Revista Brasileira de Computação Aplicada, November, 2020

DOI: 10.5335/rbca.v12i3.11117
Vol. 12, Nº 3, pp. 97–106
Homepage: seer.upf.br/index.php/rbca/index

ISSN 2176-6649

ORIGINAL PAPER

A genetic algorithm using Calinski-Harabasz index for automatic clustering problem

Suzane Pereira Lima[1] and Marcelo Dib Cruz [1]

[1]Federal Rural University of Rio de Janeiro

szlima.93@gmail.com; madibcruz@gmail.com

Received: 2020-06-01. **Revised:** 2020-09-17. **Accepted:** 2020-10-14.

Abstract

Data clustering is a technique that aims to represent a dataset in clusters according to their similarities. 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. This kind of problem is called Automatic Clustering. There are several cluster validity indexes for evaluating solutions and it is known that the quality of a result is influenced by the chosen function. From this, a genetic algorithm is described in this article for the resolution for automatic clustering using the Calinski-Harabasz Index as a form of evaluation. Comparisons between the results and other algorithms in literature are also presented. In a first analysis, fitness values equivalent or higher are found in at least 58% of the cases for each comparison. Our algorithm could also find the correct number of clusters or close values in 33 cases out of 48. In another comparison, some fitness values are lower, even with the correct number of clusters, but graphically the partitioning are adequate. Thus, it is observed that our proposal is justified and that improvements can be studied for cases in which the correct number of clusters is not found.

Keywords: Automatic Clustering Problem; Calinski-Harabasz index; Cluster Validity Index

Resumo

O agrupamento de dados é uma técnica que busca representar um conjunto de dados em grupos de acordo com as suas semelhanças. Algoritmos de agrupamento geralmente assumem que o número de grupos é conhecido. Entretanto, o número ideal de grupos é desconhecido para muitas aplicações. Este tipo de problema é conhecido como Agrupamento Automático. Existem diversas funções para a avaliação de soluções e sabe-se que a qualidade de um resultado é influenciada pela função escolhida. A partir disto, neste artigo é descrito um algoritmo genético para a resolução do agrupamento automático utilizando o índice Calinski-Harabasz como forma de avaliação. Também são apresentadas comparações dos resultados com outros algoritmos da literatura. Numa primeira análise, são encontrados valores de aptidão equivalentes ou maiores em pelo menos 58% dos casos para cada comparação. Consegue-se encontrar o número certo de grupos ou valores próximos em 33 casos de 48. Numa outra comparação, alguns valores de aptidão são inferiores, mesmo com o número de grupos correto, porém graficamente é visto que os particionamentos são adequados. Assim, observa-se que nossa proposta é justificável e aperfeiçoamentos podem ser estudados para os casos onde não é encontrado tal número correto.

Palavras-Chave: Função de Avaliação; Índice CH; Problema de Agrupamento Automático

1 Introduction

Data clustering is a technique that organizes a dataset into clusters defined by the similarities between its elements. Sometimes the number of clusters that represents the set is initially unknown. This case is called Automatic Clustering Problem (ACP), and in addition to identifying the clustering, the ideal number of clusters is part of the solution to be discovered (Linden, 2009, Cruz, 2010, Ochi et al., 2004, José-García

and Gómez-Flores, 2016, Gan et al., 2007).

There are many methods available in the literature. Cruz (2010) proposed several methods for ACP resolution using the Silhouette index (SI) as evaluation criteria. Semaan et al. (2012) presented a method for solving this problem, called MRDBSCAN, evaluating solutions by the SI function. Kettani et al. (2015) aimed to improve the initial definition of the number of clusters problem found in the K-means algorithm. The optimality was measured by Calinski–Harabasz index (CHI). Finally, Pacheco et al. (2017) presented an algorithm based on a proposal inspired by ants behavior to solve data clustering problems. The ACO algorithm performed its experiments with the SI evaluation function.

There are proposals in the literature that make use of the CHI as an evaluation criteria for the resolution of automatic clustering. Because of its simple implementation and low computational cost, it is stated that its use is a good choice so as to find solutions with good clusters formation (Kettani et al., 2015, Harsh and Ball, 2016). Thus, this article presents a Genetic Algorithm to solve the Automatic Clustering Problem. This procedure is based on an algorithm from the literature and the cluster validity index used to evaluate is the CHI. Experiments were performed and their results were compared to other works in the literature.

This paper is structured as follows: The next section presents the cluster validity index that will be applied in the ACP resolution. Section 3 talks about the methodology used in this paper. The fourth section presents, compares and analyzes the results obtained by experiments. Finally, Section 5 presents the conclusions about the work as a whole.

