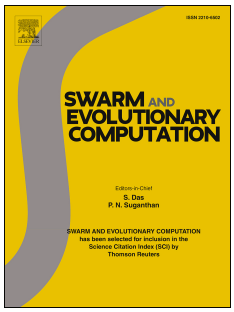
Journal Pre-proof



Unsupervised feature selection based on bio-inspired approaches

Nádia Junqueira Martarelli, Marcelo Seido Nagano

PII: S2210-6502(18)30347-X

DOI: https://doi.org/10.1016/j.swevo.2019.100618

Reference: SWEVO 100618

To appear in: *Swarm and Evolutionary Computation BASE DATA*

Received Date: 27 April 2018

Revised Date: 4 November 2019

Accepted Date: 6 November 2019

Please cite this article as: Ná.Junqueira. Martarelli, M.S. Nagano, Unsupervised feature selection based on bio-inspired approaches, *Swarm and Evolutionary Computation BASE DATA* (2019), doi: https://doi.org/10.1016/j.swevo.2019.100618.

Unsupervised feature selection based on bio-inspired approaches

Nádia Junqueira Martarelli[a,*], Marcelo Seido Nagano[a]

[a] *São Carlos School of Engineering, University of São Paulo, 400 Trabalhador São Carlense Avenue, São Carlos, São Paulo, Brazil.*

Abstract

In recent years, the scientific community has witnessed an explosion in the use of pattern recognition algorithms. However, little attention has been paid to the tasks preceding the execution of these algorithms, the preprocessing activities. One of these tasks is dimensionality reduction, in which a subset of features that improves the performance of the mining algorithm is located and algorithm's runtime is reduced. Although there are many methods that address the problems in pattern recognition algorithms, effective solutions still need to be researched and explored. Hence, this paper aims to address three of the issues surrounding these algorithms. First, we propose adapting a promising meta-heuristic called biased random-key genetic algorithm, which considers a random initial population construction. We call this algorithm as unsupervised feature selection by biased random-key genetic algorithm I. Next, we propose an approach for building the initial population partly in a deterministic way. Thus, we applied this idea in two algorithms, named unsupervised feature selection by particle swarm optimization and unsupervised feature selection by biased random-key genetic algorithm II. Finally, we simulated different datasets to study the effects of relevant and irrelevant attributes, and of noisy and missing data on the performance of the algorithms. After the Wilcoxon rank-sum test, we can state that the proposed algorithms outperform all other methods in different datasets. It was also observed that the construction of the initial population in a partially deterministic way contributed to the better performance. It should be noted that some methods are more sensitive to noisy and missing data than others, as well as to relevant and irrelevant attributes.

Keywords: Unsupervised Feature Selection, Biased Random-Key Genetic Algorithm, Particle Swarm Optimization, Clustering, Data Mining, Simulated Datasets.

Declarations of interest: none.

1. Introduction

Data clustering algorithms search for natural groups in unlabeled datasets, aiming to maximize similarities in each cluster [1, 2]. Because the data is label free, this unsupervised task is considered one of the

*Corresponding author
Email addresses: nadia.martarelli@usp.br (Nádia Junqueira Martarelli), drnagano@usp.br (Marcelo Seido Nagano)

most challenging mining activities [3].

Many fields of study have been requesting data clustering algorithms for a variety of purposes, such as customer segmentation, taxonomy definition, gene and protein function identification, disease diagnosis, biometrics, and speech and image recognition [4].

In many real-world datasets, the performance of data clustering algorithms is hampered by the presence of irrelevant attributes. This type of influence can be minimized by dimensionality reduction methods, which transform the original attributes into a set of new attributes (feature extraction methods) or select a subset of relevant attributes from the original dataset (feature selection methods). Regardless of the approach, this preprocessing task aims to help the clustering algorithm achieve more efficiency and effectiveness [5, 6].

Although dimensionality reduction is necessary to improve the clustering results, this activity is not a trivial task. Unsupervised feature selection methods, for instance, are characterized as a combinatorial problem, with an NP-hard complexity [7], which means that optimal solutions cannot be found in a viable computational time [8].

Such difficulty in finding optimal solutions extends to the three classes of existing feature selection methods, which are those that consider the intrinsic information in data (filter methods), those that are guided by the learning algorithm results (wrapper methods) [9, 10, 11], and those that are guided by the construction of a classifier (embedded methods) [12, 13, 6, 14, 15]. Filter methods are computationally lighter than the wrapper and embedded ones; however, they are not as accurate as the other two [16, 17].

