RAIRO-Oper. Res. 56 (2022) 3137–3153
https://doi.org/10.1051/ro/2022141

RAIRO Operations Research
www.rairo-ro.org

THE EFFECTIVE BRKGA ALGORITHM FOR THE *k*-MEDOIDS CLUSTERING PROBLEM

Jose Andre Brito[1,*], Gustavo Semaan[2] and Augusto Fadel[3]

Abstract. This paper presents a biased random-key genetic algorithm for k-medoids clustering problem. A novel heuristic operator was implemented and combined with a parallelized local search procedure. Experiments were carried out with fifty literature data sets with small, medium, and large sizes, considering several numbers of clusters, showed that the proposed algorithm outperformed eight other algorithms, for example, the classics PAM and CLARA algorithms. Furthermore, with the results of a linear integer programming formulation, we found that our algorithm obtained the global optimal solutions for most cases and, despite its stochastic nature, presented stability in terms of quality of the solutions obtained and the number of generations required to produce such solutions. In addition, considering the solutions (clusterings) produced by the algorithms, a relative validation index (average silhouette) was applied, where, again, was observed that our method performed well, producing cluster with a good structure.

Mathematics Subject Classification. 90C59, 62H30.

Received May 7, 2022. Accepted August 4, 2022.

1. Introduction

Clustering Analysis (CA) is a tool commonly used in a wide range of applications [12, 13], including Data Reduction, Hypothesis Generation, Business Applications and Market Research, Biology and Bioinformatics, and web mining. According to [13], CA is a multivariate analysis technique comprising a set of algorithms applied to form clusters, based on a data set formed by n objects with f variables, aiming to produce clusters with a high degree of similarity between objects in the same cluster (cohesion) and a low degree of similarity among objects in different clusters (separation) [16].

According to [17], to define the clusters and evaluate the quality of the solutions obtained, an objective function is used as a criterion, which is based in a distance metric, such as the Euclidean distance. The classic clustering problem (CP) is NP-Hard, and obtaining the optimal global solution is a highly complex computational task [10, 17].

Due to the complexity of CP and their varied applications, in recent decades, several algorithms have been developed [1, 26, 30, 37, 44, 51, 52, 56]. In particular, there is a set of heuristic algorithms of more general use in

Keywords. k-Medoids, optimization, metaheuristics, mathematical programming.

[1] National School of Statistical Sciences, Rio de Janeiro, Brazil.
[2] Fluminense Federal University, Rio de Janeiro, Brazil.
[3] Brazilian Institute of Geography and Statistics, Rio de Janeiro, Brazil.
*Corresponding author: jambrito@gmail.com

the literature, divided into two main categories: non-hierarchical and hierarchical [13,19]. Additionally, several mathematical programming formulations for clustering problems are presented in the [17,28,29,34].

About non-hierarchical approaches, there are two classical algorithms based on the prototype model: k-means/k-medoids [13] and PAM (Partitioning Around Medoids) algorithms [20]. According to [15,20], medoids correspond to the (k) most representative items of the given set of objects. Besides, medoid-based algorithms tend to produce higher-quality clusters and more robust to the presence of outliers or noises, and are used in databases whose objects have quantitative and qualitative attributes.

The goal of this article is to tackle the k-medoids clustering problem. Therefore, is proposed a heuristic algorithm that combines concepts of the BRKGA metaheuristics [11,24] with a new crossover operator and a local search procedure. The computational experiments were carried out by applying classic algorithms and the proposed BRKGA-based algorithm to fifty literature data sets, considering six associated scenarios regarding the number of clusters $k \in \{2,3,4,5,6,7\}$. From these experiments, the proposed algorithm produced better solutions to those produced by various algorithms in the literature, such as the classic PAM and CLARA algorithms. In particular, the proposed algorithm produced a high percentage of global optimal solutions. The main contributions of this paper are as follows:

- New algorithm based on the biased random-key genetic algorithms (BRKGA) is proposed to solve the k-medoids clustering problem. It provides a new option for solving this hard clustering problem using BRKGA concepts.
- A novel heuristic operator was implemented and combined with a parallelized local search procedure.
- The scalability and stability of the proposed algorithm observed from experiments carried out with fifty literature data sets, considering different numbers of clusters.
- Analysis of solutions using statistical measures and relative validation index - silhouette.

The outline of this paper is as follows: the Section 2 describes the clustering problem with k-medoids. Section 3 provides a review of the most relevant papers associated with this problem that can be found in the literature. Section 4 presents a description of the BRKGA metaheuristic, and the details of the proposed algorithm – BRKGA k-medoids clustering algorithm – BRKGAMED. Section 5 describes the data sets used in the computational experiments reported in this work, in addition to a discussion on the calibration of the parameters used in the BRKGAMED. In Section 6, results, and analyses from applying the proposed algorithm show its effectiveness in comparison with eight related k-medoids clustering algorithms, in particular, PAM and CLARA heuristics, and an integer programming formulation for the k-medoids clustering problem. Finally, Section 7 contains our summary and discussion.

2. k-MEDOIDS CLUSTERING PROBLEM

Consider a set X formed by n objects $X = \{x_1, \ldots, x_i, \ldots, x_n\}$, such that each x_i is defined by a vector $x_i = (x_i^1, x_i^2, \ldots, x_i^f)$ with f variables. From X, k objects are selected to define medoids used to form k clusters denoted by $C_1, C_2, \ldots, C_k$, so that the following constraints are satisfied:

(i) $|C_r| \geq 1, r = 1, \ldots, k.$
(ii) $\cup_{r=1}^{k} C_r = X.$
(iii) $C_r \cap C_l = \emptyset,\ r, l = 1, \ldots, k, r \neq l.$

The medoids are represented by a set $M = \{m_1, \ldots, m_r, \ldots, m_k\} (M \subset X)$, each element m_r corresponding to the index (i) of the x_i object selected as the medoid of the respective cluster C_r. Additionally, set M is defined so that the sum of the distances of each of the remaining $(n-k)$ objects of X to its nearest medoid is minimum, which is equivalent to minimizing the following objective function:

$$f_{\text{obj}} = \sum_{r=1}^{k} \sum_{\forall x_i \in C_r} d_{m_r i}. \tag{2.1}$$

According to [25], the k-medoids clustering problem is NP-Hard. This characteristic motivates adopting heuristic algorithms that, although they do not guarantee the global optima, tend to produce solutions corresponding to local optima of reasonable quality, demanding low computational time [38]. The application of a brute-force algorithm, which ensures global optima, is infeasible due to the size of the solution space S of this problem, that is, the total subsets of k-medoids, given by:

$$C_n^k = \frac{n!}{(n-k)!k!}. \tag{2.2}$$

For $k \ll n$ this expression grows explosively with k, in general, even faster than a higher-order polynomial. For example, for $n = 200$ and $k = 5$, the number of solutions is in the order of 10^9.

Additionally, this problem can be formulated as an integer linear programming problem and solved by applying exact algorithms such as branch and cut or branch and bound [50]. However, due to the number of variables $(n^2 + n)$ and restrictions $(n^2 + n + 1)$, the resolution might require significant computational time, producing, in many cases, only a local optimum or a feasible solution within this time range.