2 The cluster validity index

A solution generated by a clustering algorithm is evaluated by a cluster validity index. It analyzes the goodness of the clustering by establishing, usually, a relation between the clusters' internal cohesion and the separation between clusters. Clustering algorithms based on metaheuristics often use cluster validity indexes as objective function to be optimized (José-García and Gómez-Flores, 2016, Mishra et al., 2016).

Some works present proposals for solving the Automatic Clustering Problem using the CHI as a cluster validity index. It is stated that the use of this index is a good choice in the search for solutions with good clusters formations because, in addition to being simple to implement, its processing is not very computationally expensive. In general, its results are robust when compared to other clusters' validation methods (Kettani et al., 2015, Harsh and Ball, 2016). Thus, the cluster validity index used in this article is the CHI and it will be presented below.

2.1 Calinski–Harabasz index

The function evaluates the cohesion through the sum of distances of cluster elements in relation to their respective centroids. The separation criteria is calculated from the sum of the distances between the centroid of each cluster and the global centroid of the dataset. The computational cost of this function is not high and outperforms, generally, other cluster validity indexes. Its complexity is equals to $O(n)$. When used by a metaheuristic, maximizing its value is the objective. This function is defined as follows (Kettani et al., 2015, Mishra et al., 2016, Maulik and Bandyopadhyay, 2002, Caliński and Harabasz, 1974):

$$CH(C) = \frac{n-k}{k-1}\frac{traceB(C)}{traceW(C)} \quad (1)$$

where:

$$traceB(C) = \sum_{r=1}^{k} |c_r| dist(\bar{c}_r, \bar{x}) \quad (2)$$

$$traceW(C) = \sum_{r=1}^{k}\sum_{i=1}^{|c_r|} dist(x_i, \bar{c}_r) \quad (3)$$

3 The used methodology

In this paper, the ACP is solved by an algorithm based on the Genetic Algorithm metaheuristic. The partitional form is adopted. Using a hard clustering algorithm, the Euclidean Distance determines the similarity between the elements.

The methodology used is based on the method named Constructive Evolutionary Algorithm with Local Search 1 (AECBL1), already known in the literature. The AECBL1 solves the ACP from the concept of the Evolutionary Algorithm metaheuristic. It has two phases, an initial which is responsible for generating the initial clusters and another that presents a genetic algorithm with a local search (Cruz, 2010).

3.1 The formation of initial clusters

For the initial organization of clusters it is made use of the procedure named Generate Initial Solution 1 (GSI1). From the dataset X it organizes the elements into sets to form the genetic algorithm's initial solution. To decrease the cardinality of the problem's input data each of these temporary clusters is considered as an object by the algorithm. Its pseudocode is presented in Algorithm 1 (Cruz, 2010).

A region with points agglomeration originates a cluster. For each element is determined the shortest distance from it to some another. Then, the average of all these shortest distances is calculated, which is called as d_{avg} (Cruz, 2010).

Algorithm 1: GSI1

Input: X, α
begin
  for *i=1 to n* **do**
    $d_{min}(x_i) = \min\{d_e(x_i, x_j)\}, i \neq j, 1 \leq j \leq n$;
  end
  $d_{avg} = \frac{1}{n}\sum_{i=1}^{n} d_{min}(x_i)$;
  $r = \alpha * d_{avg}$;
  for *i=1 to n* **do**
    $N_i = circle(x_i, r)$;
    $T = T \cup N_i$;
  end
  Sort T in descending order;
  $i = 1$;
  while $T \neq \emptyset$ **do**
    $B_i = next(N_j \in T)$;
    $T = T - N_j$;
    $i++$;
  end
  Return $B = \{B_1, \ldots, B_t\}$, the initial t clusters;
end

From this, each element x_i of X is defined as the center of a circle whose radius is equivalent to $r = \alpha * d_{avg}$. Then, the group of elements belonging to each circle $N_i = circle(x_i, r)$ is generated (Cruz, 2010).

A list T stores the number of elements of each circle. It is sorted in descending order according to these cardinalities. Thus, the corresponding elements to each position of T are defined the initial clusters from this procedure, forming $B = \{B_1, B_2, \ldots, B_t\}$. These clusters do not share any elements because when a circle is chosen the elements belonging to it will not be part of any other (Cruz, 2010).

3.2 Genetic Algorithm

The methodology of this work called Genetic Algorithm with Local Search 6 (AGBL6) is based on AECBL1's evolutionary module. It corresponds to an Evolutionary Algorithm with three Local Search techniques and notions of adaptive memory (Cruz, 2010).

