São José dos Campos - SP
LV SBPO
LV Simpósio Brasileiro de Pesquisa Operacional
São José dos Campos, SP - 6 a 9 de novembro de 2023

Biased Random-key Genetic Algorithm with Q-Learning for Constrained Clustering

Gislaine Camila de Freitas *
gislaine.freitas@unifesp.br

Luiz Henrique Nogueira Lorena *
luiz.lorena@unifesp.br

Luiz Antonio Nogueira Lorena *
luiz.lorena15@unifesp.br

Antônio Augusto Chaves *
antonio.chaves@unifesp.br

* Fed. Univ. of São Paulo - UNIFESP/ICT
Parque Tecnológico - Avenida Cesare Mansueto Giulio Lattes, n° 1201 - Eugênio de Mello

ABSTRACT

This paper presents a technique to solve the Constrained Clustering Problem, a semi-supervised problem that uses external specialist knowledge to guide clustering. Several algorithms were proposed in the literature to solve this problem, but their results may differ in terms of quality. A recent paper introduced an integer linear programming model based on k-medoids that consider consensus information while clustering. Their results showed how this information could improve clustering quality. This model, however, is time-consuming in some datasets. This paper describes how a Biased Random-key Genetic Algorithm with Q-Learning (BRKGA-QL) metaheuristic was adapted to use such ideas. The experiments evaluated the quality of the solutions and computational time obtained by BRKGA-QL and the integer linear programming model in 25 datasets from the literature. The experimental results show that the BRKGA-QL was able to reduce computational burden while maintaining solution quality.

KEYWORDS. Consensus Clustering. Constrained Clustering. Metaheuristics. Genetic algorithm.

Paper topics (CI – Computational Intelligence, MP - Mathematical Programming)

DOI: 10.59254/sbpo-2023-174850
http://live-proceedings.pantheonsite.io/p/174850

1. Introduction

Clustering is a machine learning task used in different fields of science and industry. It has applications in medical imaging, market segmentation, social network analysis, data compression, and many others [Ezugwu et al., 2022]. The objective is to cluster a set of objects in the dataset according to their similarity. Usually, this task is performed in a non-supervised way, ignoring the domain knowledge from a specialist. However, several works show how such information can improve clustering quality [Wagstaff e Cardie, 2000; Wagstaff et al., 2001; Bilenko et al., 2004; Zhang et al., 2019, 2022].

This paper focuses on a variant of the Clustering task denoted as Constrained Clustering [Basu et al., 2008]. It is a semi-supervised approach that uses the external specialist domain knowledge through constraints that state which pairs of objects must be in the same group or not in the resulting clustering partition. Several Constrained Clustering techniques were proposed in the literature, from integer linear programming (ILP) models [Randel et al., 2018], heuristics [Wagstaff et al., 2001; Khashabi et al., 2015; Antoine et al., 2012; Pelleg e Baras, 2007; Gao et al., 2023], and metaheuristics de Oliveira et al. [2017]; González-Almagro et al. [2020, 2021]; González-Almagro et al. [2022].

One way to balance the outcome of different algorithms is to use a consensus technique. A recent paper [Freitas et al., 2022] proposed a k-medoids ILP model, denoted as KM-α, which uses the information provided by the consensus of different Constrained Clustering algorithms to improve solution quality. It is a bi-objective model that uses the distance between objects and a consensus matrix to guide the clustering process. Users can control the objective function emphasis through a parameter α, giving more or less emphasis on the domain knowledge presented in the consensus matrix. The authors tested this model using five constrained clustering algorithms and 25 datasets proposed in the literature. Their experimental results have shown that considering consensus information improved the quality of the obtained partitions (measured through the Adjusted Rand Index Hubert e Arabie [1985] measure). This technique, however, is time-consuming in some datasets. Since KM-α provided quality results but was time-consuming for larger problems, this paper proposes the adaptation of a metaheuristic to reduce the computational burden. The Biased Random-key Genetic Algorithm with Q-Learning (BRKGA-QL) [Chaves e Lorena, 2021; Schenekemberg et al., 2022] was selected as the base metaheuristic because of its modularity and usability (only one parameter to tune).

This paper uses the following structure. First, section 2 presents the Constrained Clustering Problem definition, applications, notation, and related research. Next, section 3 the KM-α ILP model. Section 4 describes how the BRKGA-QL metaheuristic was adapted. Section 5 presents the experimental results. Finally, Section 6 presents the concluding remarks of this work.

2. Constrained Clustering Problem

In the Constrained Clustering Problem (CCP) [Basu et al., 2008], a clustering algorithm should create a partition Π_X (a set of one or more groups) for a dataset X composed of n objects, given a set of constraints provided by an external specialist. There are two types of constraints: one states that a pair of objects should belong to the same group in the resulting partition (must-link or ML), and another one states the opposite (cannot-link or CL). Figure 1 graphically illustrates the CCP on a small dataset composed of five objects. It can be observed from the example that the resulting partition respects the ML and CL constraints provided by the external specialist.