(K-Medoids) Minimize

$$\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} \tag{2.3}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1, \ldots, n \tag{2.4}$$

$$x_{ij} \leq y_i, \qquad i = 1, \ldots, n, j = 1, \ldots, n \tag{2.5}$$

$$\sum_{i=1}^{n} y_i = k \tag{2.6}$$

$$y_i, x_{ij} \in \{0, 1\}, \qquad i, j = 1, \ldots, n. \tag{2.7}$$

In the formulation above, which was first proposed in [48], y_i is a binary variable that assumes the value 1 if object i $(i = 1, \ldots, n)$ is defined as medoid, and zero otherwise; x_{ij} is also a variable 0–1 that assumes the value 1 if object j is allocated to the cluster defined by medoid i. The objective function in (2.3) aims to minimize the distance of the objects regarding their medoid. Constraint (2.4) ensures that each object j must be associated with a single medoid. Constraint (2.5) ensures that an object j can only be associated with object i if the latter is defined as a medoid. Constraint (2.6) ensures that the number of medoids of the partition is k. In (2.7) we have the integrality constraints.

Additionally, it is important to emphasize that the k-medoids problem is similar to the p-Median problem [5,22], an important optimization problem classified as NP-Hard. Initially addressed by Hakimi [14], this problem corresponds to a classical location problem associated with several real applications. In the p-Median problem, one must determine p facilities (usually called medians) among a set of n candidates to satisfy a specific demand associated with a set of m clients to minimize transport costs and other logistical restrictions. When geographic data are grouped by applying an algorithm to solve the k-medoids problem, this corresponds to solving a basic p-Median problem, where the only objective is to minimize the distances between objects without considering other logistic constraints such as, for example, capacity.

3. Related works

A well-known and commonly used algorithm for this problem is the PAM (Partitioning Around Medoids), proposed in [20]. It determines the k-medoids by applying two procedures called Build and Swap. The authors also proposed another algorithm, called CLARA (Clustering Large Applications), which consists of combining a simple random sampling procedure and the PAM algorithm. In addition to these two algorithms, the next

references bring the most relevant works available in the literature that propose more sophisticated algorithms, some focused on efficiency (speed) and others on effectiveness (quality of solutions). As examples, the algorithm called grouping genetic algorithm (GGA) proposed in [8] and the article by Han and Ng [15], which proposes a modified version of the CLARA algorithm, called CLARANS.

In [47], medoids are defined according to the silhouette maximization. To reduce the computational time required by PAM algorithm and to produce good quality solutions, in [55] it was proposed an algorithm called CLATIN (Clustering Large Applications with Triangular Irregular Network), that uses the concept of the triangular irregular network in the swap procedure of PAM. In [4], PAM algorithm is revisited, and improvements in its swap procedure are proposed. In [32], a fast algorithm that uses the k-means algorithm to define the initial medoids is presented. In [27], Nascimento *et al.* is presented a Lagrangian heuristic for the k-medoids problem.

In [54], the similarity between objects, given by the Euclidean distance, is not used directly, but to order them. Each pair of objects is assigned an integer between 1 and n, representing the order of similarity among them. In each iteration, the medoids are updated to the most dissimilar object in relation to the other objects in the cluster and, once the maximum number of iterations has been reached, each object is allocated to the cluster with the most similar medoid. According to the authors, such strategy can find all gaussian-shaped clusters.

In a more recent study, Yu *et al.* [53] proposed an algorithm that uses a variance measure to determine medoids and focuses on efficiency. In [41], it was proposed a novel parallel k-medoids algorithm, denominated PAMAE, that can be applied to large data sets and achieves both good accuracy and efficiency. In [35, 36], faster versions of the PAM, CLARA, and CLARANS algorithms are proposed, based on improvement of the swap procedure used in PAM algorithm.

In [45, 46], it was proposed a parallel heuristic for a k-medoids clustering problem with variable number of clusters and provided a dual bound for the objective value, thus allowing one to ascertain the optimality of a solution found. In [43], it was proposed a novel fuzzy kernel k-medoids clustering algorithm for uncertain objects which works well on data sets with arbitrary-shaped clusters. In [49], the authors use an efficient method that combines the PAM and CLARA algorithms for image segmentation. In [33], Punhani *et al.* is considered a k-prototype algorithm to generate results like which product is popular among customers and generates more revenue in a particular region. In [6], the authors proposed an algorithm to minimize the number of iterations in k-medoids clustering, where the medoids value was determined by the purity value, and cluster validity was measured with the Davies–Bouldin index.

In addition to these approaches, there are works based on the application of metaheuristics, such as the genetic algorithm proposed by Lucasius *et al.* [23] for large data sets. In [39], a hybrid genetic algorithm called HKA, that combines a crossover operator with a local search procedure based on k-means algorithm. In another correlated study [40] proposed a variant of the genetic algorithm presented in 2004 that solves the k-medoids problem without considering a fixed k value, using for such a combination of a crossover operator with the Davies–Bouldin index. In [18], a hybrid algorithm is proposed that uses the CRO (Chemical Reaction Optimization) algorithm, applied to expand the search for the optimal medoid.

4. BRKGA METAHEURISTIC AND PROPOSED ALGORITHM

The biased random-key genetic algorithms – BRKGA [11, 24] is a metaheuristic that has been applied to several optimization problems [2, 7, 9, 21, 30]. In a BRKGA, the population is composed by p chromosomes that correspond to random key vectors with n real values, generated according to the uniform distribution $[0, 1]$. In each generation of BRKGA, a procedure called decoder, a selection procedure – that corresponds to an elitism strategy, and crossover and mutation operators are applied to each vector of the current population. The decoder is responsible for transforming each of the random key vectors into vectors corresponding to the feasible solutions to the optimization problem. After applying the decoder, the value of the objective function is calculated for the p feasible solutions, and such solutions are then ordered according to its corresponding value (in ascending order, in case of a minimization problem).

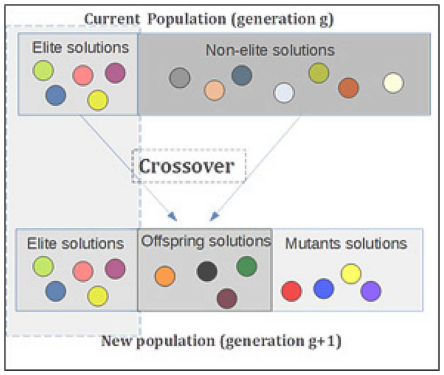

FIGURE 1. Application of BRKGA with the transition between two generations.

In order to apply the selection and crossover operators, the random key vectors associated with the population of the current generation are divided into two sets, namely: an elite set C_E containing the random key vectors corresponding to the p_e best feasible solutions (as per the calculation of the objective function and corresponding ordering of the solutions) and a non-elite set C_{NE} containing the $p - p_e$ remaining random keys vectors. The elitism strategy consists of copying the vectors of C_E to the of the next generation, then the other vectors of the next population are obtained by applying the crossover and mutation operators.

With regards to mutation, p_m random key vectors are generated analogously to the first generation, then these vectors are inserted in the next generation population. Finally, in order to complement the population of the next generation, $p - p_e - p_m$ random keys vectors are produced by applying the uniform crossover proposed by Spears and Jong [42]. For such, it is used, at each crossover execution, a vector of C_E, a vector of C_{NE} and a crossover probability. Figure 1 illustrates two generations followed by the application of the BRKGA.

4.1. BRKGA algorithm for the *k*-medoids clustering problem

The proposed algorithm, called BRKGAMED, uses BRKGA metaheuristic concepts, but differs in terms of representation and generation of population's chromosomes and as regards the use of a new crossover operator. Regarding the representation of chromosomes, p vectors v are generated to compose the first generation (initial population), each vector v defined based on k values randomly selected between 1 and n (number of objects in the data set). These values correspond to the medoids of set M, defined in Section 2, and are used to define the allocation of the other $(n - k)$ objects in data set X to the nearest medoid.

