Rio de Janeiro, v.7, n.2, p. 130-146 maio a novembro de 2015

A MULTI START HEURISTIC BASED ON GRASP FOR AN AUTOMATIC CLUSTERING PROBLEM

Marcelo Dib Cruz[a*], Luiz Satoru Ochi[b]
[a]*Departamento de Matemática*
Universidade Federal Rural do Rio de Janeiro - UFRRJ
[b]*Instituto de Computação*
Universidade Federal Fluminense - UFF

Abstract

Clustering is the process by which elements of a database are assigned to 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 present a new multi start heuristic based in GRASP for ACP including two local search operators and path relinking procedure. Computational results on a set of instances illustrate the effectiveness and the robustness of the proposed heuristic.

Keywords : Automatic Clustering Problem, Multi start heuristic, GRASP, Path Relinking

***Autor para correspondência:** e-mail: dspereira@id.uff.br

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 known as k-Clustering Problem or Clustering Problem (CP), and if not, as 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 *optimal number of clusters* (Hardy, 1996), or problem of determination of the natural number of clusters (Everitt, 2001).

There are several applications related to clustering (Cruz, 2010), including: Graphs Partitioning, Manufacturing Flexible Problem, Recognition Pattern, Image Processing, Computational Biology, Market Research, Classification of Documents, Data Mining, among others.

The objective of this work is to propose a new multi start heuristic based on GRASP for the ACP, called MSH_ACP. It is not exactly a GRASP heuristic, because the constructive phase of this heuristic does not uses the exactly definitions of the GRASP.

The MSH_ACP has an initial data processing phase to try to reduce the dimensions of the input data of the problem and generate a good initial solution through a constructive procedure. After this first phase, local searches are used, including Path Relinking, whose function is to try to refine the solutions found by the constructive procedure. The algorithm utilizes concepts of adaptive memory using a set of known solutions called *Elite Set* (ES), whose function is to store the best distinct solutions generated by MSH_ACP.

The remainder of this paper is organized as follows: in section 2 a partial literature review is done; In section 3, the ACP is described; in section 4 the MSH_ACP 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.

A large number of clustering algorithms have been presented in the literature. 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.

Metaheuristic algorithms has also been in use in ACP problems. 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 ; Wikaisuksakul,2014), Simulated Annealing (Saha & Bandyopadhyay, 2010 ; Saha & Bandyopadhyay, 2013), Ant Colony (Chowdhury & Das, 2012, Kuo et. al.,2014) , Neural Network (Tsenga et al., 2004 ; Ai-sheng & Qi, 2011), Particle Swarm Optimization (Das et al., 2008; Liu et al.,2014) 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) 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. In (Semaan et al.,2010 ; Mok et al., 2012; Yu H. et. al. ,2014) they use concept of Graph Partitioning .

In (Agraval et al., 2005 ; Wang et al., 2007 ; 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: Let X be a set of n objects $X=\{x_1, x_2, x_3 \dots x_n\}$, where each object x_i is a tuple $(x_{i1},x_{i2},\dots, x_{ip})$, 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 $X=\{C_1,C_2,\dots C_k\}$ 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:

$$C_i \neq \phi,\ for \quad i = 1,..,k \qquad (1)$$

$$C_i \cap C_j = \phi,\ for\ i, j = 1,..,k\ and\ i \neq j \qquad (2)$$

$$\bigcup_{i=1}^{k} C_i = X \qquad (3)$$

Another definition needed in this paper is the concept of centroid of a cluster $C_i = \{x_1, \ldots, x_t\}$. Centroid is the mean position of all the objects in all of the coordinate directions. Let v_i the centroid of C_i and is given by the equation

$$v_i = \frac{1}{t}\sum_{j=1}^{t} x_j \qquad (4)$$

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

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

$$s.t. \qquad C \subset \underline{C} \qquad (6)$$

where $C = \{C_1, C_2, \ldots C_k\}$ is a particular partition of clusters and $\underline{C}$ is a set of all possible partition of the set X, with $k=2,\ldots,n-1$ 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_{ij}. The average similarity of x_i for all points $x_j \in C_w$ is given by $a(x_i)$ where