After GSI1 processing the algorithm begins its execution by initializing a population. In sequence, G_{max} iterations will be performed related to the Genetic Algorithm's generations. In each generation, individuals are selected for reproduction, crossover and mutation operations. The selected pairs are defined as follows: the first one is chosen among the 50% fittest and the second among the entire population, both selected randomly. There are no repetitions in the choice of individuals. The number of pairs that will pass the crossover is defined according to a rate p_c, it is applied in the two-point method. Then the mutation operator is applied with a probability p_m, it exchanges one of the characters from one of the individuals in the pair. If any generated individual represents an invalid configuration, without clusters, another one is randomly generated to replace it. Remembering that when generating some chromosome, either by applying an operator or during a search process, it is evaluated by CHI, used here as a fitness (Cruz, 2010).

The Individual Inversion local search is applied to the fittest individuals in the population at each t iterations. The Path-Relinking runs every r iterations on the best individual of the population and the best of the Elite set. The Elite set has a size of five individuals and it saves the best solution in each iteration, it must be better than the worst solution in this set and all the others, and at the end of processing the Peer Exchange search runs in this set (Cruz, 2010).

In conclusion, the algorithm returns the best of all solutions after completing the processing of operations. Algorithm 2 shows the method's operation (Cruz, 2010).

Algorithm 2: AGBL6

Input: $X, T_{pop}, G_{max}, p_c, p_m, \alpha, t, r$
begin
  $G = GSI1(X, \alpha)$;
  $P = generateInitialPopulation(G, T_{pop})$;
  for $k = 1$ *to* G_{max} **do**
    for $i = 1$ *to* p_c **do**
      $selection(p_1, p_2)$;
      $crossover(p_1, p_2)$;
      $mutation(p_2, p_m)$;
      $checkIndividuals(p_1, p_2)$;
      $evaluateSolutions(p_1, p_2)$;
    end
    if $k\%t = 0$ **then**
      $individualInversion(P)$;
    end
    if $k\%r = 0$ **then**
      $pathRelinking(P, elite)$;
    end
    $updateEliteSet(P, elite)$;
  end
  $peerExchange(elite)$;
  Return the best solution S;
end

4 Computational experiments

The developed work was implemented in the C++ programming language using the g++ compiler in version 4.8.4.

It used 64 datasets to perform the tests, they are all numerical and they are known in the literature. Each one belongs to one of the following collections: UCI datasets (iris, wine, yeast, glass, thyroid, and breast) (*Clustering basic benchmark*, 2018, *Machine Learning Repository*, 2017), DIM-sets (high) (dim32, dim64, dim128, and dim256) (*Clustering basic benchmark*, 2018), A-sets (a1, a2, and a3) (*Clustering basic benchmark*,

2018), Shape sets (R15 and D31) (*Clustering basic benchmark*, 2018, Veenman et al., 2002), Behaved (It has 17 datasets named according to their size and number of clusters with a letter "c" at the end. Example: 300p4c) (Cruz, 2010), No-behaved (It has 28 datasets named according to their size and approximate number of clusters with a "1" character at the end. Example: 100p5c1) (Cruz, 2010), and Other instances (ruspini, maronna, 200DATA, and Broken ring) (Cruz, 2010, Wang et al., 2007, Fisher, 1936, Maronna and Jacovkis, 1974, Ruspini, 1970). The input parameters were defined empirically in the algorithm (Cruz, 2010).

- α: Tested values vary within the range [0.5,12], according to the particularities of the dataset.
- T_{pop}: The population size was defined as 1/3 of the number of generated initial clusters, ie 1/3 of chromosome size. Its value is limited to a maximum of 30 individuals.
- G_{max}: The algorithm had the number of generations set to 50.
- p_c: The number of pairs of selected individuals for the crossover operation is equivalent to 40% of the population size.
- p_m: The chance of the mutation operation being applied to a selected individual was set at 10%.
- t: It was stipulated to be applied every five iterations. Thus, for 20% of the best individuals in the population the search is performed in the following generations: 5, 10, 15, 20, 25, 30, 35, 40, 45, and 50. The operator is applied similarly to the AECBL1 algorithm, having the same frequency set.
- r: The application of Path-Relinking has been defined for the following generations: 18, 28, 38, and 48. The AECBL1 performs the search this way and it was decided to keep it as well.

The following tables present a comparison between the algorithm presented in this work and some literature proposals, as follows: AECBL1, MRDBSCAN, AK-means, and ACO (Cruz, 2010, Semaan et al., 2012, Kettani et al., 2015, Pacheco et al., 2017). To compare to others methods, it is necessary to normally execute the algorithm, find the solution and use the Silhouette index, since most of the methods in the literature use this index to show their results. So AGBL6 uses the CHI to find its solutions and, in the end, with its final solution, the algorithm uses the SI to compare to others algorithms. In each table the highest results will be highlighted in bold. It will be considered equivalent results those with SI values up to 0.02 difference, so possible rounding of the strategies of other works are disconsidered. Each algorithm from this work was performed 30 times for each instance.

