.S. & Pais , A.A.C.C, Formosinho, S.J. (2007). Improving hierarchical cluster analysis: A new method with outlier detection and automatic. Chemometrics and Intelligent Laboratory Systems 87. pp. 208 – 217 (hierarquical)

Agrawal, R. & Gehrke, J. & Gunopulos, D. & Raghavan, P. (2005). Automatic Subspace Clustering of High Dimensional Data. Data Mining and Knowledge Discovery, 11, pp. 5–33

Chang, D.& Zhang, X.& Zheng, C.& Zhang, D.(2010). A robust dynamic niching genetic algorithm with niche migration for automatic clustering problem. Pattern Recognition 43. pp. 1346–1360(GA)

Chowdhury A. & Das S. (2012) Automatic shape independent clustering inspired by ant dynamics . Swarm and Evolutionary Computation 3. pp. 33–45 (ant colony)

Cruz, M. D. (2010) O problema de Clusterizacao Automatica. Tese de Doutorado. COPPE-Eng. De Sistemas e Computacao. UFRJ. Orientador. Adilson Elias Xavier e Luiz Satoru Ochi. http://www.cos.ufrj.br/uploadfiles/1281027685.pdf

Das, S. & Abraham, A. & Konar, A. (2008) Automatic kernel clustering with a Multi-Elitist Particle Swarm Optimization Algorithm. Pattern Recognition Letters 29. pp. 688–699 (particle swarm optimization)

Derya, B. & Alp, K. (2007). ST-DBSCAN: An algorithm for clustering spatial–temporal data. Data & Knowledge Engineering 60 , pp. 208-221

Duan, L. & Xu, L. & Guo, F. & Lee, J. & Yan, B. (2007); A local-density based spatial clustering algorithm with noise, Information Systems 32;pp. 978-986

Everitt, B. S., 2001. *Cluster Analysis.* Oxford University Press, New York.

Fisher, R. (1936). The use of multiple measurements in taxonomic problems. Annual Eugenics 7,pp. 179-188.

Garai, G. & Chaudhuri, B. B.(2004). A novel genetic algorithm for automatic clustering. Pattern Recognition Letters, pp. 173-187.

Glover, F. & Kochenberger, G. A. (2003) Handbook of Metaheuristics. Kluwer Academic Publishers.

Hardy, A. (1996). On the number of clusters. *Computational Statistics and Data Analysis* **23**, pp. 83–96.

Hastie, T. & Tibshirani, R. & Friedman, J. (2001). The Elements of Statistical Learning. Data Mining, Inference, and prediction. Springer.

He, H. & Tan, Y., (2012) A two-stage genetic algorithm for automatic clustering. Neurocomputing 81 pp. 49–59 (GA)

Kaufman, L. & Rousseeum, P. J. (1990). Finding Groups in Data : An introduction to cluster Analysis. A Wiley-Intercience publication.

Liu R. & Jiao L.& Zhang X.& Li, Y. (2012) Gene transposon based clone selection algorithm for automatic clustering. Information Sciences (204). pp. 1–22 (Gene Transposon)

Liu, Y.& Wu X. & Shen , Y. (2011) Automatic clustering using genetic algorithms. Applied Mathematics and Computation 218. pp. 1267–1279(GA)

MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, 1, pp. 281-297

Maronna, R. & Jacovkis, P. M. (1974). Multivariate clustering procedures with variable metrics. Biometrics30, pp. 499-505.

Mok P.Y. & Huang H.Q. & Kwok Y.L. & Au J.S.(2012) A robust adaptive clustering analysis method for automatic identification of clusters . Pattern Recognition 45. pp 3017–3033. (graph partitioning)

Nascimento, M.C.V.& Toledo, F.M.B & Carvalho, A.C.P.L.F. (2010) Investigation of a new GRASP-based clustering algorithm applied to biological data, Computers & Operations Research, 37, 1381-1388.

Nosovskiy, G. V.& Liu, D.& Sourina, O. (2008) Automatic clustering and boundary detection algorithm based on adaptive influence function. Pattern Recognition 41. pp. 2757 – 2776 (sem parametro)

Pailla, A. & Parada, V. & Trindade, A. R. & Ochi, L. S. (2010). A numerical comparison between simulated annealing and evolutionary approaches to the cell formation problem". Expert Systems with Application - ELSEVIER – Volume 37, pp. 5076-5083.