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

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

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

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

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

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

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

and the Silhouette Index(SI) is

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

4. Methodology

Metaheuristics are general high-level procedures that coordinate simple heuristics and rules to find good quality solutions to computationally difficult combinatorial optimization problems. Among them, we find GRASP. (Resende & Ribeiro,2011)

GRASP, which stands for Greedy Randomized Adaptive Search Procedures is a multi-start, or iterative metaheuristic, in which each iteration consists of two phases: construction and local search. The construction phase builds a solution. If this solution is not feasible, a repair procedure should be applied to attempt to achieve feasibility. If feasibility cannot be reached, it is discarded and a new solution is created. Once a feasible solution is obtained, its neighborhood is investigated until a local minimum is found during the local search phase. The best overall solution is kept as the result.

5. The MSH_ACP Algorithm

The MSH_ACP proposed here is a method composed of two phases: Construction and Local Search. The Construction phase attempts to reduce the dimensions of the input data and generates initial solutions of good quality. These two goals are achieved through a constructive procedure based on concepts of connected components. The Local Search phase of MSH_ACP consists of 3(three) local search procedures including Path Relinking.

5.1 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 three procedures: Generate Partial Clusters (GPC), Merge Partial Cluster (MPC) and Generate_RCL.

```
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.          $N_i = circle(x_i, r)$
9.          $T = T \cup N_i$
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 $C = \{C_1, C_2, \ldots C_t\}$ , 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, $d_{mean}$, 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 ( $\exists x_r \in C_i$ and $\exists x_s \in C_k$ such that $|| x_r - x_s || < d_{adj}$ ) THEN
8.              $C_k = C_k \, U \, C_i$
9.          END IF
10.     END
11. END FOR
12. Let $C = \{C_1, C_2, \ldots C_m\}$ 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 $d_{adj} = v \cdot d_{mean}$ (v is a input parameter). Then, in lines 4-11 there is an aggregation of clusters of small cardinality ($\leq$min points, where min is a input parameter), which 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.

Consider the initial clusters as $C = \{C_1,...,C_m\}$ and v_i, $i = 1,2,...m$ the centroid (equation 4) of the cluster C_i. Let $B= (B_1,B_2,...B_m)$ be an auxiliary binary string of m positions that helps in the generation of a solution for this problem. 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 set B.

The use of adaptive memory in metaheuristics is very promising as seen by (Glover & Kochenberger,2003 ; Pailla et al., 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 solutions that satisfy a diversity index. The ES is used at two different moments : in the middle of the algorithm to perform the local search Path Relinking and to perform local search Exchange pairs at the end of the algorithm.

Before generating an initial solution, a procedure is performed to construct a list RCL (Restrict Candidates List). This list is formed by a subset of the best candidates. The element to be incorporated into the partial solution is randomly selected among those of the RCL. In this work, the list stores the best numbers of the clusters found. For that, several solutions are generated, and the ones with the best Silhouette Index value are selected. But, only the numbers of elements with value 1 in the binary string B, for each best solution, are stored in the RCL. The procedure, called Generate_RCL, uses initially the initial clusters $C = \{C_1,...,C_m\}$ and a binary string $B = (B_1,B_2,...B_m)$.

Then, at each iteration, is generated a new B_i with k elements of value 1, k ranging from 2 to m-1. The k elements with value 1 in the B_i are chosen randomly among the m possible elements. Then, it is generated a solution using the B_i and C. Each solution is evaluated using the Silhouette Index and the values k of the best solution are stored in the RCL, whose values are different. This procedure is repeated maxiter (an input parameter) times. After the generation of the RCL, the algorithm has a good estimative of the optimal number of clusters.