A comparison between AGBL6 and AECBL1, MRDBSCAN and ACO algorithms is shown in Table 1. The “Dataset” column indicates the name of each dataset. The column “AGBL6” includes the results obtained using the CHI function as an evaluation. The “MRDBSCAN”, “AECBL1”, and “ACO” columns contain the results of the respective literature algorithms: MRDBSCAN, AECBL1, and ACO. The “Literature” column corresponds to the known information in the literature. Each dataset has its results presented in three rows. The first, “Number of clusters”, contains the number of clusters obtained in the final solution. The second, “SI value”, includes the SI values for the final solution. The third, “CHI value”, presents the values obtained by the CHI in AGBL6.

Considering the AGBL6, the technique showed good results, often their SI values tied with these works of the literature. In comparison to AECBL1, similar results were found in 28 instances. Lower values were obtained in only 19 cases, and it won once in the broken ring dataset. Although AGBL6 lost in approximately 39% of cases, it obtained results equivalent to those of AECBL1 in about 58% of datasets. Regarding MRDBSCAN algorithm, SI values were tied in 17 instances, AGBL6 won in 25 others, and its results were lower only five times. Thus, the AGBL6 obtained SI values equal to or greater than MRDBSCAN in at least 89% of cases. Considering ACO, there was equivalence in the values in 15 datasets. Higher values were recorded in 16 cases, and eight times AGBL6 resulted in lower results. Thus, the results of AGBL6 tied or won ACO in at least 79% of instances.

About the number of clusters, AGBL6 finds the correct value for 22 instances. AECBL1 resulted in the correct value more times, for a total of 37 datasets. The other two, MRDBSCAN and ACO, found in 14 and 15 situations, respectively. Considering also the discovery of close values, both AGBL6 and ACO obtained a certain degree of similarity to 11 instances. MRDBSCAN values were close to 17 times, and AECBL1 in eight situations.

A comparison between AGBL6 and AK-means algorithms is shown in Table 2. The name of each dataset is displayed in the “Dataset” column. The “AGBL6” column presents the results generated from AGBL6. The results obtained by AK-means are presented in the “AK-means” column. The “Literature” column corresponds to the known information in the literature. Each dataset has its results presented in three rows. The first, “Number of clusters”, contains the number of clusters generated by the solution. The second, “SI value”, displays the values obtained by the Silhouette Index for the final solution. The last, “CHI value”, indicates the values obtained by the respective function in AGBL6.

When comparing the results it is observed that the AGBL6 obtained lower SI values in 12 cases, and tied or won in four.

The AGBL6 generated the correct number of clusters for half of the instances. AK-means hits this value in 13 cases. Considering values close to correct, with a difference of up to two units, AGBL6 generated for two datasets, and AK-means in one case.

Table 1: Results obtained from comparisons between AGBL6, MRDBSCAN, AECBL1 and ACO algorithms. *(to be continued)*

Dataset		AGBL6	MRDBSCAN	AECBL1	ACO	Literature
Maronna	Number of clusters SI value CHI value	2 **0,562172** 274,996	2 **0,562**	4 **0,5745**	2 **0,562**	4
200DATA	Number of clusters SI value CHI value	3 **0,823151** 530,28	3 **0,823**	3 **0,8231**	3 **0,823**	4
Broken ring	Number of clusters SI value CHI value	2 **0,687451** 159,964		5 0,4995		5
100p3c	Number of clusters SI value CHI value	3 **0,785802** 198,116	3 **0,786**	3 **0,7858**	3 **0,786**	3
100p7c	Number of clusters SI value CHI value	7 **0,833863** 186,113	7 **0,834**	7 **0,8338**	7 **0,834**	7
100p10c	Number of clusters SI value CHI value	10 **0,833613** 140,566	8 0,692	10 **0,8336**	10 **0,834**	10
100p2c1	Number of clusters SI value CHI value	2 **0,74274** 257,271	2 **0,743**	2 **0,7427**		2
100p3c1	Number of clusters SI value CHI value	3 **0,580263** 108,262	5 0,104	3 **0,5802**	4 0,133	3
100p5c1	Number of clusters SI value CHI value	5 0,684301 116,551	2 0,423	7 0,6958	17 **0,729**	5
100p7c1	Number of clusters SI value CHI value	2 **0,473892** 135,312	2 -0,013	7 **0,4911**	23 0,326	7
200p4c	Number of clusters SI value CHI value	4 **0,772547** 298,394	4 **0,773**	4 **0,7725**	4 **0,773**	4
200p2c1	Number of clusters SI value CHI value	2 **0,76417** 636,475	6 0,625	2 **0,7642**	2 **0,749**	2
200p3c1	Number of clusters SI value CHI value	2 0,648382 361,114	2 0,648	3 **0,6797**	2 0,648	3
200p4c1	Number of clusters SI value CHI value	4 **0,744936** 280,164	3 0,623	4 **0,7449**		4
200p7c1	Number of clusters SI value CHI value	2 0,53906 301,242	3 0,392	13 **0,5759**	8 0,310	7
200p12c1	Number of clusters SI value CHI value	2 0,524953 298,195	3 0,403	13 **0,5753**	12 0,321	12
300p3c	Number of clusters SI value CHI value	3 **0,766375** 634,296	3 **0,766**	3 **0,7663**	3 **0,766**	3
300p2c1	Number of clusters SI value CHI value	2 **0,77669** 805,498	4 0,621	2 **0,7764**	2 0,758	2
300p3c1	Number of clusters SI value CHI value	2 0,63254 489,531	2 0,640	3 **0,6768**	2 **0,690**	3