Once the allocation has been made, the objective function of equation (2.1) is calculated, then elite set C_E is defined as the p_e vectors associated with the medoids that produces the lowest values of the objective function, and the remaining $(p - p_e)$ vectors from set C_{NE}.

As in the case of a standard BRKGA, the vectors of set C_E are copied to the population of the next generation and the $(p - p_e)$ remaining vectors are obtained from the application of the crossover and mutation operators. In respect to mutation, p_m vectors are produced analogously to the generation of the initial population. After crossover, the new population will consist of the p_e vectors of set C_E and $(p-p_e)$ vectors produced from applying the mutation and crossover operators described above. Algorithm 1 shows the pseudo-code of the BRKGAMED algorithm. Algorithm 2 and Table 1 bring, respectively, the pseudo-code associated with the crossover operator – which has the function of a local search, where the best improvement strategy is considered – and an example of its application. In Table 1, the vector v_a corresponds to a set of k medoids deriving from C_E, and vector v_b

corresponds to a set of k medoids deriving from C_{NE}. Table 1 shows an example of the application (lines 2–6) of the crossover described in the Algorithm 2, considering $k = 3$ and the vectors $v_a = (2, 5, 10)$ and $v_b = (1, 3, 8)$.

Algorithm 1: Pseudo-code of BRKGAMED algoritm.

```
1  Generate p vectors v with k values between 1 and n (Initial Population)
2  while stopping criteria are not satisfied do
3      Calculate f_obj value for each vector applying equation (2.1)
4      Sort solutions (p vectors) by fitness (Eq. (2.1))
5      Classify solutions as elite and non-elite defining sets C_E and C_NE
6      Copy to next population the p_e vectors (medoids) of the set C_E
7      Generates p_m mutants vectors with k values between 1 and n and copy them to the next population
8      Combine elite and non-elite vector and generate (p − p_m − p_e) vectors to the next population applying
       crossover – Algorithm 2
```

Algorithm 2: Crossover operator.

```
1  f_best ← +∞;
2  for l ← 1 to k do
3      M_a ← Combine each element of v_b with v_a \ v_a[l]
4      M_b ← Combine each element of v_a with v_b \ v_b[l]
5      Add M_a to M and Add M_b to M
6  for l ← 1 to 2k^2 do
7      v_ab ← M[l,] (l-th vector of medoids)
8      Allocate (n − k) objects to the nearest medoid of v_ab defining clusters C_1, ..., C_k
9      for r ← 1 to k do
10         Determine the x_j ∈ C_r whose sum of distances to others (|C_r| − 1) objects of C_r is minimal
11         if x_j ≠ v_ab[r] then v_ab[r] ← x_j
12     Calculate objective function f_obj(v_ab)
13     if f_obj(v_ab) < f_best then
14         f_best ← f_obj(v_ab)
15         v_c ← v_ab
```

TABLE 1. Crossover example.

i	v	Medoids	i	v	Medoids
	v_a	2 5 10		v_b	1 3 8
	v_b	1 3 8		v_a	2 5 10
1	M_a	1 5 10	1	M_b	2 3 8
		3 5 10			5 3 8
		8 5 10			10 3 8
2	M_a	2 1 10	2	M_b	1 2 8
		2 3 10			1 5 8
		2 8 10			1 10 8
3	M_a	2 5 1	3	M_b	1 3 2
		2 5 3			1 3 5
		2 5 8			1 3 10

Upon analysis of the first loop (line 2 – Combination medoids) of Algorithm 2, each of the k elements of v_a is combined with all subsets of v_b formed by $(k-1)$ elements (C_k^{k-1}) and *vice versa*. Therefore, each execution of the procedure in this algorithm produces $2k^2$ ($2 \times k \times C_k^{k-1}$) new chromosomes, of which the one (vector v_c) with the lowest associated value of the objective function (lines 13–15) is selected.

In order to determine the best set of medoids among the $2k^2$ sets (M lines), that is, vector v_c corresponding to the lowest value of objective function, firstly the medoids of each M line are assigned to vector v_{ab}, then the remaining $(n-k)$ $X \setminus v_{ab}$ objects are allocated to the nearest medoid of v_{ab}, thus defining the k clusters $C_1, \ldots, C_r, \ldots, C_k$ (lines 6–8).

Then, it is evaluated for each of the clusters C_r $(r = 1, \ldots, k)$, which object (x_j) that, when defined as medoid, has the lowest sum of distances to the other C_r objects. If $\exists x_j \in C_r$ such that $\forall x_s \in C_r$ $(x_s \neq x_j)$, $x_z \neq v_{ab}[r]$, where $z = \arg\min_{xs} d_{x_j x_s}$, x_j will be the new medoid of cluster C_r and the r-th position of v_{ab} is updated (lines 9–11). This implies testing $|C_r| - 1$ objects by cluster as possible medoids of the cluster.

5. Data sets and parameters calibration

Experiments were carried out with 50 literature data sets to evaluate the performance of BRKGAMED against algorithms from the literature and the formulation described in Section 2. Additionally, in these data sets, the number of objects (n) ranges between 49 and 5000, and the number of variables (f) ranges between 1 and 1213, as shown in Table 2. For the purposes of comparability and reproducibility of the experiments, the R function that implements BRKGAMED and all data sets are available in github.com/jambrito/BRKGAMED.

Table 2. Summary of data sets.

Data set	n	f	Data set	n	f
2-FACE	200	2	IONOSPHERE	351	34
200DATA*	200	2	IRIS	150	4
400P3C	400	2	MARONNA	200	2
A1	3000	2	MORESHAPES*	489	2
AGGREGATION	788	2	NEW-THYROID	215	5
BANKNOTE	1372	5	NORMAL300	300	2
BREASTB_N	49	1213	NUMBERS2	540	2
BROKEN-RING	800	2	OUTLIERS	131	2
BUPA	345	6	PARKINSONS	195	23
CHART*	600	60	PIB_MINAS	853	1
COMPOUND	399	2	PIB100	100	1
CONCRETE_DATA	1030	9	PRIMA_INDIANS	569	8
DBLCA	141	661	RUSPINI	75	2
DBLCB	180	661	SONAR	208	60
DOWJONES	750	4	SPHERICAL_4D3C	400	3
ECOLI	336	7	SPRDESP	645	2
FACE	296	2	SYNTHETIC_CONTROL	600	51
FORESTFIRES	517	7	TRIPADVISOR	980	10
GAMMA400	500	3	UNIFORM400	400	2
GAUSS9*	900	2	UNIFORM700	700	2
GLASS	214	9	VOWEL2	528	2
HABERMAN	306	3	WAVEFORM21	5000	21
HAYES-ROTH	132	6	WDBC*	569	30
INDIAN	583	9	WINE	178	13
INDOCHINA_COMBAT	72	4	YEAST	1484	7

Notes. *Data sets used in the calibration experiments.

A fundamental issue for any metaheuristic to have a reasonable performance concerns determining the values associated with its set of parameters. In the case of the BRKGAMED algorithm, the values of the parameters were defined using as reference the recommendations made in [11] and a preliminary calibration experiment using parameters and values in Table 3.

TABLE 3. BRKGAMED – parameters for calibration experiments.