3. KM-α ILP model

KM-α is a bi-objective ILP model for the CCP proposed in Freitas et al. [2022] as an adaptation to the *k-Medoids Constrained* (KMC) ILP model (Randel et al. [2018]). In KMC, the

DOI: 10.59254/sbpo-2023-174850
http://live-proceedings.pantheonsite.io/p/174850

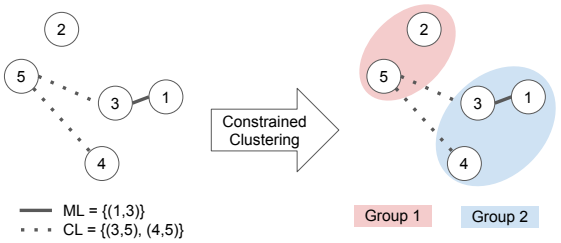


Figure 1: Resulting partition Π_X given a dataset X composed of $n = 5$ objects and a set of ML and CL constraints.

euclidean distance is used to cluster the objects, and the ML and CL constraints are explicitly considered in the mathematical model. In KM-α, however, such constraints are considered indirectly in its objective. The objective considers two criteria: minimizing the distance between objects and medoids and maximizing the pairwise information that considers if an object and medoid should be together. The pairwise information is represented by a consensus matrix created based on the solutions of different constrained clustering algorithms. Finally, a parameter α controls which side of the objective function the final user wants to emphasize more (distance or consensus). By not considering constraints explicitly, KM-α can be used to solve problems CCP with hard, soft, and group constraints Law et al. [2004] or constrained overlapping clustering Scripps e Tan [2010]. The KM-α ILP model is defined as:

$$\text{Minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{n}[(1-\alpha)d_{ij} - \alpha c_{ij}]x_{ij}$$

$$\text{Subject to} \quad \sum_{j=1}^{n} x_{jj} = k \tag{1}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j = 1, ..., n \tag{2}$$

$$x_{jj} \geq x_{ij} \qquad \forall j = 1, ..., n, \forall i = 1, ..., n \tag{3}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i = 1, ..., n, \forall j = 1, ..., n \tag{4}$$

The KM-α ILP model uses as input a distance matrix (D), consensus matrix (C), and the parameters α and k. The parameter $\alpha \in [0, 1]$ controls the balance between distance and consensus objectives, while k indicates the number of groups the user wants to obtain. The value d_{ij} represents the distance between objects i and j in the distance matrix D. In contrast, c_{ij} is provided by the consensus matrix C. This value represents the number of solutions from the constrained clustering algorithms that assign the objects i and j to the same group. The n by n consensus matrix C is created based on the partitions obtained by different constrained clustering heuristics.

4. BRKGA-QL

BRKGA-QL was introduced in Chaves e Lorena [2021] to reduce the burden of parameter tuning of the original BRKGA technique proposed in Gonçalves e Resende [2011]. It uses the Q-Learning Reinforcement Learning algorithm Watkins [1989] to learn the best parameters of the BRKGA during the evolutionary process. Further enhancements were proposed for BRKGA-QL in Schenekemberg et al. [2022] to the diversification and intensification of the Q-Learning technique and the diversity of the elite set.

DOI: 10.59254/sbpo-2023-174850
http://live-proceedings.pantheonsite.io/p/174850

The enhanced version of the BRKGA-QL algorithm Schenekemberg et al. [2022] is used in this paper. Figure 2 presents its main components. It follows a modular structure, and the final user only needs to implement the problem-dependent components: the decoder and the local search. Moreover, it requires only one parameter: the amount of time (seconds) the algorithm should execute. Hence, this reduces the amount of burden of parameter tuning. This parameter is tuned based on the complexity of the proposed decoders and local search.

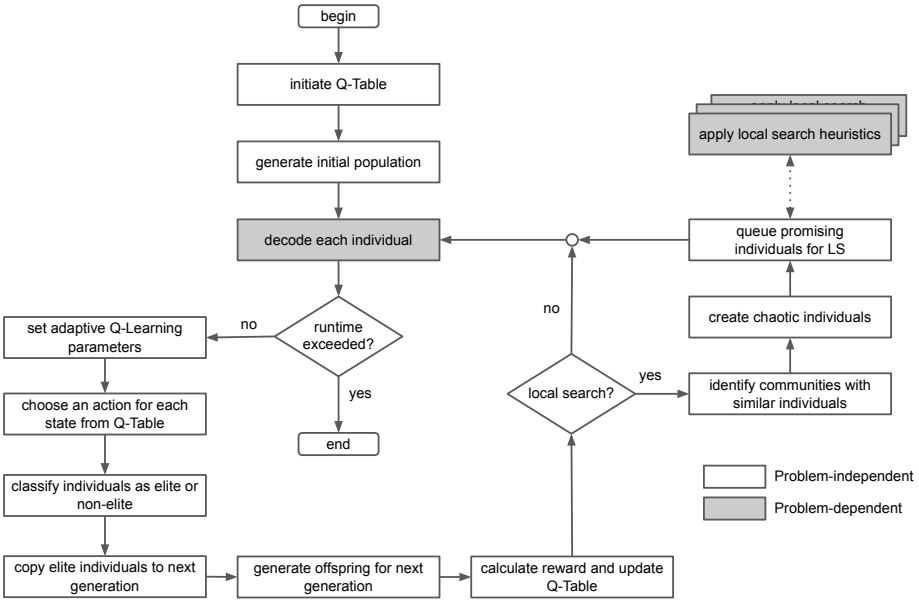


Figure 2: BRKGA-QL main components. Adapted from Schenekemberg et al. [2022].

4.1. Solution Representation

Figure 3 presents the solution representation adopted in the BRKGA-QL. Every position of the random-keys vector (r) represents a medoid. The decoder transforms the vector r to a CCP solution vector (s), and the encoder transforms the solution vector s back to its random-keys form k. The remaining objects are assigned to groups during fitness evaluation (more details in the next subsection).