Table 1. *Continuation*

Dataset		AGBL6	MRDBSCAN	AECBL1	ACO	Literature
300p4c1	Number of clusters SI value CHI value	2 0,503781 350,574	3 0,269	2 **0,5910**		4
300p6c1	Number of clusters SI value CHI value	2 0,548274 386,583	2 0,549	8 **0,6636**	11 0,577	6
300p13c1	Number of clusters SI value CHI value	2 **0,545101** 342,347	3 0,404	13 **0,5644**	5 0,449	13
400p3c	Number of clusters SI value CHI value	3 **0,798579** 919,626	3 **0,799**	3 **0,7985**	3 **0,799**	3
400p4c1	Number of clusters SI value CHI value	2 0,541096 513,395	2 0,379	4 **0,5989**	2 0,382	4
400p17c1	Number of clusters SI value CHI value	2 **0,513234** 573,552	14 0,183	2 **0,5138**	24 0,193	17
500p3c	Number of clusters SI value CHI value	3 **0,824936** 1454,37	3 **0,825**	3 **0,8249**	3 **0,825**	3
500p4c1	Number of clusters SI value CHI value	2 0,63033 896,475	2 0,305	5 **0,6595**		4
500p6c1	Number of clusters SI value CHI value	2 0,429675 505,718	12 0,495	6 **0,6287**	20 0,557	6
600p15c	Number of clusters SI value CHI value	16 0,747162 437,05	15 **0,781**	15 **0,7812**	15 **0,781**	15
600p3c1	Number of clusters SI value CHI value	3 **0,721168** 1146,35	2 0,687	3 **0,7209**	3 0,661	3
700p4c	Number of clusters SI value CHI value	4 **0,796956** 1181,22	4 **0,797**	4 **0,7969**	4 **0,797**	4
700p15c1	Number of clusters SI value CHI value	2 0,387273 652,843	2 0,123	15 **0,6804**		15
800p23c	Number of clusters SI value CHI value	24 0,763279 492,824	23 **0,787**	23 **0,7873**	20 0,724	23
800p4c1	Number of clusters SI value CHI value	4 **0,70434** 1003,7	2 0,509	4 **0,7021**		4
800p10c1	Number of clusters SI value CHI value	2 **0,467051** 878,184	2 0,079	2 **0,4681**	30 0,092	10
800p18c1	Number of clusters SI value CHI value	2 0,417692 811,169	24 0,266	19 **0,6914**	16 0,628	18
900p5c	Number of clusters SI value CHI value	5 **0,716048** 928,315	5 **0,716**	5 **0,7160**	5 **0,716**	5
900p12c	Number of clusters SI value CHI value	12 **0,840808** 1016,1	12 **0,841**	12 **0,8408**	11 0,818	12