Parameter	Description	Values
p	Size of population	50, 75, 100
g	Number of generations	25, 50, 100
p_e	Size of elite population	$0.1p$, $0.2p$, $0.3p$
p_m	Size of mutant population	$0.6p$, $0.7p$, $0.8p$
p_c	Crossover probability	0.75, 0.80, 0.85, 0.90
n_g	Generations without improvement	$0.25g$, $0.30g$, $0.35g$

More specifically, five data sets have been selected (marked with an asterisk in Table 2 out of the 50 data sets available, then BRKGAMED was applied 10 times in each data set to $k \in \{3, 4, 5\}$ and considering 972 combinations of the parameters p, g, p_e, p_m, p_c, n_g – accounting to 145 800 executions (number of data sets $\times$ k values $\times$ combinations of parameters $\times$ 10).

Considering each combination of the six parameters above, data sets, and k values, it was calculated the average of the objective function values (Eq. (2.1)) obtained in the 10 executions. Then, taking as the best combination the one corresponding to the largest number of solutions with lowest average values (in all data sets and k values), we have the following combination: $p = 50$, $g = 50$, $p_e = 0.2$ $p = 10$, $p_m = 0.7$ $p = 35$, $p_c = 0.85$ and $n_g = 0.35g$.

6. EXPERIMENTS

This section presents results related to the application of the BRKGAMED, the formulation described in Section 2 and eight algorithms from the literature: PAM and CLARA proposed in [20], FASTPAM, FASTCLARA, FASTCLARANS proposed in [35], HKA algorithm proposed in [39], PARK algorithm proposed in [32] and RANK algorithm proposed in [54]. The authors implemented the BRKGAMED and HKA algorithms using the R programming language, and the other algorithms are available in functions implemented in three R packages, as shown in Table 4. The formulation was implemented using solver GUROBI (version 9.5.1) available in the gurobi package in R. Additionally, all experiments related to applying the nine algorithms and the formulation were carried out on a computer with 16 GB of RAM and AMD FX-6300 six cores 3.50 GHz processor.

TABLE 4. Algorithms and their packages.

Algorithm	Package	Function
PAM, CLARA	cluster	pam, clara
FASTPAM, FASTCLARA, FASTCLARANS	fastkmedoids	fastpam, fastclara, fastclarans
PARK, RANK	kmed	fastkmed, rankkmed

Considering the multicore architecture features of the computer used in the experiments, combined with package parallel available in the R language, which has functions that allow to implement parallelism, the crossover operator was parallelized in the BRKGAMED algorithm. Two experiments were carried to properly present the results of the algorithm. In the first one, presented in Section 6.1, the purpose was to evaluate the effectiveness of the algorithm in achieving reasonable quality solutions for the 50 data sets and different k values. The second experiment, presented in Section 6.2, sought to evaluate the stability of BRKGAMED considering repeated executions of the algorithm for a subset of data sets.

6.1. Experiment I – Analysis of the performance of algorithms

In this experiment, BRKGAMED and eight algorithms from the literature were applied in the fifty literature data sets, considering k ranging from 2 to 7 (300 solutions for each algorithm). The formulation (GUROBI solver) was applied with a maximum running time of 3 h, except for the WAVEFORM21 data set (5000 objects), in which the solver presented an error.

As for choosing the value of k, given the number of data sets involved in the experiment, it would not be feasible to carry out a detailed analysis for each case. An alternative would be to adopt a common practice in the literature on clustering, in which $k \in \{2, \ldots, \lceil\sqrt{n}\rceil\}$ (see [3, 31]), where n is the number of objects in the data set. However, the experiment carried out sought to compare the performance of the methods, regarding the stability and quality of the solutions, in terms of the silhouette index. Once data sets with the number of objects varying between 49 and 5000 were considered and the adoption of an upper bound for k as a function of n would not favor this objective, since there would be no comparability between all the solutions produced, the upper bound for k adopted in the experiment considered the smallest value of n among the data sets used, that is, $k \in \{2, \ldots, \lceil\sqrt{49}\rceil\}$, since, for two distinct data sets A and B, with the number of objects n_A and n_B and being $n_A < n_B, k \in \{2, \ldots, \lceil\sqrt{n_A}\rceil\}$ meets the upper bound suggested in the literature for both data sets.

Besides, using the same k range and the maximum running time of 3 h for the GUROBI solver. The formulation was applied to 49 data sets, except for the WAVEFORM21 data set, consisting of 5000 objects, in which the solver presented an error (out of memory error when running the model). The parameters used in BRKGAMED were defined in a calibration experiment. For the HKA algorithm, were adopted the parameter values defined in [39]. In the algorithms from the literature were considered the default values of the parameters of the functions presented in Table 4.

This experiment made possible to get the objective function values (Eq. (2.1)), the processing times and the allocation of the objects to their respective clusters. The object function values were used to determine, by number of clusters, the following results: (i) percentage of global optimal solutions produced by the algorithms, based on the total of global optimal solutions produced by the formulation within the maximum time of 3 h; (ii) percentage of best solutions – best solution produced considering the nine algorithms and the formulation, not necessarily corresponding to a global optima and (iii) summary statistics calculated based on the relative gaps (Eq. (6.1)) obtained from the difference between the best solution (s_{best}) and the solution produced by each algorithm and the formulation (s_{algf}) – for the fifty data sets (for data set WAVEFORM1, the best solution was considered to be that associated with at least one out of the nine algorithms) and $k \in \{2, 3, 4, 5, 6, 7\}$.

$$\text{gap} = 100 * (s_{algf} - s_{\text{best}})/s_{\text{best}}. \tag{6.1}$$

It is possible to verify, upon analysis of Table 5, the efficacy of BRKGAMED against other algorithms, considering the percentage of global optimal solutions produced. For the number of clusters equal to 3, BRKGAMED achieved the optimal in 100% of cases. Moreover, the lowest percentages of this algorithm, almost around 88%, occurred for $k = 6$ clusters. Additionally, the PAM, PAMF, and HKA algorithms presented the closest percentages of global optimal, with respect to BRKGAMED. The most favorable scenario for these three algorithms occurred for $k = 2$, when the differences were, in percentage points, respectively, of 2.1% (HKA) and 14.6% (PAM and PAMF). For $k \in \{3, 4, 5, 6, 7\}$, such differences varied between 29.2% (HKA, $k = 3$) and 72.9% (HKA, $k = 7$).

Additionally, the worst results are associated with the CLARAF and CLARANSF algorithms, with percentages of global optimal below 7% and even 0% $k = 6$ and $k = 7$. The RANK algorithm failed to produce the global optimum for all data sets and number of clusters.

Upon analysis of Table 6, associated with gaps between the solutions (except for the maximum gap), BRKGAMED algorithm generally produced gaps values of 0%. In this table, cells highlighted with shades of gray (from the lightest to the darkest) and in italics/underline correspond, respectively, to gap values with mean, median (Md) and 3rd quartile (Q_3) within the following ranges $[0, 0.1\%]$, $(0.1\%, 0.5\%]$, $(0.5\%, 1.0\%]$ and $(1.0\%, 5.0\%]$. In particular, upon analysis of the mean gap, the worst BRKGAMED result was 0.5%, for $k = 6$.

TABLE 5. Percentage of global optimum by algorithm and number of clusters.