To obtain a good feature subset in a feasible runtime, scientific works have proposed heuristic methods to solve the unsupervised feature selection problem. Many meta-heuristics have already been used for this purpose in both single- and multi-objective optimization scenarios.

In the single-objective optimization context, the genetic algorithm (GA) has been widely applied. In the works of Shamsinejadbabki and Saraee [18], and Abualigahm, Khader and Al-Betar [19], the authors used the GA to select a subset of features in text clustering. A modified GA has been used by Hong, Chen, and Lin [20] to group the attributes in a new feature selection method. Kouser and Priyam [21] implemented the GA to reduce the dimensionality on high-dimensional datasets. A constructive GA-version modeled by Mararelli and Nagano [22], and Anusha and Sathiaseelan [23] also used the GA, but for a multi-objective optimization problem.

Particle Swarm Optimization (PSO) has also been used to solve the unsupervised feature selection problem. This meta-heuristic has already been applied in relative reduction technique [24], filter unsupervised feature selection for consensus clustering [25], gene selection context [26], big data text clustering [27], document clustering [28], and static and dynamic clustering-based methods [29].

In addition to the aforementioned works, there are others that used different heuristic methods for the unsupervised feature selection problem, such as binary bat algorithm (BBA) [30], gravitational search algorithm (GSA) [31], simulated annealing (SA) as a strategy for multi-objective optimization [32], ant colony

optimization (ACO) in graph clustering, modified BBA [33], modified ACO in wrapper-based method [34], and GSA in a multi-objective approach [35].

Although there are already good heuristic approaches to the unsupervised feature selection problem, there is still room for improvement and open issues to be addressed, such as exploring the behavior of other meta-heuristics for this problem, improving the steps of traditional meta-heuristics, and analyzing the performance of the algorithms in specific datasets, such as those with different levels of noisy and missing data, and with a different number of relevant and irrelevant attributes.

An example of a meta-heuristic that has not yet been modeled to the unsupervised feature selection problem is the biased random-key genetic algorithm (BRKGA) [36]. This algorithm obtained excellent results in other optimization problems, many of which were combinatorial problems, as is the case with the feature selection problem.

With respect to unsupervised feature selection, a little-explored improvement point in population-based methods is to change the way the initial population is generated. Usually, the generation of individuals is random to achieve a greater comprehensiveness of the search space. However, if the initial population is also composed of deterministically generated individuals (in the case of this problem, generated by filter methods), the optimization algorithm will gain a starting advantage, because it will have individuals located in promising regions of the search space, in addition to individuals who will be at randomly determined locations, thereby providing a greater balance, between promising regions and random regions, from the beginning of the search.

In the work of Martarelli and Nagano [22], the authors used intrinsic information from data, calculating the variance of the attributes, to build the initial population. However, for this purpose, they did not use the filter feature selection methods, which can be applied for building the initial populations in a better way.

With this in mind, this paper seeks to solve some of the problems surrounding unsupervised feature selection, especially with regard to (i) modeling BRKGA for this process; (ii) partially building the initial population in a deterministic way using filter feature selection methods, and (iii) studying the performance of the algorithms in datasets with different numbers of relevant and irrelevant attributes and different levels of noisy and missing data.

Hence, we propose three unsupervised feature selection algorithms using k-means as the clustering algorithm, wherein the first two are called unsupervised feature selection by biased random-key genetic algorithm I and II (UFSBRKGA-I and UFSBRKGA-II, respectively), and the third one is called unsupervised feature selection by particle swarm optimization (UFSPSO). The only difference between UFSBRKGA-I and UFSBRKGA-II is the fact that in the former, the initial population is randomly built, and in the latter the initial population is partially created in a deterministic way. The UFSPSO follows the same population building idea as UFSBRKGA-II, using the following filter methods: Laplacian score (LS), unsupervised discriminative feature selection (UDFS) and variance thresholding for feature selection (VTFS).

This paper is organized as follows. Section 2 outlines a brief background about dimensionality reduction. Section 3 shows related works in this field. Sections 4, 5, and 6 describe the proposed algorithms, UFSPSO, UFSBRKGA-I, and UFSBRKGA-II, respectively. In section 7, we explain the design of our experiments, the competitor methods, the parameters of the algorithms, and the simulated datasets. In section 8, we show the results and the discussion. Finally, section 9 presents final remarks.

2. Literature review

Dimensionality reduction is one of the preprocessing tasks belonging to the process of knowledge extraction of datasets, called knowledge discovery database. This activity aims to find a reduced number of attributes, which improves the performance of learning algorithms [12, 13, 37]. There are two main methods for finding these relevant attributes subsets, the feature extraction and feature selection (FS) methods.