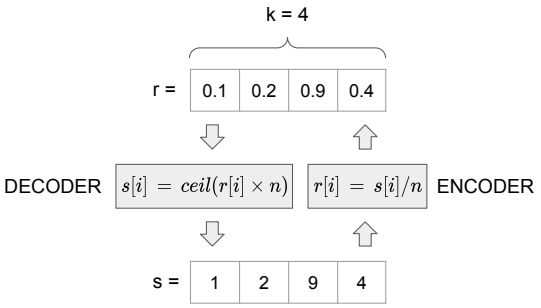


Figure 3: Solution representation for a CCP problem with $k = 4$ medoids and $n = 10$ objects.

Algorithm 1 represents the proposed decoder. Line 4 is reponsible for mapping the

DOI: 10.59254/sbpo-2023-174850
http://live-proceedings.pantheonsite.io/p/174850

random-key value ($r[i]$) from the continuous space $\{r[i] \in \mathbb{R} \mid 0 \leq r[i] \leq 1\}$ to a discrete solution space $\{s[i] \in \mathbb{N} \mid 1 \leq s[i] \leq n\}$. Meanwhile, Algorithm 2 is responsible for the inverse mapping. Algorithm 3 was proposed to create feasible solutions for the CCP since, for instance, a solution with repeated medoids can not exist. The *CREATESOLUTION* function receives as input the number of medoids (k) and the number of objects in the dataset (n) and outputs a random-key vector (r).

Algorithm 1: DECODER

Input: Random-keys vector (r), number of medoids (k) and objects (n)
Output: Solution vector (s)
1 **function** *DECODER (r, k, n)*
2 $s \leftarrow [0, 0, \ldots, 0]$ // vector of size k
3 **for** $i \leftarrow 1$ **to** k **do**
4 $s[i] \leftarrow ceil(r[i] * n])$
5 **end**
6 **return** s
7 **end**

Algorithm 2: ENCODER

Input: Solution vector (s), number of medoids (k) and objects (n)
Output: Random-keys vector (r)
1 **function** *ENCODER (s, k, n)*
2 $r \leftarrow [0.0, 0.0, \ldots, 0.0]$ // vector of size k
3 **for** $i \leftarrow 1$ **to** k **do**
4 $r[i] \leftarrow s[i]/n$
5 **end**
6 **return** r
7 **end**

The auxiliary vector *medoids* in Line 3 controls if a medoid was already selected while constructing a solution. The loop in Lines 4-10 selects the k medoids. The *do-while* (Lines 5-7) prevents the selection of repeated medoids. Finally, in Line 9, the medoid selected for the ith position is encoded to a value in the continuous space $\{r[i] \in \mathbb{R} \mid 0 \leq r[i] \leq 1\}$.

Algorithm 3: CREATESOLUTION

Input: Number of groups (k) and objects (n)
Output: Random-keys vector (r)
1 **function** *CREATESOLUTION (k, n)*
2 $r \leftarrow [0.0, 0.0, \ldots, 0.0]$ // vector of size k
3 $medoids \leftarrow [0, 0, \ldots, 0]$ // vector of size n
4 **for** $i \leftarrow 1$ **to** k **do**
5 **do**
6 $id \leftarrow randomint(1, n)$
7 **while** $medoids[id] = 1$;
8 $medoids[id] \leftarrow 1$
9 $r[i] \leftarrow id/n$ // Encode
10 **end**
11 **return** r
12 **end**

Algorithm 4: FITNESS

Input: Solution vector (s), number of medoids (k) and objects (n), α parameter, the distance (D) and consensus (C) matrices
Output: Fitness value (fit)
1 **function** *FITNESS (s, k, n, α, D, C)*
2 $fit \leftarrow 0.0$
3 **for** $i \leftarrow 1$ **to** n **do**
4 $fit_{best} \leftarrow \infty$
5 **for** $j \leftarrow 1$ **to** k **do**
6 $fit_{i,s[j]} \leftarrow (1-\alpha)D[i, s[j]] - \alpha C[i, s[j]]$
7 **if** $fit_{i,s[j]} < fit_{best}$ **then**
8 $fit_{best} \leftarrow fit_{i,s[j]}$
9 **end**
10 **end**
11 $fit \leftarrow fit + fit_{best}$
12 **end**
13 **return** fit
14 **decoderend**

DOI: 10.59254/sbpo-2023-174850
http://live-proceedings.pantheonsite.io/p/174850

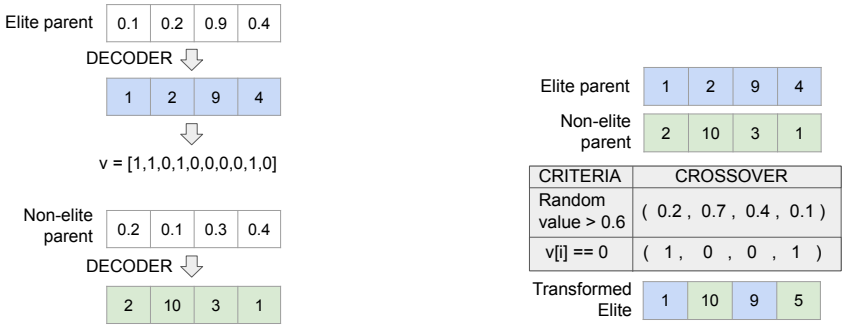
4.2. Fitness evaluation

Algorithm 4 presents the proposed fitness function that resembles the objective function of the KM-α model. It receives as input a decoded solution vector (s), the number of medoids (k), the number of objects (n), the α parameter, the distance (D) and consensus (C) matrices. The algorithm outputs the fitness value (fit). The asymptotic complexity of the algorithm is $O(n * k)$.