Algorithms	k					
	2	3	4	5	6	7
BRKGAMED	93.8	100.0	95.8	95.8	87.5	89.6
CLARA	8.3	2.1	4.2	2.1	4.2	2.1
CLARAF	22.9	18.8	6.3	6.3	0.0	0.0
CLARANSF	6.3	4.2	6.3	2.1	0.0	0.0
HKA	91.7	70.8	62.5	43.8	27.1	16.7
PAM	79.2	72.9	70.8	58.3	47.9	52.1
PAMF	79.2	70.8	70.8	54.2	47.9	52.1
PARK	33.3	14.6	14.6	6.3	2.1	0.0
RANK	0.0	0.0	0.0	0.0	0.0	0.0
Nglobal*	48	48	48	48	48	48

Notes. *Number of global optimal produced by the formulation – cpu time of 3 h, considering the 49 data sets. Global optimum not obtained for A1 data set and for the WAVEFORM21 data set, solver presented an error.

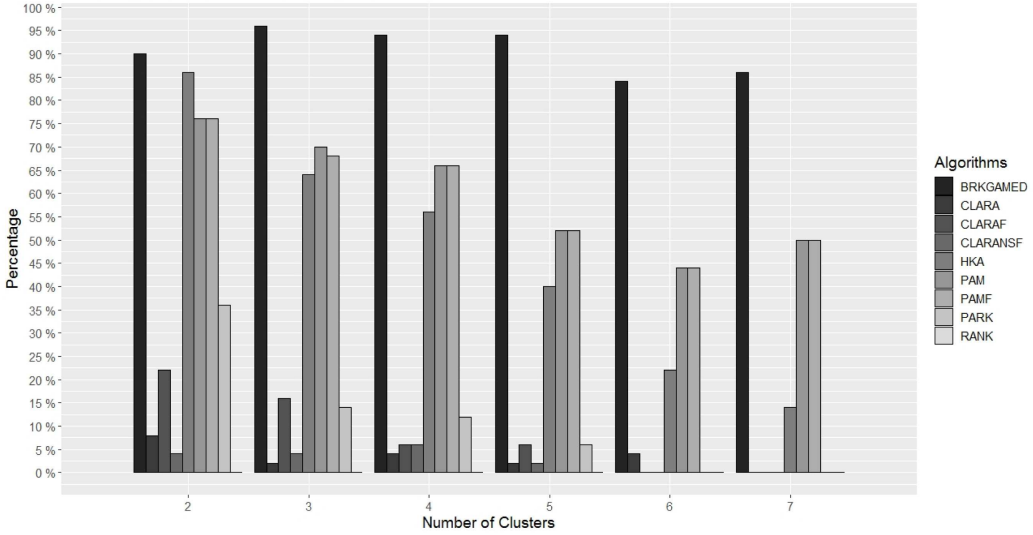

FIGURE 2. Percentage of best solutions by algorithms and number of clusters.

Based on such cells, PAM and PAMF algorithms present the closest gaps in relation to BRKGAMED (less than 1.0%), followed by the HKA algorithm with gaps of up to 1.0% for k between 2 and 5, and gaps of up to 2.0% for $k = 6$ and 7.

CLARA, CLARANSF, PARK, and RANK algorithms have the largest gaps. In particular, RANK algorithm presented the worst results regardless of the number of clusters, with gaps varied between 29% and 47% in the mean gap and between 20.6%, and 26.9% in the median gap.

To conclude the analyses associated with the solutions, Figure 2 shows the percentages of best solutions produced by algorithm *versus* number of clusters, where, once again, BRKGAMED algorithm significantly outperformed the other ones, with percentages between 88% and 100%, followed by the PAM, PAMF and HKA algorithms.

TABLE 6. Relative gaps by algorithm and number of clusters.

Algorithms	$k=2$						$k=3$					
	Min	Q_1	Md	$Mean$	Q_3	Max	Min	Q_1	Md	$Mean$	Q_3	Max
BRKGAMED	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0
CLARA	0.0	0.5	1.4	1.7	2.6	7.4	0.0	1.6	2.2	3.0	4.1	15.6
CLARAF	0.0	0.0	0.2	0.5	0.6	2.9	0.0	0.1	0.4	1.0	0.8	10.8
CLARANSF	0.0	0.4	1.2	1.4	2.0	4.5	0.0	0.7	1.5	2.3	2.5	18.2
HKA	0.0	0.0	0.0	0.0	0.0	1.3	0.0	0.0	0.0	0.1	0.1	1.7
PAM	0.0	0.0	0.0	0.3	0.0	9.6	0.0	0.0	0.0	0.2	0.0	1.6
PAMF	0.0	0.0	0.0	0.3	0.0	9.6	0.0	0.0	0.0	0.3	0.1	5.7
PARK	0.0	0.0	0.4	2.5	1.9	42.5	0.0	0.1	1.1	3.5	3.8	36.5
RANK	0.2	10.8	20.6	29.0	36.9	175.7	0.5	11.0	24.5	33.5	41.0	316.5
	$k=4$						$k=5$					
BRKGAMED	0.0	0.0	0.0	0.0	0.0	1.5	0.0	0.0	0.0	0.1	0.0	5.0
CLARA	0.0	1.7	4.0	4.1	5.6	24.2	0.0	3.1	4.8	5.3	5.7	49.6
CLARAF	0.0	0.3	0.8	1.5	1.6	12.8	0.0	0.4	1.1	1.5	2.2	6.1
CLARANSF	0.0	0.7	1.9	3.0	3.5	27.2	0.0	1.6	2.5	3.9	3.8	34.8
HKA	0.0	0.0	0.0	0.3	0.2	3.6	0.0	0.0	0.1	0.9	0.5	26.5
PAM	0.0	0.0	0.0	0.3	0.2	3.5	0.0	0.0	0.0	0.3	0.3	1.6
PAMF	0.0	0.0	0.0	0.3	0.2	3.5	0.0	0.0	0.0	0.3	0.2	3.4
PARK	0.0	0.4	1.8	5.9	4.9	78.1	0.0	1.5	4.7	7.6	7.4	76.0
RANK	2.2	12.8	23.2	36.5	48.6	146.8	2.2	14.3	26.9	47.0	66.0	289.0
	$k=6$						$k=7$					
BRKGAMED	0.0	0.0	0.0	0.5	0.0	23.1	0.0	0.0	0.0	0.4	0.0	18.8
CLARA	0.0	3.5	5.8	7.2	7.4	85.7	0.0	3.9	6.3	6.8	8.9	44.8
CLARAF	0.1	0.9	1.4	2.0	2.6	10.9	0.1	0.8	1.8	2.5	2.9	18.3
CLARANSF	0.1	2.0	3.2	4.5	4.3	41.4	0.2	1.3	2.8	5.0	4.3	63.9
HKA	0.0	0.0	0.2	1.4	1.0	39.7	0.0	0.2	0.5	2.0	1.8	50.9
PAM	0.0	0.0	0.0	0.4	0.5	2.7	0.0	0.0	0.0	0.5	0.6	4.6
PAMF	0.0	0.0	0.0	0.4	0.4	2.7	0.0	0.0	0.0	0.5	0.6	4.6
PARK	0.0	2.9	4.1	9.0	6.6	128.5	0.2	3.2	6.2	13.6	9.6	162.5
RANK	2.3	13.8	26.4	43.6	61.7	188.4	2.2	12.1	24.9	41.8	48.2	281.7

Complementing the results produced by BRKGAMED, it was performed a comparative analysis of the processing times required by BRKGAMED, HKA, and the mathematical formulation. It is noteworthy that both algorithms are evolutionary [24], therefore, they work with populations (sets of solutions) and combination, mutation, and elitism operators. Such approach requires intensive computation and requires more processing time in the search for good quality solutions.