Table 1. *Continuation*

Dataset		AGBL6	MRDBSCAN	AECBL1	ACO	Literature
1000p6c	Number of clusters	6	6	6	5	6
	SI value	**0,73567**	**0,736**	**0,7356**	0,709	
	CHI value	1172,81				
1000p14c	Number of clusters	14	15	14	15	14
	SI value	**0,830566**	0,808	**0,8306**	0,808	
	CHI value	1015,62				
1000p5c1	Number of clusters	4	2	5	11	5
	SI value	0,614515	0,164	**0,6391**	0,586	
	CHI value	853,24				
1000p27c1	Number of clusters	2	3	25	35	27
	SI value	0,477777	-0,293	**0,5186**	0,313	
	CHI value	1211,12				
1100p6c1	Number of clusters	6	5	6	12	6
	SI value	**0,673319**	0,369	**0,6717**	0,618	
	CHI value	1063,59				
1300p17c	Number of clusters	2	18	17		17
	SI value	0,442087	**0,806**	**0,8229**		
	CHI value	1299,85				
1500p6c1	Number of clusters	6	18	6	10	6
	SI value	**0,645448**	0,123	**0,6436**	**0,630**	
	CHI value	1343,79				
1800p22c	Number of clusters	23	23	22		22
	SI value	0,772767	**0,791**	**0,8036**		
	CHI value	1329,69				
2000p11c	Number of clusters	2	11	11	11	11
	SI value	0,516363	**0,713**	**0,7129**	**0,713**	
	CHI value	2603,8				
2000p9c1	Number of clusters	2	2	9	15	9
	SI value	0,501528	0,164	**0,6230**	0,572	
	CHI value	2532,81				

Table 2: Results obtained from comparisons between AGBL6 and AK-means algorithms. *(to be continued)*

Dataset		AGBL6	AK-means	Literature
ruspini	Number of clusters	4	4	4
	SI value	0,737657	**0,9086**	
	CHI value	112,015		
iris	Number of clusters	2	3	3
	SI value	0,686393	**0,7786**	
	CHI value	305,805		
wine	Number of clusters	2	3	3
	SI value	**0,61704**	0,5043	
	CHI value	282,443		
glass	Number of clusters	2	15	7
	SI value	**0,634285**	**0,6514**	
	CHI value	14,1576		
thyroid	Number of clusters	2	3	2
	SI value	0,602425	**0,7773**	
	CHI value	66,4384		
breast	Number of clusters	2	2	2
	SI value	0,593488	**0,7542**	
	CHI value	984,485		
yeast	Number of clusters	2	2	10
	SI value	**0,561033**	0,4102	
	CHI value	87,0216		

Table 2. *Continuation*

Dataset		AGBL6	AK-means	Literature
dim32	Number of clusters SI value CHI value	16 0,945562 1661,72	16 **0,9962**	16
dim64	Number of clusters SI value CHI value	16 0,966338 2379,18	16 **0,9985**	16
dim128	Number of clusters SI value CHI value	16 0,974642 3046,76	16 **0,9991**	16
dim256	Number of clusters SI value CHI value	16 **0,982946** 4356,83	16 **0,9996**	16
R15	Number of clusters SI value CHI value	15 0,752739 439,674	15 **0,9361**	15
D31	Number of clusters SI value CHI value	2 0,393086 2901,52	31 **0,9222**	31
a1	Number of clusters SI value CHI value	2 0,502558 4131,56	20 **0,7892**	20
a2	Number of clusters SI value CHI value	2 0,460388 6061,87	35 **0,7911**	35
a3	Number of clusters SI value CHI value	2 0,377202 6526,36	50 **0,7949**	50

Some AGBL6 results obtained unsatisfactory SI values being lower than other algorithms during the comparisons. Considering AK-means, it was observed that some SI values of their solutions are much higher than those obtained by AGBL6. However, for some datasets, although the AGBL6 SI result is smaller, the correct number of clusters is found. Considering these facts, it was decided to perform a more careful analysis. Some of the resolutions in which the correct number of clusters is found by AGBL6, and the instance dimensions allow for illustration, will be graphically examined. Solutions of four $\mathbb{R}^2$ datasets will be presented below, they are different in types and sizes, as follows: ruspini, R15, 300p2c1, and 1000p6c. Each figure was made by gnuplot program, via command line, and the illustrations represent the organization of the clusters generated for each instance.

Graphically, it is clear that the configuration of clusters for four datasets is adequate, presenting some homogeneity in each cluster and good demarcations between the clusters. Although not obtaining higher silhouette values visually it is concluded that by the correct number of clusters found the generated solutions can be considered optimal, its partitions are correct. The AGBL6 algorithm is a valid strategy but it is not 100% accurate. Therefore, it can be improved to fit cases where the correct number of clusters is not found.

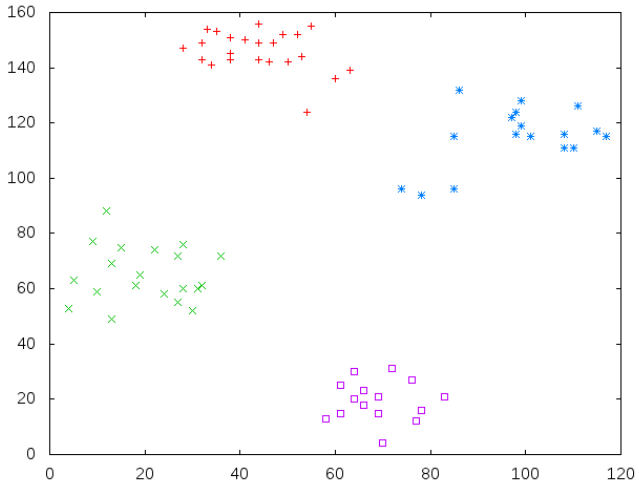

Figure 1: Solution generated for ruspini using AGBL6 algorithm.