Algorithm 4 aims to assign the remaining objects of the dataset to groups defined by the medoids of the vector solution (s). In Line 2, the algorithm assigns zero to the fitness variable (fit), then it starts a loop at Lines 3-12 that evaluates the best group for each object i. Line 4 defines a variable (fit_{best}) to store the fitness that gives the best partial increment to the problem's objective function. The loop in Line 5 checks who is the best medoid for an object. Line 6 evaluates the benefit of assigning an object i to the same group as medoid $s[j]$. Line 7 tests if the current medoid being evaluated is the best; if so, Line 8 stores it in fit_{best}. Finally, at Line 11, the solution's fitness is incremented with the best partial increment (fit_{best}).

4.3. Crossover and Mutation

The original parametric crossover operator from the BRKGA-QL used in this paper [Schenekemberg et al., 2022] was adapted to maintain the feasibility of the created individuals (Figure 4). The crossover function comprises two phases: first, it decodes the elite and non-elite random-keys vectors. An auxiliary vector (v) is also created to control which medoids exist in the elite solution. In the next phase, the crossover operation is executed. It uses the elite parent as its base and only accepts an allele from the non-elite parent if two criteria are satisfied: the random number is greater than the crossover probability and if the solution does not already contain the medoid being exchanged.



(a) Preprocessing. (b) Parametric Crossover.

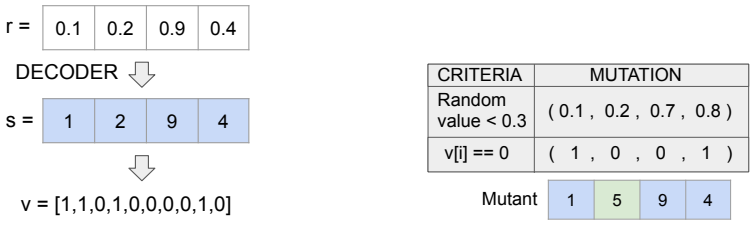
Figure 4: Graphical representation of the modified parametric crossover operator. Figura (a) presents the preprocessing steps, while (b) shows the crossover operation.

The original mutation operator from the BRKGA-QL was also adapted to maintain the feasibility of the created individuals (Figure 5). This function also comprises two phases: first, it decodes the selected random-keys vector and creates an auxiliary vector (v) to be used during mutation. In the next phase, the mutation is performed in an allele only if two criteria are satisfied: the random number is less than the mutation probability and if the solution does not already contain the medoid being exchanged.

5. Experimental Results

All experiments were conducted on an Intel i7-10750H with 32GB RAM using Windows operating system. The BRKGA-QL was implemented in C++, ILP model in Julia v1.8.4 [Bezanson

DOI: 10.59254/sbpo-2023-174850
http://live-proceedings.pantheonsite.io/p/174850



(a) Preprocessing. (b) Mutation Operator.

Figure 5: Graphical representation of the modified mutation operator. Figura (a) presents the preprocessing steps, while (b) shows the mutation operation.

et al., 2017] using the package JuMP v1.6 [Dunning et al., 2017]. Finally, CPLEX v20.1 [IBM, 1987-2020] was used as the optimization solver. Table 1 presents the details about the datasets used in the experiments. It shows their names, the total number of objects, classes, and the source from the literature that provided it. The ML and CL constraints for the datasets were proposed in Baumann [2020] [1]. Three types of constraints were created based on the original labels of the datasets and the number of objects in the dataset (10%, 15%, and 20%).

Table 1: Datasets.

Name	Objects	Classes	Source
Soybean	47	4	*UCI*
Zoo	101	7	*Keel*
Appendicitis	106	2	*Keel*
Iris	150	3	*Keel*
Tae	151	3	*Keel*
Hayesroth	160	3	*Keel*
Wine	178	3	*Keel*
Sonar	208	2	*Keel*
Glass	214	6	*Keel*
Newthyroid	215	3	*Keel*
Spectfheart	267	2	*Keel*
Heart	270	2	*Keel*
Circles	300	2	*Github*
Spiral	300	2	*Github*
Moons	300	2	*Github*
Haberman	306	2	*Keel*
Ecoli	336	8	*Keel*
Bupa	345	2	*Keel*
Ionosphere	351	2	*Keel*
Movement Libras	360	15	*Keel*
Monk2	432	2	*Keel*
Saheart	462	2	*Keel*
Led7Digit	500	10	*Keel*
Breast Cancer	569	2	*Sklearn*
Vehicle	846	4	*Keel*

The following constrained clustering heuristics were selected to evaluate the proposed BRKGA-QL: BLPKM Baumann [2020], TVClust Khashabi et al. [2015], and LCVQE Pelleg e Baras [2007]. They were selected because of their efficiency (computational time) and relatively

[1] https://drive.google.com/drive/u/1/folders/1sjnPYitey8q9zrPKa_YpFS7iNKTT15QH

DOI: 10.59254/sbpo-2023-174850
http://live-proceedings.pantheonsite.io/p/174850

different views on how to solve the constrained clustering problem. The BLPKM was implemented in Julia and we used the TVClust and LCVQE implementation avaiable in Github[2].

Figure 6 graphically illustrates the sequence (from top to bottom) adopted in the experiments. Initially, it presents the type of data the problem uses: a dataset in tabular form and a file that stores the pairs of constraints. Next, it presents the input format the Constrained Clustering Algorithms use: a distance matrix D, a set of ML and CL constraints, and the number of groups (k). The next panel (Constrained Clustering output) presents an example of the execution of the CCP algorithms and the resulting partitions and consensus matrix C. Panel "Consensus Clustering input" shows the input format for the consensus techniques: the distance matrix D, the consensus matrix C, and the number of groups (k). Finally, the last panel presents the partitions obtained by the consensus algorithms for different values of the α parameter.