The other algorithms considered, such as PAM and CLARA, are fast (of the order of up to 5 s per data set), although they produced a smaller number of global optimal and best solutions when compared to the BRKGA algorithm, as shown in Table 5, and Figure 2. Table 7 shows the mean and median associated with the processing times (in seconds) required by BRKGAMED, HKA, and the formulation. In general, BRKGAMED presented lower values than those demanded by HKA and the formulation. Compared to HKA, BRKGAMED was up to 30 times faster (median and $k = 4$).

Another way to evaluate the quality of solutions produced by clustering algorithms generally concerns the application of an index associated with the relative validation criterion. In this work, it was used the average silhouette, which, according to [20], allows to evaluate how proper is the allocation of each object in its cluster, regarding the distance to all other objects in the data set. Figure 3 shows the proportions of best average silhouettes (in relation to the 50 data sets), by number of clusters and by algorithm, associated with the

TABLE 7. Average and median computational time by number of clusters (seconds).

k	Mean			Median		
	BRKGAMED	HKA	Model	BRKGAMED	HKA	Model
2	26	203	281	7	147	25
3	26	203	298	7	147	23
4	29	416	323	11	327	16
5	36	413	318	15	330	18
6	47	446	308	24	371	19
7	60	491	310	29	388	19

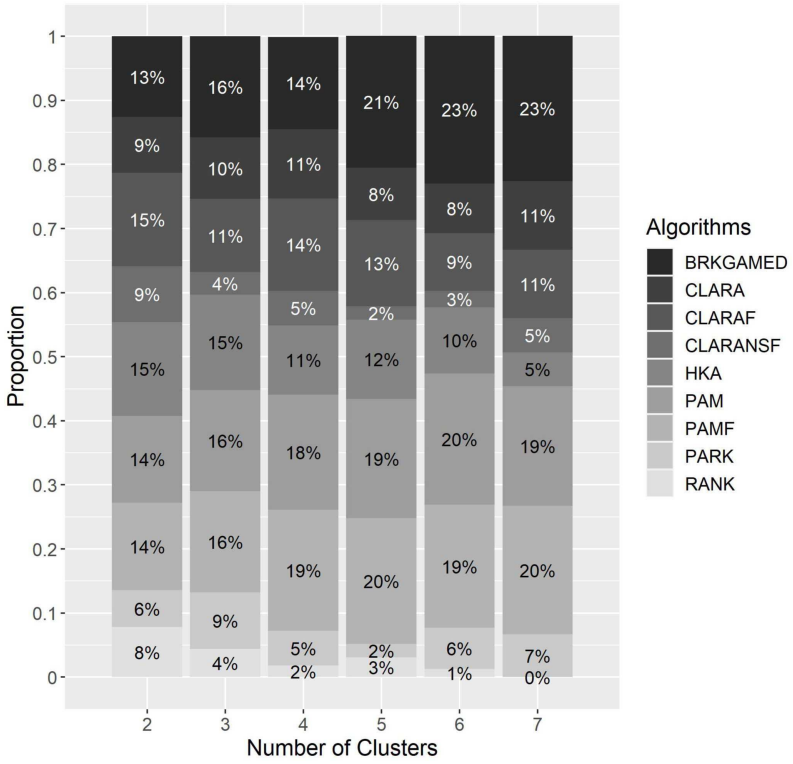

FIGURE 3. Proportion of the Best Average Silhoutte produced by algorithms.

highest values of average silhouette. So, in terms of proportion, it is observed that BRKGAMED performed reasonably.

6.2. Experiment II – BRKGAMED stability

To evaluate BRKGAMED stability, the algorithm with the highest percentages of best solutions, a second experiment was carried out with a subset of the data sets in Table 1. More specifically, considering ten data sets and $k \in \{3, 4, 5, 6\}$ (a total of 40 scenarios), the algorithm was applied 50 times in each of the data sets presented in Table 8 and, in each execution, both the objective function value associated with the best solution and the total number of generations required to produce such solution were stored.

TABLE 8. Statistics obtained from objective function values.

Data sets	$k=3$					$k=4$				
	s_{best}	Min	$Mean$	$cv\%$	Max	s_{best}	Min	$Mean$	$cv\%$	Max
BANKNOTE	1.242	1.242	1.242	**0.000**	1.242	1.112	1.112	1.112	**0.000**	1.112
BROKEN-RING	0.804	0.804	0.804	**0.000**	0.804	0.609	0.609	0.609	**0.000**	0.609
BUPA	1.762	1.762	1.762	**0.000**	1.762	1.660	1.660	1.660	0.051	1.665
CONCRETE_DATA	2.472	2.472	2.472	**0.000**	2.472	2.299	2.299	2.300	0.018	2.301
FORESTFIRES	1.850	1.850	1.850	**0.000**	1.850	1.731	1.731	1.731	0.016	1.733
HABERMAN	1.127	1.127	1.127	**0.000**	1.127	0.985	0.985	0.985	**0.000**	0.985
IOSNOSPHERE	4.217	4.217	4.217	**0.000**	4.217	3.961	3.961	3.961	**0.000**	3.961
NEW-THYROID	1.214	1.214	1.214	**0.000**	1.214	1.077	1.077	1.077	**0.000**	1.077
NUMBERS2	0.780	0.780	0.780	0.023	0.781	0.602	0.602	0.602	**0.000**	0.602
WDBC	3.991	3.991	3.991	**0.000**	3.991	3.840	3.840	3.841	0.042	3.848
	$k=6$					$k=7$				
BANKNOTE	1.010	1.010	1.010	0.030	1.012	0.938	0.938	0.938	0.087	0.940
BROKEN-RING	0.515	0.515	0.515	**0.000**	0.515	0.468	0.468	0.468	0.018	0.469
BUPA	1.572	1.572	1.573	0.118	1.579	1.515	1.515	1.517	0.193	1.525
CONCRETE_DATA	2.176	2.176	2.176	0.069	2.186	2.061	2.061	2.061	0.049	2.066
FORESTFIRES	1.658	1.659	1.660	0.177	1.669	1.59	1.590	1.591	0.164	1.600
HABERMAN	0.890	0.890	0.890	0.043	0.892	0.831	0.831	0.831	0.064	0.834
IOSNOSPHERE	3.808	3.808	3.808	0.036	3.813	3.665	3.663	3.664	0.018	3.665
NEW-THYROID	0.984	0.984	0.984	**0.000**	0.984	0.925	0.926	0.926	0.048	0.928
NUMBERS2	0.525	0.525	0.525	0.006	0.526	0.448	0.448	0.448	**0.000**	0.448
WDBC	3.719	3.715	3.717	0.053	3.721	3.601	3.600	3.601	0.009	3.601

TABLE 9. Statistics associated with total generations.