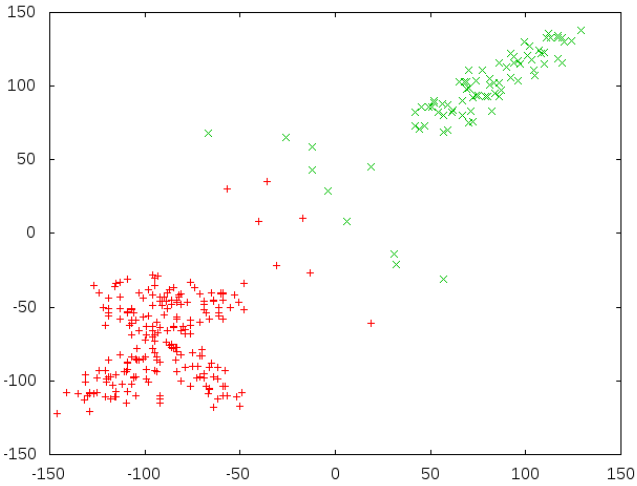

Figure 3: Solution generated for 300p2c1 using AGBL6 algorithm.

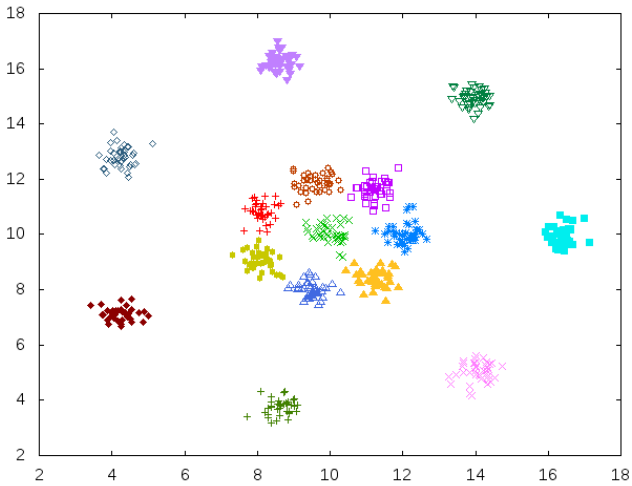

Figure 2: Solution generated for R15 using AGBL6 algorithm.

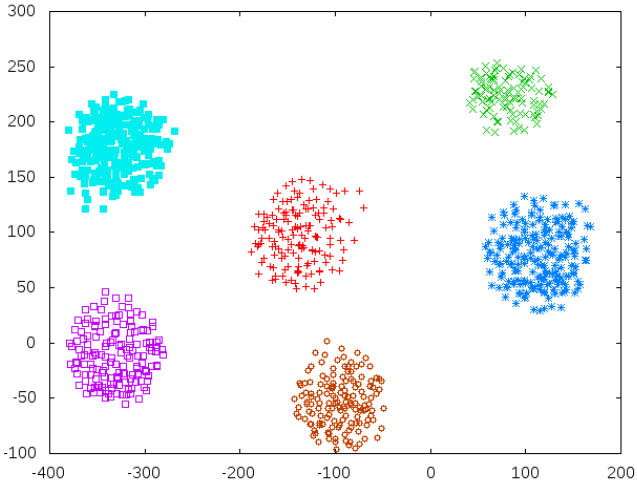

Figure 4: Solution generated for 1000p6c using AGBL6 algorithm.

5 Conclusions and future works

This work presented a methodology for the resolution of Automatic Clustering Problem based on Genetic Algorithm metaheuristic using the Calinski–Harabasz index for the formation of clusters. The results obtained were compared to other works in the literature.

When comparing AGBL6 to other studies, it was found that it was able to result fitness values equivalent to or higher than ACO's and MRDBSCAN's in 79% of the cases and 89%, respectively. There were also ties with AECBL1 for 58% of the datasets. About the number of clusters, in the first comparison to literature works, AGBL6 obtained this value for 22 instances out of 48, second only to AECBL1.

From the comparison to AK-means it was seen that often their fitness values are higher than the results obtained by AGBL6, even when the correct number of clusters was found by this proposal. From a total of 16 datasets in ten times, AGBL6 was able to get the correct number of clusters or values close to it. Thus, a detailed analysis was performed, observing the solutions' results graphically. It was found that the partitions are adequate, they are constituted by clusters of appropriate formation. Thus, it can be concluded that the AGBL6 proposal is justified, however, improvements should be studied to correct cases where the correct number of clusters is not found.