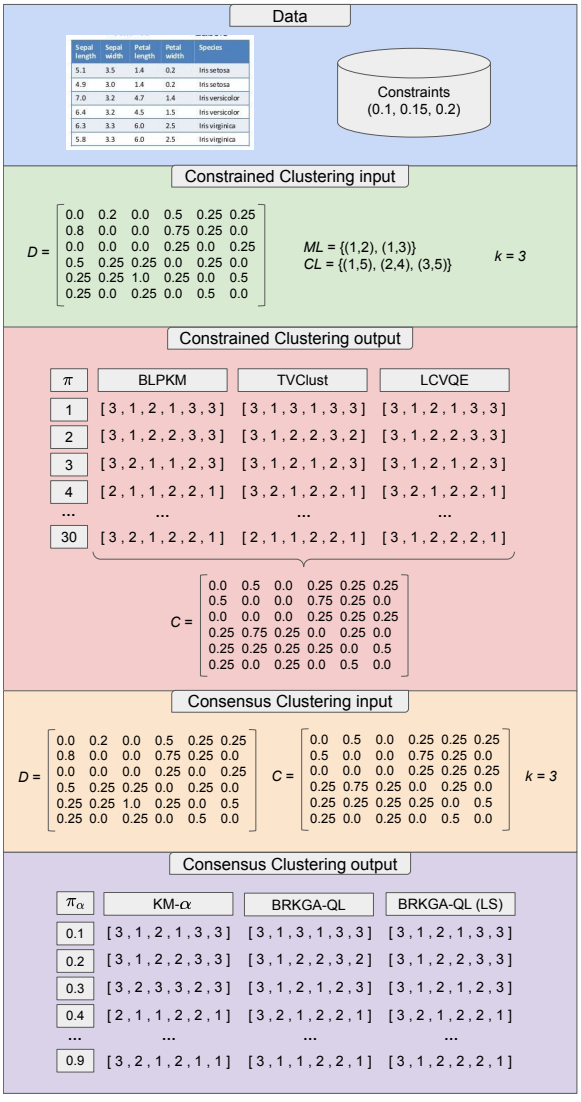


Figure 6: Schematic illustration of the experiments.

[2] https://github.com/GermangUgr/TFG

DOI: 10.59254/sbpo-2023-174850
http://live-proceedings.pantheonsite.io/p/174850

The study conducted 30 executions of the constrained clustering heuristics without any time limitations and a single execution of KM-α for each α value without any time restrictions. The algorithm performs at least one complete iteration. Additionally, BRKGA-QL was executed ten times using α values ranging from 0 to 1, in increments of 0.1, with a time limit of 30 seconds per execution. This time limit was stipulated empirically. One of the factors considered was the number of carried out to be experiments. An experiment with a sample of each dataset showed that 30 seconds was enough for the method to converge to a local optimum.

The quality of the partitions obtained by the heuristics and the consensus techniques was evaluated using the Adjusted Rand Index (ARI) metric Hubert e Arabie [1985] against the original dataset classes. The ARI value lies between 0 and 1, where the value 1 represents that the partitions are identical. Table 2 presents the experimental results for the heuristics and the consensus techniques. The time measurement unit is seconds. The column denoted as α in KM-α and BRKGA-QL represents the α that provided the solution with the highest ARI score. The "time" columns represent the time (in seconds) to achieve the best solution.

The results presented in Table 2 show that the best algorithm in terms of ARI was BLPKM, which obtained the highest ARI values for almost all instances. However, this algorithm explicitly considers the ML and CL constraints in the mathematical model, which may result in infeasible solutions for instances that contain bridge objects, such as those considered in the article Scripps e Tan [2010]. It is also noted that the results of the KM-α and BRKGA-QL algorithms were improved by the consensus information, as the best value of α for both algorithms was greater than 0 for all instances. BRKGA-QL was computationally faster than KM-α, especially for instances with a larger number of objects.

The consensus results were affected by the algorithms considered, as two of them (TVClus and LCVQE) did not perform well in isolation. This directly affected the consensus information of the KM-α and BRKGA-QL algorithms. Furthermore, within this approach, the search duration was restricted, which might have resulted in the method not discovering the optimal scenario due to the brief timeframe.

6. Conclusions

This paper described how the Biased Random-key Genetic Algorithm with Q-Learning (BRKGA-QL) metaheuristic was adapted to work as a consensus technique for the Constrained Clustering Problem. The BRKGA-QL used a recent ILP-based consensus model, denoted as KM-α, as its basis. Its fitness function was adapted to work as the objective function of KM-α. In addition, a custom decoder and local search were proposed for the consensus.

The computational results showed that the KM-α and BRKGA-QL algorithms benefited from the consensus matrix information, improving their results for values greater than zero of the α parameter. The BRKGA-QL algorithm was better in terms of computational time. In future work, the performance of the BRKGA-QL algorithm could be evaluated on instances with bridge objects where some constraints are not necessarily satisfied. In this case, the goal would be to minimize the number of these constraints.

Acknowledgment

This work was supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001; FAPESP under Grant No. 2018/15417-8 and Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) under Grants 301752/2019-2, 312747/2021-7, and 302261/2019-2.