```
 Procedure Generate_RCL (C ,B, MaxIter)
1.  Let  C= {$C_1$,$C_2$,...,$C_m$}  a  set of m initial clusters
2.  Let B= ($B_1$,$B_2$,...$B_m$) a binary string of m positions
3.    FOR i = 1 to  MaxIter  DO
4.            FOR k = 2 to m-1 DO
5.                      $B^1$= Choose_elements_1_randomly(k,$B^0$)
6.                      $s_0$ = Generate Solution ( $B^1$, C)
7.                      Update RCL($s_0$,$B^1$)
8.            END FOR
9.   END FOR
10.   Return  RCL
11. FIM procedure
```

Figure 3: The procedure *Generate_RCL*

The procedure Generate_RCL is presented in Figure 3. In lines 1 and 2 the set of initial clusters C and the auxiliary binary string B are given. Afterwards, an iterative process is run, that at each step, generates a B_i with k elements with value 1, k = 2, .., m-1. Then a solution is generated and evaluated. The identifier k of the best solution is stored in the RCL. This is indicated by the lines 4 and 8. This iterative process is repeated maxiter times as shown in line 3. This procedure returns a RCL list.

The procedures GCP, MCP and Generate_RCL are executed only once. However, to generate an initial solution, at each iteration, an element nc $\in$RCL is chosen. Then, a binary string B is generated with nc elements with value 1. An initial solution is generated using B and C.

5.2. The Local Search Phase

The local search phase needs to be efficient to improve the solutions obtained in the construction phase, especially in Combinatorial Optimization problems, where there are already very efficient constructive heuristics.

The basic idea of this first local search procedure, called Single Inversion (SGI), is to improve the current solution by analyzing solutions close to it. For that, this search switches the value of each element of the set B (1 to 0, or 0 to 1) one at a time, generates a new solution and calculates the new value of the Silhouette Index. But, the algorithm only accepts the change if the new value is better than the previous one.

For example, imagine that the current binary string B is (0101101). Initially, the first element is switched. Then the new B is (1101101). So, a solution is generated and its Silhouette Index is calculated. If this solution has a Silhouette Index value greater than the previous one, then it will be the new current B. Next, the second element is switched. The new B is now (1001101). A new solution is again generated. If this solution has a Silhouette Index value higher than the previous one, then the change is accepted. Otherwise, the previous B is kept. The search ends when all elements of B were 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 binary string B can improve the new solution.

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 one better than the others. The PR consists of tracing the path between a base solution and a target solution , that should be of good quality and evaluate the intermediate solutions obtained along the path. The objective of this search is to find better solutions than the base and target solution.

In this work, the Path Relinking algorithm uses a path from the solution of better quality (B^0) to the solution of lower quality (B^1). The goal of the PR is to insert, at each iteration, an element of the 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 B^1 and exchange by the elements of B^0, if they are distinct, and repeat this procedure for all the elements that compose 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 B^0 is equal to B^1.

Another argument for the use of PR in the ACP is that intermediate solutions with different number of clusters may be obtained when analyzing two different solutions, which is a primary objective of the problem.

The third local search procedure proposed, called Exchange Pair (EP), is an intensive search, that changes the elements with two different values of the solution. For example, suppose that the binary string B is (10111010). Firstly, the first and the second element of B are exchanged. So, the new B is (01111010). If the new solution generated by B improves the value of the Silhouette Index from the old one, 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 were tested.

This local search tries to find different solutions without changing the number of elements with value 1 in B found by the best solutions generated by the algorithm. Due to the high computational time required, this search is done only at the end of the heuristic and for the best solutions of ES.