In future activities, the use of a multi-objective

optimization method can be a promising strategy in the search for better solutions. Another possibility that can also be explored is to adopt different cluster validity indexes at different stages of the algorithm. A function can be employed only for the initial phase of clusters formation and another for continuity of processing.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

Caliński, T. and Harabasz, J. (1974). A dendrite method for cluster analysis, *Communications in Statistics-theory and Methods* **3**(1): 1–27. https://doi.org/10.1080/03610927408827101.

Clustering basic benchmark (2018). Available at http://cs.joensuu.fi/sipu/datasets/.

Cruz, M. D. (2010). *O problema de clusterização automática*, PhD thesis, COPPE/UFRJ, Rio de Janeiro. Available at https://www.cos.ufrj.br/uploadfile/1281027685.pdf.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems, *Annals of eugenics* **7**(2): 179–188. https://doi.org/10.1111/j.1469-1809.1936.tb02137.x.

Gan, G., Ma, C. and Wu, J. (2007). *Data clustering: theory, algorithms, and applications*, Vol. 20, Siam. https://doi.org/10.1137/1.9780898718348.

Harsh, A. and Ball, J. E. (2016). Automatic k-expectation maximization (a k-em) algorithm for data mining applications, *Journal of Computations & Modelling* **6**(3): 43–85. Available at http://www.scienpress.com/Upload/JCM/Vol%206_3_3.pdf.

José-García, A. and Gómez-Flores, W. (2016). Automatic clustering using nature-inspired metaheuristics: A survey, *Applied Soft Computing* **41**: 192–213. http://dx.doi.org/10.1016/j.asoc.2015.12.001.

Kettani, O., Ramdani, F. and Tadili, B. (2015). Ak-means: an automatic clustering algorithm based on k-means, *Journal of Advanced Computer Science & Technology* **4**(2): 231. http://dx.doi.org/10.14419/jacst.v4i2.4749.

Linden, R. (2009). Técnicas de agrupamento, *Revista de Sistemas de Informação da FSMA* **4**: 18–36. Available at http://www.fsma.edu.br/si/edicao4/FSMA_SI_2009_2_Tutorial.pdf.

Machine Learning Repository (2017). Available at http://archive.ics.uci.edu/ml/index.php.

Maronna, R. and Jacovkis, P. M. (1974). Multivariate clustering procedures with variable metrics, *Biometrics* pp. 499–505. https://doi.org/10.2307/2529203.

Maulik, U. and Bandyopadhyay, S. (2002). Performance evaluation of some clustering algorithms and validity indices, *IEEE Transactions on pattern analysis and machine intelligence* **24**(12): 1650–1654. https://doi.org/10.1109/TPAMI.2002.1114856.

Mishra, S., Saha, S. and Mondal, S. (2016). A multiobjective optimization based entity matching technique for bibliographic databases, *Expert Systems with Applications* **65**: 100–115. http://dx.doi.org/10.1016/j.eswa.2016.07.043.

Ochi, L. S., Dias, C. R. and Soares, S. S. F. (2004). Clusterização em mineração de dados, *Instituto de Computação-Universidade Federal Fluminense-Niterói* p. 26. Available at https://www.researchgate.net/publication/251910507.

Pacheco, T. M., Brugiolo, L., Ströele, V. and Sã, S. (2017). Metaheurística inspirada no comportamento das formigas aplicada ao problema de agrupamento, *XIII Congresso Brasileiro de Inteligência Computacional*. https://doi.org/10.21528/CBIC2017-133.

Ruspini, E. H. (1970). Numerical methods for fuzzy clustering, *Information Sciences* **2**(3): 319–350. https://doi.org/10.1016/S0020-0255(70)80056-1.

Semaan, G. S., Cruz, M. D., Brito, G. d. M. and 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, *Journal of the Brazilian Computational Intelligence Society* **10**(4): 242–262. http://dx.doi.org/10.21528/lnlm-vol10-no4-art4.

Veenman, C. J., Reinders, M. J. T. and Backer, E. (2002). A maximum variance cluster algorithm, *IEEE Transactions on pattern analysis and machine intelligence* **24**(9): 1273–1280. http://dx.doi.org/10.1109/TPAMI.2002.1033218.

Wang, X., Qiu, W. and Zamar, R. H. (2007). Clues: A non-parametric clustering method based on local shrinking, *Computational Statistics & Data Analysis* **52**(1): 286–298. https://doi.org/10.1016/j.csda.2006.12.016.