DOI: 10.59254/sbpo-2023-174850
http://live-proceedings.pantheonsite.io/p/174850

Table 2: Experimental results.

		KM-α			BRKGA-QL			BLPKM		TVClust		LCVQE	
Dataset	Constraint	α	ARI	Time	α	ARI	Time	ARI	Time	ARI	Time	ARI	Time
Soybean	0.1	0.1	**1.0**	0.05	0.1	**1.0**	0.08	**1.0**	0.11	0	0.06	**1.0**	0.13
	0.15	0.1	**1.0**	0.05	0.1	**1.0**	0.05	**1.0**	0.08	0	0.09	**1.0**	0.13
	0.2	0.1	**1.0**	0.05	0.1	**1.0**	0.82	**1.0**	0.09	0	0.06	**1.0**	0.13
Zoo	0.1	0.6	0.7	0.24	0.6	0.7	0.22	**0.97**	0.08	0.72	0.78	0.94	0.02
	0.15	1.0	0.71	0.24	1.0	0.71	9.17	**0.95**	0.15	0.84	0.85	0.94	0.01
	0.2	0.3	0.7	0.22	0.3	0.7	4.04	**0.96**	0.19	0.79	0.69	0.95	0.02
Appendicitis	0.1	0.4	0.53	0.34	0.4	0.53	0.09	**0.62**	0.08	0.56	0.13	0.45	0.14
	0.15	0.5	0.8	0.3	0.5	0.8	0.11	**1.0**	0.06	0.69	0.47	0.39	0.01
	0.2	0.5	0.76	0.48	0.5	0.76	0.09	**1.0**	0.05	0.91	0.59	0.05	0.36
Iris	0.1	0.4	0.83	0.77	0.4	0.83	24.44	**0.89**	0.19	0.94	0.65	0.69	0.13
	0.15	0.6	**0.96**	0.71	0.6	**0.96**	0.42	**0.96**	0.19	**1.0**	0.42	0.94	0.14
	0.2	0.2	**1.0**	0.77	0.2	**1.0**	0.29	**1.0**	0.12	**1.0**	2.96	0.96	0.13
Tae	0.1	0.9	0.11	1.55	0.9	0.11	0.43	**0.19**	0.17	0.07	0.32	0.09	0.01
	0.15	0.9	0.35	1.2	0.9	0.35	0.26	**0.74**	0.2	0.07	0.77	0.04	0.4
	0.2	0.7	0.31	1.45	0.7	0.31	0.26	**0.98**	0.1	0.07	1.28	0.05	0.49
Hayesroth	0.1	1.0	0.05	1.17	1.0	0.05	2.1	**0.25**	0.2	0.17	0.86	0.12	0.01
	0.15	1.0	0.82	1.57	1.0	0.82	0.42	**0.98**	0.2	0.4	1.49	0.13	0.03
	0.2	1.0	0.86	1.48	1.0	0.86	0.38	0.94	0.19	**0.96**	1.24	0.1	0.41
Wine	0.1	0.1	**0.91**	1.88	0.1	**0.91**	29.28	0.66	0.21	0.68	0.9	0.69	0.34
	0.15	0.1	0.85	1.68	0.1	0.85	0.34	0.57	0.09	**0.92**	1.67	0.46	0.01
	0.2	0.1	**0.92**	2.02	0.1	**0.92**	10.17	0.58	0.16	0.9	0.82	0.45	0.42
Sonar	0.1	1.0	0.78	2.99	1.0	0.78	0.26	**0.8**	0.11	0	0.17	0.05	0.14
	0.15	1.0	0.82	3.55	1.0	0.82	0.29	**0.98**	0.11	0	0.17	0.57	0.44
	0.2	0.5	**1.0**	3.84	0.5	**1.0**	0.32	**1.0**	0.09	0	0.19	0.04	0.83
Glass	0.1	0.8	0.26	3.08	0.8	0.26	11.82	**0.43**	0.37	0.29	1.84	0.22	0.48
	0.15	0.8	0.59	2.93	0.8	0.59	22.15	**0.8**	1.13	0.29	2.09	0.32	0.49
	0.2	0.3	0.63	3.48	0.3	0.63	13.47	**0.96**	0.91	0.29	1.64	0.3	0.75
Newthyroid	0.1	0.2	0.95	3.05	0.2	0.95	0.33	**0.97**	0.17	0.95	1.6	0.95	0.14
	0.15	0.2	**1.0**	3.1	0.2	**1.0**	4.05	**1.0**	0.1	0.98	0.51	0.98	0.16
	0.2	0.2	**0.98**	2.85	0.2	**0.98**	15.91	**0.98**	0.28	**1.0**	0.91	0.95	0.14
Spectfheart	0.1	0.3	0.01	6.45	0.3	0.01	0.79	**0.87**	0.14	0	0.42	0.01	0.72
	0.15	1.0	**1.0**	18.13	1.0	**1.0**	0.47	**1.0**	0.09	0	0.22	0.13	1.14
	0.2	0.2	-0.01	6.89	0.2	-0.01	0.39	**1.0**	0.14	0	0.26	-0.01	1.48
Heart	0.1	0.8	0.82	5.52	0.8	0.82	1.16	**0.9**	0.11	0.76	2.39	0.72	0.16
	0.15	1.0	0.93	5.47	1.0	0.93	0.39	**1.0**	0.12	**1.0**	1.41	0.64	0.12
	0.2	0.9	**1.0**	5.29	0.9	**1.0**	0.77	**1.0**	0.16	**1.0**	1.72	0.71	0.15
Circles	0.1	1.0	0.43	8.65	1.0	0.43	1.12	**0.87**	0.11	0.15	1.22	0.01	0.4
	0.15	1.0	0.38	8.2	1.0	0.38	1.42	**1.0**	0.1	**1.0**	2.03	0.01	0.77
	0.2	1.0	0.45	7.92	1.0	0.45	1.74	**1.0**	0.16	**1.0**	2.07	0.01	1.34