In short, the MSH_ACP generates an initial solution and applies the local search Single Inversion. At each t iterations, the algorithm also applies the local search Path Relinking. In the end, the algorithm applies the local search Exchange Pairs to the ES.

6. Computational Results

To analyze the performance of the hybrid method *MSH_ACP* here proposed, tests were carried out on computer. The *MSH_ACP* was compared two times. The first was with a set of instances from literature. And the second was compared with two algorithms from the literature, known as *CLUES (*Wang et al., 2007) and MRDBSCAN (Seeman et al.,2012).

The MSH_ACP was implemented in C++ compiler using Ubuntu Linux 7.5 environment. It was used a computer with processor Intel Xeon Quad-core 3.0 Ghz with 16G of RAM to count the processing time.

The MSH_ACP was tested with some instances from literature: RuspiniDataSet (Ruspinil, 2006), Iris Plants Database (Fisher, 1936), MaronnaDataSet (Maronna & Jacovkis, 1974), 200DATA (Fisher, 1936), VowelDataSet (Hastie et al., 2001) e Broken Ring (Wang et al., 2007), Wine data set and Yeast Data Set (Bache. & Lichman,2013). The table 1 shows the instances.

Table1: The instances from Literature

Instance	Number of points	Atributes
RuspiniDataSet	75	2
Íris Plants Database	150	4
MaronnaDataSet	200	2
200DATA	200	2
VowelDataSet	533	2
Broken Ring	800	2
Wine Data Set	178	13
Yeast Data set	1484	7

Other instances were built in (Cruz,2010). 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.

The parameters used in *MSH_ACP* 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 size of the RCL is 7 elements. The number of iterations performed is 35. The maxiter value is equal to 5. The Local search Path Relinking runs four times in iterations 15, 20, 25 and 30. The size of the Elite Set (ES) was set to 5 (five) elements. The choice of the values of these parameters was made after a series of preliminary tests. 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 and is not necessary to specify parameters.

Table2: Comparison between MSH_ACP with and without SGI, PR and EP

		CP + SGI			**CP + SGI + PR**			**CP + SGI + PR+EP**		
Instance	**best**	**IS**	**NC**	**%**	**IS**	**NC**	**%**	**IS**	**NC**	**%**
100p3c	0.7858	0.7858	3	**0,00**	0.7858	3	**0,00**	0.7858	3	**0,00**
2007c1	0,5701	0.5365	5	5,89	0.5532	9	2,96	0,5701	8	**0,00**
300p2c1	0,7764	0.7764	2	**0,00**	0.7764	2	**0,00**	0,7764	2	**0,00**
500p4c1	0,6597	0.6597	3	**0,00**	0.6597	3	**0,00**	0,6597	3	**0,00**
700p15c1	0,6804	0.6489	20	4,63	0.6804	15	**0,00**	0,6804	15	**0,00**
1000p27c1	0.5161	0.4985	26	3,42	0.5112	24	0,95	0.5161	24	**0,00**

Initially, to verify the potential of local search Single Inversion (SGI), Path Relinking(PR) and Exchange Pairs (EP), the MSH_ACP was performed for a limited number of instances with and without the local search, using the same runtime as a stopping criterion . The run time used is the time necessary to perform 35 iterations to the entire MSH_ACP (includes Construction Phase (CP) , SI,PR and EP). The results of the simulations are shown in Table 2. The algorithms were run 5 (five) times for each instance. In this Table, the column Instance shows the name of the instance. The column Best show the best value of Silhouette Index found by the algorithms. The column IS shows 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. The results indicate the importance of including effective procedures for local search in the proposed algorithm. In fact, the inclusion of the PR and EP improved the performance of the algorithm, mainly in the more complicated instances as VowelDataSet and 1000p27c1.

Table3: Comparison between MSH_ACP with and the best values from literature

		MSH_ACP			
Instance	**Best**	**SI**	**t(s)**	**NC**	**%**
RuspiniDataSet	0.738	0.738	0,8	4	**0.00**
Íris Plants Database	0.687	0.686	1,9	2	0.15
MaronnaDataSet	0.575	0.575	2,2	4	**0.00**
200DATA	0.823	0.823	2,8	3	**0.00**
VowelDataSet	0.448	0.418	11,4	3	6.70
Broken Ring	0.500	0.500	25,2	5	**0.00**
Wine Data Set	0.630	0.630	1,3	3	**0.00**
Yeast Data set	0.601	0.601	122,7	2	**0.00**

Table 3 shows the results from the *MSH_ACP* in a set of instances from literature. The results show that the MSH_ACP find the best values in six instances in the total of 8.