Feature extraction approaches project the original feature space into a low-dimensional space, resulting in a transformed attributes subset [38, 39]. Although this kind of method is computationally efficient, the original attributes space is not preserved, which might render the features less intelligible. Traditional feature extraction approaches are principal component analysis [40], factorial analysis [41], projection pursuit [42, 43], singular value decomposition [44], and linear discriminant analysis [45].

Feature selection methods select a relevant features subset from the original set. Although they maintain the original data space, this kind of approach is computationally more expensive. There are three main methods that carrying out the feature selection, that are those we described in introduction section (filter, wrapper and embedded methods). It is also possible to describe feature selection methods regarding the label presence in datasets.

Supervised FS methods aim to select attributes that improve the accuracy of supervised learning algorithms, excluding redundant features (those that have an intensive relation to other features), and irrelevant ones (those that can misguide the algorithm, or do not contribute to pattern recognition).

Some of the traditional filter-supervised FS methods are Fisher score, mutual information [46, 47] and correlation coefficient [48, 14]. Some popular wrapper-supervised FS methods are sequential approaches (e.g., forward, backward, and floating) [49, 50] and heuristics methods, which are also popular in embedded methods. Recent work on supervised FS can be found in [51, 52, 53].

Unsupervised FS methods select features that contribute to finding hidden structures in unlabeled data, improving the performance of the data clustering algorithm [54]. Some popular filter-unsupervised FS methods are LS [55] and those that consider feature similarity [56].

Some well-known wrapper-unsupervised FS methods are unsupervised feature selection for multi-cluster data [57] and UDFS [58]. Examples of unsupervised embedded FS methods are the unsupervised feature selection method based on ant colony optimization [59], and embedded unsupervised feature selection (EUFS)

[60]. Recent works on unsupervised FS can be found in [60, 11, 61, 62, 63, 64, 65, 66, 67, 15].

In addition to supervised and unsupervised learning, there is also the semi-supervised approach, which uses the few labels in the dataset to carry out feature selection and improve classifier efficiency. Recent work can be found at [64].

3. Related works

In the unsupervised FS problem, the GA has been extensively used. The seminal work is by Siedlecki and Sklansky [68], in which the GA achieved a better efficiency and efficacy than previous algorithms for the unsupervised FS problem. Lleti, Orti, Sarabia, and Sanchez [69] used the GA for FS in the clustering problem (k-means), concluding that GA was very useful for large datasets. In Liu and Ong's [70] work, the GA was applied as an optimization algorithm to carry out FS in the market segmentation problem. The authors highlighted the fact that classification model accuracy increased after the FS. Recent works on this are [71, 72, 73].

Over the years, new GA-inspired algorithms have been created. These new GA-versions introduce one or more changes in the seven basic steps of the traditional algorithm, aiming to improve the optimization algorithm's effectiveness and efficiency. Two of these new approaches are the constructive genetic algorithm (CGA) [74] and the BRKGA [36].

To address the fragility of GA in directly evaluating schemata, Lorena and Furtado [74] proposed using the CGA for the clustering problem. In addition to directly evaluation, this algorithm calculates the fitness function through a bi-objective optimization function, allowing for a dynamic population size through the generations. In addition, a new individual receives genetic information in one of the best individuals from that generation and from another individual randomly selected. These changes provide more mechanisms for the algorithm to perform exploration and exploitation in the search space, making it more powerful.

The CGA's performance has been proven by many scientific works, which have addressed problems such as (i) minimization of the maximum number of open piles during a cutting process [75], (ii) gate matrix layout [76], (iii) proportionate flexible flow shop scheduling [77], (iv) makespan minimization of flow shop scheduling [78], (v) p-median location [79], (vi) point feature cartographic label placement [80], (vii) transit route network design [81], and (viii) unsupervised feature selection [22].

The BRKGA [36], which is derived from the random-key genetic algorithm (RKGA) [82], proposes some changes to improve the algorithm's performance and its usability. This algorithm divides the procedures into an independent and dependent part of the problem so that only the dependent part is related to the problem characteristics at hand.

Moreover, the BRKGA creates new individuals by choosing allele-by-allele from the parents, allowing good alleles to be present in the next generation. The selection of parents occurs in the same way as in CGA.

However, in the BRKGA, a percentage of the best individuals necessarily compose the next generation, and mutant individuals compose a predefined percentage of the next generation [36].