Data sets	$k=3$				$k=4$			
	Min	$Mean$	Md	Max	Min	$Mean$	Md	Max
BANKNOTE	2	6	5	14	3	11	10	41
BROKEN-RING	4	8	8	23	3	6	6	18
BUPA	2	6	6	14	2	12	10	47
CONCRETE_DATA	2	9	8	27	3	8	7	19
FORESTFIRES	2	4	4	9	3	8	7	27
HABERMAN	3	10	10	28	3	6	6	13
IOSNOSPHERE	2	4	4	13	3	9	8	23
NEW-THYROID	1	3	3	5	2	5	4	8
NUMBERS2	2	13	12	40	2	6	6	12
WDBC	2	6	5	19	2	8	7	23
	$k=5$				$k=6$			
BANKNOTE	4	9	8	32	4	9	9	25
BROKEN-RING	2	7	6	16	4	10	9	25
BUPA	3	14	13	36	3	15	14	36
CONCRETE_DATA	4	12	11	23	6	14	13	27
FORESTFIRES	3	11	10	36	5	17	17	45
HABERMAN	3	10	8	29	4	15	12	38
IOSNOSPHERE	3	8	7	30	3	7	6	23
NEW-THYROID	2	6	5	9	3	8	7	30
NUMBERS2	3	11	9	30	3	6	6	10
WDBC	4	16	12	43	4	14	14	31

Notes. *Results obtained from the 50 BRKGAMED executions.

In Table 8, s_{best} corresponds to the value of the objective function (according to equation (2.1) obtained considering Experiment I and the other columns show the statistics associated with the values of objective function obtained from the 50 executions of BRKGAMED, namely: minimum (Min), mean ($Mean$), and maximum (Max), in addition to the coefficient of variation (cv) in percentage values. It is possible to verify, upon analysis of said table that, in 35 out of the 40 scenarios evaluated (87.5%), the minimum value (Min) obtained for the objective function was equal to the value of the solution produced by BRKGAMED in Experiment I (s_{best}). In addition, the cv was equal to zero in 50% of the scenarios and was less than 0.20% in the remaining cases. It means that, in most runs, the algorithm produced the same solution, which also corresponds to the best solution, since the mean solution, in most cases, was equal to the minimum solution.

Complementing the analyses from this experiment, Table 9 presents, for the same scenarios, the minimum (Min), mean ($Mean$), median (Md), and maximum (Max) values obtained from the total number of generations demanded by BRKGAMED to produce the best solution in each execution. Considering that the parameters associated with the maximum number of generations (g) and the number of generations without improvement (n_g) were defined, respectively, as 50 and 18. It can be seen that, in most cases, it took a few generations for BRKGAMED to produce good quality solutions. Considering the mean and median values – gray cells (scenarios where the algorithm reached the best solution within 15 generations), it is possible to verify that, in most scenarios, BRKGAMED required 30% of the total number of generations to achieve good quality solutions. In addition, in about half of the cases, the maximum number of generations was around 25 (50% of the total number of generations).

7. Conclusions and future work

In this work, we presented an algorithm (BRKGAMED) to solve the k-medoids clustering problem, which is NP-Hard. This algorithm combined BRKGA metaheuristics concepts with a new proposed crossover operator, which incorporates a local search procedure. To evaluate the performance of this algorithm, we performed several experiments with fifty data sets of varying sizes, comparing our algorithm with well-known PAM and CLARA, its variants, and other clustering algorithms proposed for this same problem. Additionally, an integer programming formulation was applied to solve this problem – which allowed the evaluation of the percentage of global optimal solutions produced by the algorithm.

The BRKGAMED, the algorithms from the literature, and the formulation were applied to such data sets to produce solutions with number of clusters ranging from 2 to 7, where were evaluated percentages of global optimal solutions, percentages of best solutions, relative gaps, and average silhouette.

Regarding the global optimal solutions, BRKGAMED produced, in general, percentages above 90%. For $k = 3$ it was obtained 100% global optimal solutions, and the lowest percentage observed was in the order of 88% ($k = 6$). Besides were observed percentages between 88% and 100% (Fig. 2), while evaluating the best quality (or winning) solutions. In this sense, from the global optimal and the best solutions, the results obtained in these experiments showed that BRKGAMED consistently outperformed other algorithms, including PAM and HKA algorithms.

However, when evaluating the gaps in Table 6, it is observed that the BRKGA algorithm had, on average, better performance than the other algorithms for $k \leq 5$. But, for $k = 6$ and $k = 7$, the average gaps between the BRKGA algorithm and the PAM algorithm are very close, with a slight superiority of the PAM algorithm for $k = 6$. Finally, BRKGAMED produced average silhouettes of reasonable quality when compared to other algorithms, as shown in Figure 3.

In the second experiment, where BRKGAMED was applied 50 times to a subset of 10 data sets, it was possible to observe the algorithm stability regarding the quality of the solutions produced and the number of generations required to produce such solutions. This statement is corroborated by the mean and the low values of the coefficient of variation of the objective function values (Tab. 8). Thus, based on the results presented in this article, considering experiments involving data sets of varying size, it was possible to verify the efficacy of

BRKGAMED against other algorithms found in the literature, which indicates that this algorithm is a relevant alternative to be considered when solving the problem of k-medoids.

As future work, we plan to develop a new crossover operator incorporating a local search procedure, based on the VNS metaheuristic and Path Relinking procedure, to produce reasonable quality solutions, demanding less processing time. Another possibility is to solve the k-medoids problem without previously defining, the number of clusters, which characterizes the automatic clustering problem. To attain this goal, BRKGAMED can be adapted using the average silhouette combined with the objective function, so as to define the ideal number of clusters.

Acknowledgements. The researchers were partly funded by the CNPq grant number 405044/2021-6, and PROPPI/UFF (FOPESQ).

References

[1] D. Aloise and C. Contardo, A new global optimization algorithm for diameter minimization clustering. In: Proceedings of Global Optimization Workshop (GOW16), Portugal, edited by D. Aloise. University of Minho/Algoritmi Research Centre. (2016) 171–174.

[2] J.A.M. Brito, T.M. Veiga and P.L.N. Silva, An optimization algorithm applied to the one-dimensional stratification problem. *Surv. Methodol.* **45** (2019) 295–315.

[3] R. Campello, E.R. Hruschka and V. Alves, On the efficiency of evolutionary fuzzy clustering. *J. Heuristics* **15** (2009) 43–75.

[4] S. Chu, J.F. Roddick and J. Pan, Improved search strategies and extensions to k-medoids-based clustering algorithms. *Int. J. Bus. Intell. Data Min.* **3** (2008) 212–231.

[5] M. Daskin, Network and Discrete Location: Models, Algorithms, and Applications, 2nd edition. John Wiley & Sons (2013).

[6] R. Dinata, S. Retno and N. Hasdyna, Minimization of the number of iterations in k-medoids clustering with purity algorithm. *Rev. Intell. Artif.* **35** (2021) 193–199.

[7] A.C. Fadel, L.S. Ochi, J.A.M. Brito and G.S. Semaan, Microaggregation heuristic applied to statistical disclosure control. *Inf. Sci.* **548** (2021) 37–55.

[8] E. Falkenauer, Genetic Algorithms and Grouping Problems. John Wiley & Sons (1998).

[9] P. Festa, A biased random-key genetic algorithm for data clustering. *Math. Biosci.* **245** (2013) 76–85.

[10] M. Garey and D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company (1979).

[11] J. Gonçalves and M. Resende, Biased random-key genetic algorithms for combinatorial optimization. *J. Heuristics* **17** (2011) 487–525.

[12] D. Gunopulos, Clustering overview and applications. In: Encyclopedia of Database Systems, edited by L. Liu and M.T. Özsu. Springer, Boston, MA (2009).

[13] J. Hair, W. Black, B. Babin and R. Anderson, Multivariate Data Analysis, 8th edition. Cengage Learning (2018).

[14] S. Hakimi, Optimum location of switching centers and the absolute centers and medians of a graph. *Oper. Res.* **12** (1964) 450–459.