Table 4 shows the comparison between the *MSH_ACP*, CLUES and MRDBSCAN. The columns have the same meaning as in Table 2.

Table4: Comparison between MSH_ACP, CLUES and MRDBSCAN

		MSH_ACP			**CLUES**			**MRDBSCAN**		
Instance	**Best**	**SI**	**NC**	**%**	**SI**	**NC**	**%**	**SI**	**NC**	**%**
100p3c	0.7858	0.7858	3	**0.00**	0.7858	3	**0.00**	0.7858	3	**0.00**
100p3c1	0.5966	0.5802	3	2.75	0.5966	3	**0.00**	0.1044	5	82.50
100p5c1	0.7034	0.6958	7	1.08	0.7034	6	**0.00**	0.4235	2	39.79
100p7c	0.8339	0.8338	7	0.01	0.8338	7	0.01	0.8339	7	**0.00**
100p7c1	0.5511	0.4868	27	11.67	0.5511	7	**0.00**	-0.0127	2	102.30
100p10c	0.8336	0.8336	10	**0.00**	0.8336	10	**0.00**	0.6917	8	17.02
200p2c1	0.7642	0.7642	2	**0.00**	0.5912	3	22.64	0.6246	2	18.27
200p3c1	0.6797	0.6797	3	**0.00**	0.674	3	0.84	0.6484	2	4.60
200p4c	0.7725	0.7725	4	**0.00**	0.7725	4	**0.00**	0.7725	4	**0.00**

200p4c1	0.7544	0.7449	4	1.26	0.7544	4	**0.00**	0.6227	3	17.46
200p7c1	0.5701	0.5701	8	**0.00**	0.5553	9	2.60	0.3922	3	31.21
200p12c1	0.5695	0.5695	8	**0.00**	0.5624	9	1.25	0.4033	3	29.18
300p2c1	0.7764	0.7764	2	**0.00**	0.4899	5	36.90	0.6208	4	20.04
300p3c	0.7664	0.7663	3	0.01	0.552	3	27.79	0.7664	3	**0.00**
300p3c1	0.6768	0.6768	3	**0.00**	0.5924	4	12.47	0.6397	2	5.48
300p4c1	0.5924	0.591	2	0.24	0.5924	7	**0.00**	0.269	3	54.59
300p6c1	0.6607	0.6607	8	**0.00**	0.5788	7	12.40	0.5485	2	16.98
400p3c	0.7986	0.7985	3	0.01	0.7985	3	0.01	0.7986	4	**0.00**
400p4c1	0.6204	0.6018	4	3.00	0.6204	4	**0.00**	0.379	2	38.91
400p17c1	0.5524	0.5138	2	6.99	0.5524	15	**0.00**	0.1832	14	66.84
500p3c	0.8249	0.8249	3	**0.00**	0.8249	3	**0.00**	0.8249	3	**0.00**
500p4c1	0.6597	0.6597	3	**0.00**	0.515	3	21.93	0.3054	2	53.71
500p6c1	0.6684	0.6287	6	5.94	0.6684	6	**0.00**	0.4945	12	26.02
600p3c1	0.7209	0.7209	3	**0.00**	0.7028	3	2.51	0.6868	2	4.73
600p15c	0.7812	0.7812	15	**0.00**	0.7526	16	3.66	0.7812	15	**0.00**
700p4c	0.797	0.7969	4	0.01	0.7969	4	0.01	0.797	4	**0.00**
700p15c1	0.6804	0.6804	15	**0.00**	0.6595	17	3.07	0.1227	2	81.97
800p4c1	0.7143	0.7033	4	1.54	0.7143	4	**0.00**	0.5088	2	28.77
800p10c1	0.5071	0.4681	2	7.69	0.5071	8	**0.00**	0.0792	2	84.38
800p18c1	0.6941	0.6914	19	0.39	0.6941	19	**0.00**	0.2655	24	61.75
800p23c	0.7874	0.7873	23	0.01	0.7387	22	6.18	0.7874	23	**0.00**
900p5c	0.716	0.716	5	**0.00**	0.6129	7	14.40	0.716	5	**0.00**
900p12c	0.8409	0.8408	12	0.01	0.7975	10	5.16	0.8409	12	**0.00**
1000p5c1	0.6391	0.6391	5	**0.00**	0.558	7	12.69	0.164	2	74.34
1000p6c	0.7357	0.7356	6	0.01	0.7356	6	**0.00**	0.7357	6	**0.00**
1000p14c	0.8306	0.8306	14	**0.00**	0.7674	11	7.61	0.8085	15	2.66
1000p27c1	0.5631	0.5161	24	8.35	0.5631	14	**0.00**	-0.2934	3	152.10
Average				1.42			5.40			30.99

It was observed that the difference between the average deviation (column %) is large, 1.42 to 5.40 and 30.99, meaning that the MSH_ACP is closer to the best solutions. In addition to have an average better, MSH_ACP also achieves the best values in a larger number of instances. The MSH_ACP reaches the best values in 19 instances, the CLUES reaches the best values in 17 instances and MRDBSCAN reaches the best values in 12 instances, in a total of 37. Table 5 shows the run time of the *MSH_ACP*.

Table5: The time of execution of MSH_ACP

Instance	**t(s)**	**Instance**	**t(s)**	**Instance**	**t(s)**

100p3c	1,8	300p3c	4,2	700p4c	36,4
100p3c1	1,3	300p3c1	4,2	700p15c1	21,8
100p5c1	1,4	300p4c1	4,3	800p4c1	26,8
100p7c	1,3	300p6c1	4,3	800p10c1	28,7
100p7c1	1,2	400p3c	5,6	800p18c1	19,2
100p10c	1,4	400p4c1	4,3	800p23c	27,7
200p2c1	2,1	400p17c1	6,2	900p5c	33,2
200p3c1	2,5	500p3c	6,8	900p12c	47,9
200p4c	2,4	500p4c1	6,7	1000p5c1	55,4
200p4c1	2,4	500p6c1	6,6	1000p6c	47,4
200p7c1	2,3	600p3c1	9,3	1000p14c	63,2
200p12c1	2,3	600p15c	15,2	1000p27c1	74,1
300p2c1	4,1	600p15c	15,2		

One observation to be made is that the more reasonable was also utilizes tests using other stopping criteria such as a timeout, or reaching a target value. However, this was not possible because we don’t have the source code of CLUES and MRDBSCAN, just the results.

7. Conclusions

This paper presents a multi start algorithm MSH_ACP that uses concepts of GRASP and Path Relinking to the ACP. The goal of this research is to develop a method for determining the number of clusters. It is a very difficult problem, because there is a lot of possible solutions. We believe that the use of a construction phase of good quality can improve the performance of the local search phase, since it reduces the input data.