Spiral	0.1	1.0	0.39	7.8	1.0	0.39	1.24	**0.88**	0.08	0.12	0.87	0.02	0.13
	0.15	0.8	**1.0**	8.25	0.8	**1.0**	1.08	**1.0**	0.11	**1.0**	2.37	0.02	0.81
	0.2	0.8	**1.0**	7.74	0.8	**1.0**	2.27	**1.0**	0.17	**1.0**	1.13	0.15	1.21
Moons	0.1	0.8	**0.99**	7.4	0.8	**0.99**	1.9	**0.99**	0.07	**0.99**	2.4	0.63	0.15
	0.15	0.5	**1.0**	9.39	0.5	**1.0**	1.14	**1.0**	0.17	**1.0**	1.32	0.82	0.44
	0.2	0.3	**1.0**	6.85	0.3	**1.0**	1.62	**1.0**	0.18	**1.0**	0.71	**1.0**	0.12
Haberman	0.1	0.3	0.01	9.17	0.3	0.01	1.92	**0.93**	0.11	0.34	1.29	0	0.56
	0.15	0.3	**1.0**	16.26	0.3	**1.0**	1.21	**1.0**	0.12	**1.0**	2.95	0.01	0.92
	0.2	0.4	**1.0**	14.24	0.4	**1.0**	1.01	**1.0**	0.17	**1.0**	1.01	0	1.57
Ecoli	0.1	0.7	0.45	11.39	0.7	0.45	3.44	**0.77**	1.4	0.73	6.69	0.62	0.03
	0.15	0.6	0.66	11.6	0.6	0.66	14.82	**0.95**	0.99	0.81	7.92	0.75	0.64
	0.2	0.3	0.65	12.41	0.3	0.65	8.45	**0.99**	1.15	0.87	7.35	0.77	0.82
Bupa	0.1	0.6	0.32	13.09	0.6	0.32	1.28	**0.94**	0.11	0	0.89	0.01	0.52
	0.15	0.4	0.36	17.48	0.4	0.36	1.67	**1.0**	0.16	0	0.78	0.01	1.15
	0.2	0.3	0.01	16.67	0.3	0.01	1.47	**1.0**	0.2	0.01	1.08	0.01	1.87
Ionosphere	0.1	0.3	0.53	14.18	0.3	0.53	1.76	**0.94**	0.14	0	0.24	0.05	0.76
	0.15	0.3	0.32	14.32	0.3	0.32	1.3	**1.0**	0.15	0	0.28	0	1.28
	0.2	0.4	**1.0**	13.17	0.4	**1.0**	1.64	**1.0**	0.21	0	0.34	0	2.03
Movement Libras	0.1	0.2	0.34	13.05	0.2	0.34	14.77	**0.37**	6.28	0	2.31	0.35	0.08
	0.15	1.0	0.4	14.09	1.0	0.4	19.1	**0.41**	6.61	0	3.06	0.36	3.52
	0.2	0.9	0.55	14.8	0.9	0.55	13.87	**0.63**	19.99	0	3.13	0.38	3.6
Monk2	0.1	1.0	0.87	32.26	1.0	0.87	1.09	0.97	0.17	**0.99**	1.17	0.65	0.5
	0.15	0.6	0.86	24.82	0.6	0.86	2.06	**1.0**	0.2	**1.0**	1.31	0.63	1.05
	0.2	0.6	**1.0**	26.4	0.6	**1.0**	1.55	**1.0**	0.3	**1.0**	1.84	0.87	1.08
Saheart	0.1	0.8	0.68	34.59	0.8	0.68	1.57	**0.99**	0.3	**0.99**	4.94	0.14	1.1
	0.15	0.8	**1.0**	39.92	0.8	**1.0**	1.05	**1.0**	0.22	**1.0**	8.97	0.01	2.2
	0.2	0.6	**1.0**	32.97	0.6	**1.0**	4.76	**1.0**	0.37	**1.0**	1.81	0	4.05
Led7Digit	0.1	0.9	0.64	46.83	1.0	0.64	29.44	**0.69**	4.13	0.3	15.45	0.58	0.03
	0.15	1.0	0.86	47.15	1.0	0.86	5.44	**0.89**	4.23	0.33	10.06	0.68	1.0
	0.2	1.0	0.96	47.42	1.0	0.96	12.27	**1.0**	4.03	0.35	14.8	0.65	1.57
Breast Cancer	0.1	0.3	**0.99**	60.52	0.3	**0.99**	1.8	**0.99**	0.2	0.07	3.78	0.96	0.15
	0.15	0.3	**1.0**	60.22	0.3	**1.0**	4.98	**1.0**	0.35	0.11	4.6	**1.0**	0.17
	0.2	0.2	**1.0**	56.82	0.2	**1.0**	7.05	**1.0**	0.54	0.97	8.67	**1.0**	0.18
Vehicle	0.1	1.0	0.38	526.14	1.0	0.38	28.48	**0.96**	1.11	0.37	23.64	0.08	2.52
	0.15	1.0	0.43	249.49	1.0	0.43	32.35	**1.0**	1.4	0.69	55.65	0.08	4.86
	0.2	0.7	**1.0**	220.07	0.7	**1.0**	30.03	**1.0**	2.41	0.69	24.23	0.07	8.56

DOI: 10.59254/sbpo-2023-174850
http://live-proceedings.pantheonsite.io/p/174850

References

Antoine, V., Quost, B., Masson, M.-H., e Denœux, T. (2012). Cecm: Constrained evidential c-means algorithm. *Computational Statistics & Data Analysis*, 56(4):894–914. ISSN 0167-9473. URL https://www.sciencedirect.com/science/article/pii/S0167947310003646.