[15] J. Han and R. Ng, Clarans: a method for clustering objects for spatial datamining. *IEEE Trans. Knowl. Data Eng.* **14** (2002) 1003–1016.

[16] J. Han, J. Pei and H. Tong, Data Mining: Concepts and Techniques, 4th edition. Morgan Kaufmann Publishers (2022).

[17] P. Hansen and B. Jaumard, Cluster analysis and mathematical programming. *Math. Program.* **79** (1997) 191–215.

[18] A. Hudaib, M. Khanafseh and O. Surakhi, An improved version of k-medoid algorithm using CRO. *Modern Appl. Sci.* **12** (2018) 116–127.

[19] R.A. Johnson and W.D. Wichern, Applied Multivariate Statistical Analysis, 6th edition. Pearson (2018).

[20] L. Kaufman and P. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis. Wiley-Interscience (1990).

[21] M. Kong, J. Pei, H. Cheng and P. Pardalos, A BRKGA-de algorithm for parallel-batching scheduling with deterioration and learning effects on parallel machines under preventive maintenance consideration. *Ann. Math. Artif. Intell.* **88** (2020) 237–267.

[22] G. Laporte, S. Nickel and F. da Gama, Location Science, 2nd edition. Springer (2019).

[23] C. Lucasius, A. Dane and G. Kateman, On k-medoid clustering of large data sets with the aid of a genetic algorithm: background, feasibility and comparison. *Anal. Chim. Acta* **282** (1993) 647–669.

[24] R. Martí, P. Pardalos and M. Resende, Handbook of Heuristics, 1st edition. Springer (2018).

[25] N. Megiddo and K. Supowit, On the complexity of some common geometric location problems. *SIAM J. Comput.* **13** (1984) 182–196.

[26] M. Nascimento, F. Toledo and A. de Carvalho, Investigation of a new grasp- based clustering algorithm applied to biological data. *Comput. Oper. Res.* **37** (2010) 1381–1388.

[27] M. Nascimento, F. Toledo and A. Carvalho, A hybrid heuristic for the k-medoids clustering problem. In: GECCO'12: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation. Association for Computing Machinery, New York, NY, USA (2012) 417–424.

[28] M. Negreiros and A. Palhano, The capacitated centred clustering problem. *Comput. Oper. Res.* **33** (2006) 1639–1663.

[29] M. Negreiros, N. Maculan, P. Batista, J. Rodrigues and A. Palhano, Capacitated clustering problems applied to the layout of it-teams in software factories. *Ann. Oper. Res.* (2020). DOI: 10.1007/s10479-020-03785-4.

[30] R. Oliveira, A. Chaves and L. Lorena, A comparison of two hybrid methods for constrained clustering problems. *Appl. Soft Comput.* **54** (2017) 256–266.

[31] M.K. Pakhira, S. Bandyopadhyay and U. Maulik, A study of some fuzzy cluster validity indices, genetic clustering and application to pixel classification. *Fuzzy Sets Syst.* **155** (2005) 191–214.

[32] H. Park and C. Jun, A simple and fast algorithm for k-medoids clustering. *Expert Syst. App.* **36** (2009) 3336–3341.

[33] R. Punhani, V.P.S. Arora and A. Sai Sabitha, K-prototype algorithm for clustering large data sets with categorical values to established product segmentation. In: Proceedings of Data Analytics and Management, edited by D. Gupta, Z. Polkowski, A. Khanna, S. Bhattacharyya and O. Castillo. Springer, Singapore (2022) 343–353.

[34] M. Rao, Cluster analysis and mathematical programming. *J. Am. Stat. Assoc.* **66** (1971) 622–626.

[35] E. Schubert and P.J. Rousseeuw, Faster k-medoids clustering: Improving the pam, clara, and clarans algorithms. In: Similarity Search and Applications, edited by G. Amato, C. Gennaro, V. Oria and M. Radovanović. Springer International Publishing, Cham (2019) 171–187.

[36] E. Schubert and P.J. Rousseeuw, Fast and eager k-medoids clustering: $O(k)$ runtime improvement of the pam, clara, and clarans algorithms. *Inf. Syst.* **101** (2021) 101804.

[37] G. Semaan, *Algoritmos para o Problema de Agrupamento Automático*. Ph.D. thesis, Federal Fluminense University (2013).

[38] G.S. Semaan, J.A. de Moura Brito, I. Machado Coelho, E. Franco Silva, A. Cesar Fadel, L. Satoru Ochi and N. Maculan, A brief history of heuristics: from bounded rationality to intractability. *IEEE Latin Am. Trans.* **18** (2021) 1975–1986.

[39] W. Sheng and X. Liu, A hybrid algorithm for k-medoid clustering of large data sets. In: Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753) **1** (2004) 77–82.

[40] W. Sheng and X. Liu, A genetic k-medoids clustering algorithm. *J. Heuristics* **12** (2006) 447–466.

[41] H. Song, J.G. Lee and W.S. Han, PAMAE: parallel k-medoids clustering with high accuracy and efficiency. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Association for Computing Machinery, New York, NY, USA (2017) 1087–1096.

[42] V.M. Spears and K.A.D. Jong, On the virtues of parameterized uniform crossover. In: Proceedings of the Fourth International Conference on Genetic Algorithms. (1991) 230–236.

[43] B. Tavakkol and Y. Son, Fuzzy kernel k-medoids clustering algorithm for uncertain data objects. *Pattern Anal. App.* **24** (2021) 1287–1302.

[44] L. Tseng and S. Yang, A genetic approach to the automatic clustering problem. *Pattern Recogn.* **34** (2001) 415–424.

[45] A. Ushakov and I. Vasilyev, Near-optimal large-scale k-medoids clustering. *Inf. Sci.* **545** (2021) 344–362.

[46] A.V. Ushakov and I. Vasilyev, A parallel heuristic for a k-medoids clustering problem with unfixed number of clusters. In: 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). (2019) 1116–1120.

[47] M.J. van der Laan, K.S. Pollard and J. Bryan, A new partitioning around medoids algorithm. *J. Stat. Comput. Simul.* **78** (2003) 575–584.

[48] H. Vinod, Integer programming and theory of grouping. *J. Am. Stat. Assoc.* **64** (1969) 506–517.

[49] X. Wang and X. Wang, A fast k-medoids clustering algorithm for image segmentation based object recognition. *J. Rob. Autom.* **4** (2021) 202–211.

[50] L. Wolsey, Integer Programming, 2nd edition. Wiley (2020).

[51] D. Xu and Y. Tian, comprehensive survey of clustering algorithms. *Ann. Data Sci.* **2** (2015) 165–193.

[52] R. Xu and D. Wunsch, Survey clustering algorithms. *IEEE Trans. Neural Netw.* **16** (2005) 645–678.

[53] D. Yu, G. Liu, M. Guo and X. Liu, An improved k-medoids algorithm based on step increasing and optimizing. *Expert Syst. App.* **92** (2018) 464–473.

[54] S. Zadegan, M. Mirzaie and F. Sadoughi, Ranked k-medoids: a fast and accurate rank-based partitioning algorithm for clustering large datasets. *Knowl.-Based Syst.* **39** (2013) 133–143.

[55] Q. Zhang and I. Couloigner, A new and efficient k-medoid algorithm for spatial clustering. *Lecture Notes Comput. Sci.* **3482** (2005) 181–189.

[56] S. Zhu, L. Xu and E. Goodman, Evolutionary multi-objective automatic clustering enhanced with quality metrics and ensemble strategy. *Knowl. Based Syst.* **188** (2020) 105018.