We also believe that local searches used were another important factor for good performance MSH_ACP. Although, increasing the processing time per iteration, we found that MSH_ACP requires a smaller number of iterations to reach a target value.

Finally, the results of MSH_ACP compared with some algorithms of the literature show the efficiency of the algorithm, as best results achieved.

As future work, we will run the algorithm on cases with a larger number of attributes and with others methods from the literature, that uses different functions.

8. References

(1) 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

(2) 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

(3) 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

(4) Bache, K. & Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

(5) 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

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

(7) 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

(8) 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

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

(10) 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

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

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

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

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

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

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

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

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

(19) Kumar V & Chhabra J. & Kumar D. (2014) Automatic cluster evolution using ravitational search algorithm and its application on image segmentation. Engineering applications of Artificial Intelligence 29 . pp. 93–103

(20) Kuo R. J. & , Huang Y. D. & Lin C. c, Wu Y. & Zulvia F. (2014) Automatic kernel clustering with bee colony optimization algorithm. Information Sciences 283. Pp. 107–122

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

(22) Liu R. & Chen Y. & Jiao L. & Li Y. (2014) A particle swarm optimization based simultaneous learning framework for clustering and classification. Pattern Recognition 47. Pp. 2143–2152

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

(24) 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

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

(26) 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.

(27) 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.

(28) 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

(29) 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.

(30) 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.

(31) Resende M. & Ribeiro C. C (2011), "GRASP: Greedy Randomized Adaptive Search Procedures," Search Methodologies, Springer.

(32) 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.

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

(34) 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

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

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

(37) 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

(38) Semaan, G. S. & Cruz, M. D. & Brito, J. A. M. & ; Ochi, L. S. (2012) . Proposta de um Método de Classificação Baseado em Densidade para a Determinação do Número Ideal de Grupos em Problemas de Clusterização. Learning and Nonlinear Models, v. 10, pp. 242-262

(39) 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.

(40) 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

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

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

(43) 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.

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

(45) Yu H. & Liu Z. & Wang, G. (2014) An automatic method to determine the number of clusters using decision-theoretic rough set. International Journal of Approximate Reasoning 55.pp 101–115

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

(47) 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.

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

(49) Wikaisuksakul, S. (2014). A multi-objective genetic algorithm with fuzzy c-means for automaticdata clustering. Applied Soft Computing 24. Pp. 679–691