Basu, S., Davidson, I., e Wagstaff, K. (2008). Constrained clustering. advances in algorithms, theory, and applications. *CRC Press*, p. 1–4.

Baumann, P. (2020). A binary linear programming-based k-means algorithm for clustering with must-link and cannot-link constraints. In *2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, p. 324–328.

Bezanson, J., Edelman, A., Karpinski, S., e Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98.

Bilenko, M., Basu, S., e Mooney, R. J. (2004). Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning*, p. 11.

Chaves, A. A. e Lorena, L. H. N. (2021). An adaptive and near parameter-free brkga using q-learning method. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, p. 2331–2338. IEEE.

de Oliveira, R. M., Chaves, A. A., e Lorena, L. A. N. (2017). A comparison of two hybrid methods for constrained clustering problems. *Applied Soft Computing*, 54:256–266. ISSN 1568-4946. URL https://www.sciencedirect.com/science/article/pii/S1568494617300388.

Dunning, I., Huchette, J., e Lubin, M. (2017). Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320.

Ezugwu, A. E., Ikotun, A. M., Oyelade, O. O., Abualigah, L., Agushaka, J. O., Eke, C. I., e Akinyelu, A. A. (2022). A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence*, 110:104743.

Freitas, G. C., Lorena, L. H. N., e Lorena, L. A. N. (2022). Algoritmos de consenso para o problema de agrupamento de dados com restrições. In *Anais do LIV Simpósio Brasileiro de Pesquisa Operacional*. URL https://proceedings.science/sbpo/sbpo-2022/trabalhos/algoritmos-de-consenso-para-o-problema-de-agrupamento-de-dados-com-restricoes.

Gao, J., Tao, X., e Cai, S. (2023). Towards more efficient local search algorithms for constrained clustering. *Information Sciences*, 621:287–307.

Gonçalves, J. F. e Resende, M. G. (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525.

González-Almagro, G., Suárez, J. L., Luengo, J., Cano, J.-R., e García, S. (2022). 3shacc: Three stages hybrid agglomerative constrained clustering. *Neurocomputing*, 490:441–461.

DOI: 10.59254/sbpo-2023-174850
http://live-proceedings.pantheonsite.io/p/174850

González-Almagro, G., Luengo, J., Cano, J.-R., e García, S. (2020). Dils: Constrained clustering through dual iterative local search. *Computers & Operations Research*, 121:1–5.

González-Almagro, G., Luengo, J., Cano, J.-R., e García, S. (2021). Enhancing instance-level constrained clustering through differential evolution. *Applied Soft Computing*, 108:107435. ISSN 1568-4946. URL https://www.sciencedirect.com/science/article/pii/S1568494621003586.

Hubert, L. e Arabie, P. (1985). Comparing partitions. *Journal of classification*, 2(1):193–218.

IBM (1987-2020). Ibm ilog cplex 20.1.

Khashabi, D., Wieting, J., Liu, J. Y., e Liang, F. (2015). Clustering with side information: From a probabilistic model to a deterministic algorithm. *arXiv preprint arXiv:1508.06235*.

Law, M. H., Topchy, A., e Jain, A. K. (2004). Clustering with soft and group constraints. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, p. 662–670. Springer.

Pelleg, D. e Baras, D. (2007). K-means with large and noisy constraint sets. In Kok, J. N., Koronacki, J., Mantaras, R. L. d., Matwin, S., Mladenič, D., e Skowron, A., editors, *Machine Learning: ECML 2007*, p. 674–682, Berlin, Heidelberg. Springer Berlin Heidelberg. ISBN 978-3-540-74958-5.

Randel, R., Aloise, D., Mladenović, N., e Hansen, P. (2018). On the k-medoids model for semi-supervised clustering. In *International Conference on Variable Neighborhood Search*, p. 13–27. Springer.

Schenekemberg, C. M., Chaves, A. A., Coelho, L. C., Guimarães, T. A., e Avelino, G. G. (2022). The dial-a-ride problem with private fleet and common carrier. *Computers & Operations Research*, 147:105933.

Scripps, J. e Tan, P.-N. (2010). Constrained overlapping clusters: minimizing the negative effects of bridge-nodes. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 3(1):20–37.

Wagstaff, K. e Cardie, C. (2000). Clustering with instance-level constraints. *AAAI/IAAI*, 1097: 577–584.

Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S., et al. (2001). Constrained k-means clustering with background knowledge. In *Icml*, volume 1, p. 577–584.

Watkins, C. J. C. H. (1989). Learning from delayed rewards.

Zhang, H., Basu, S., e Davidson, I. (2019). A framework for deep constrained clustering–algorithms and advances. *arXiv preprint arXiv:1901.10061*.

Zhang, X., Zheng, Y., Ye, X., Peng, Q., Wang, W., e Li, S. (2022). Clustering with implicit constraints: A novel approach to housing market segmentation. *Transactions in GIS*, 26(2):585–608.

DOI: 10.59254/sbpo-2023-174850
http://live-proceedings.pantheonsite.io/p/174850