Pelleg, D & Moore, A. (2000). X-means: Extending K-means with efficient estimation of the number of clusters. Proceeding of the 17th International Conference on Machine Learning, pp.727-734.

Rao, M. R.; (1971); Cluster Analysis and Mathematical Programming; Journal of the American Statistical Association, vol. 66.pp.622-626.

Roldan, J. F.(2008) Analise multivariada de agrupamentos de dados utilizando tecnicas rigidas e difusas. Tese de Mestrado. Orientadores. Airton Fontenele Sampaio Xavier. e Marcos Jose Negreiros Gomes. Universidade Estadual do Ceara.

Ruspini, E. H. (1970). Numerical methods for fuzzy clustering. Information Science.pp. pp. 319-350.

Saha, S. & Bandyopadhyay, S. (2008). A Point Symmetry-Based Clustering Technique for Automatic Evolution of Clusters. IEEE Transactions on Knowledge and Data Engineering, pp.1-18 (GA)

Saha, S. & Bandyopadhyay, S. (2010)A symmetry based multiobjective clustering technique for automatic evolution of clusters. Pattern Recognition 43 .pp. 738 -751. (simulated annealing)

Saha, S. & Bandyopadhyay, S. (2013) generalized automatic clustering algorithm in a ultiobjective framework. Applied Soft Computing 13. pp 89–108 (simulated annealing)

Semaan, G. S. & Brito, J. A. M. & Ochi, L. S. (2010). Efficient Algorithms for the capacity and connectivity graph partition problem. Proc. of the CILAMCE 2010 - XXXI Iberian-Latin-American Congress on Computational Methods in Engineering (CD-ROM), Buenos Aires

Semaan, Gustavo S., Cruz, M.D., Brito, Jose André M., and Ochi, Luiz Satoru. (2013) "Proposta de um método de classificação baseado em densidade para a determinação do número ideal de grupos em problemas de clusterização". Learning & Nonlinear Models (L&NLM), Volume 10(4), pp: 242-262.

Silva, A. R. V. & Ochi, L. S. (2010). Hybrid Algorithms for Dynamic Resource-Constrained Project Scheduling Problem**.** To appear in Lecture Notes in Computer Science (LNCS) - Proc. of the 7th International Workshop on Hybrid Metaheuristics (HM2010)., Vienna

Soares, S. S. R. & Ochi, L. S. & Drummond, L. M. (2006). Um algoritmo de construção e busca local para o Problema de Clusterização de Bases de Dados. Tendências de Matemática Aplicada e Computacional, Vol. 7(1), pp. 109-118.

Tsenga, C.L. & Chena,Y.H. & Xua,Y.Y.& Paob, H.T. & Fua, H. (2004) A self-growing probabilistic decision-based neural network with automatic data clustering. Neurocomputing 61. pp. 21 – 38 (neural network)

Tseng, L. Y. & Yang, S. B (2001). A genetic approach to the automatic clustering problem; Pattern Recognition 34, pp. 415-424

Vinod, H. D.,(1969); Integer Programming and Theory of Groups; J. American Statistic Association . Vol 64. pp. 506-519.

Zalik, K. R. (2008). An efficient k´-means clustering algorithm; Pattern Recognition Letters 29, pp. 1385-1391

Zhong, C.& Miao, D. & Wang, R.& Zhou, X. (2008). DIVFRP: An automatic divisive hierarchical clustering method based on the furthest reference points. Pattern Recognition Letters 29. pp. 2067–2077 (divisive hierarquical)

Yin,P. Y. & Chen, L. H. (1994) A new non-iterative approach for clustering, Pattern Recognition Lett., 15 (1994), pp. 125–133.

Yujian, L. (2006). A clustering algorithm based on maximal θ-distant subtrees, Pattern Recognition, pp. 1425-1431

Wang, X. & Qiu, W. & Zamar, R. H. (2007). CLUES: A non-parametric clustering method based on local shrinking. Computational Statistics & Data Analysis 52, pp. 286-298.

Welch, J. W. (1983). Algorithmic complexity: three NP-hard problems in computacional statistics. Journal of Statistical Computation and Simulation 15. pp. 17-25.