The BRKGA has been applied in many types of problems, such as in scheduling [83, 84], orthogonal packing [85, 86], assembly line balancing [87] and manufacturing cell formation [88]. In each of these type of problems, the BRKGA outperformed the GA. Some other issues the BRKGA addressed are: (i) 2D and 3D bin packing [89], (ii) multi-traveling salesmen [90], and (iii) job shop scheduling [91].

There are also some studies that used BRKGA in the data clustering problem, such as in the work of Festa [92], who compared the BRKGA to four variants of the greedy randomized adaptive search procedure with path-relinking (forward, backward, mixed, and randomized). The results showed that BRKGA produced good-quality solutions for all instances [93].

4. UFSPSO

The traditional PSO algorithm [94] creates an initial population randomly, with n_p particles (solutions), evaluates each particle, updates the best solution found by each particle (p_best), and the best solution of the swarm found so far (g_best). From there, the velocity (v) of each particle and its positions (x) are calculated, and finally, the algorithm starts a new iteration. Usually, the algorithm stops when it reaches the maximum of iterations ($maxIter$).

The velocity is updated by equation:

$$v_i(t+1) = w(v_i(t)) + c1(r_1(p_{best}(t) - x(t))) + c2(r_2(g_{best}(t) - x(t))) \quad (1)$$

where $v_i(t+1)$ is the velocity of the particle i in the $t+1$ iteration; w, $c1$, and $c2$ are the coefficient of inertia, the cognitive coefficient, and the social coefficient, respectively, which are user-defined parameters, and r_1 and r_2 are two random values generated from the interval [0,1]. The position of each particle is also updated, summing the previous particle position with the calculated velocity in 1.

The PSO is computationally cheap and easy to implement, and it has few parameters. One of the challenges is to define the coefficients (w, $c1$, $c2$) of equation 1, because they contribute to the convergence and exploration behavior of the solution space.

The UFSPSO follows the same steps described above, except the way the initial population is created, which will be described in the next subsections. Considering this, we show the UFSPSO pseudocode in Algorithm 1.

Algorithm 1: UFSPSO

```
Initialize the parameters (nSelect, k_max, n_p, w, c1, c2 and maxIter);
Build the population, with n_p particles, randomly and deterministically;
while it does not reach the stopping criteria do
    Evaluate each particle in the population;
    Update the p_best and g_best;
    Update the velocity and the positions;
end
```

4.1. Initial population

The initial population is randomly created, except for three individuals, who are generated in a deterministic way, using the LS, UDFS, and VTFS. The LS and the UDFS filter methods issue a rank of attributes, wherein the first attribute is the one that most contributes to the data clustering algorithm, followed by the second, and so on. Each method generates this rank in a different way.

The LS considers locality preserving projection and Laplacian eigenmaps. It is based on the fact that samples with the same class label are often close to each other. Thus, this method evaluates the features by locality preserving power. The UDFS works with discriminant analyses and $\ell_{2,1}$-norm minimization. For this method, the number of classes in a dataset could be fitted by a linear predictor.

In addition to these two particles generated by the LS and the UDFS algorithms, we also generate a third particle using the variance thresholding for feature selection method (VTFS), which does not select attributes that have a smaller variation than a previously defined thresholding, pt.

However, this third particle is only generated if the number of attributes selected by the VTFS is greater than or equal to the relevant feature subset size ($nSelect$), which was previously determined in the UFSPSO. This parameter is randomly defined from the interval $[1,\ p-1]$, where p is the number of attributes of the full dataset.

In the case when the number of attributes selected by the VTFS is less than the the relevant feature subset size, these attributes are considered, but those that are lacking to complete the subset are generated randomly. If repeated attributes are found in the individual string, the VTFS method is no longer considered and therefore, the third particle is randomly generated.

The number of groups is defined randomly from the interval $[2,\ k_{max}]$, where k_{max} is the number of different labels in the dataset. The velocity for the number of groups is defined separately.

4.2. Evaluation

For assessment purposes, we selected the external quality index of the clustering, called the adjusted rand index (ARI). The ARI, which was proposed by Hubert and Levin [95], has been used in many works in unsupervised learning to evaluate the data clustering algorithms results, as can be seen in [96, 97, 98].

The external indices compare the clustering structure C, resulting from a data clustering algorithm, with an independent and predefined partition P of the dataset X with n registers. For this, the index is build considering the following rules, where x_i and x_j are two observations from the dataset X.

Case 1 x_i and x_j belong to the same cluster of C and the same category of P. The number of objects in this case is represented by the letter a.

Case 2 x_i and x_j belong to the same cluster of C, but different categories of P. The number of objects in this case is represented by the letter b.

Case 3 x_i and x_j belong to different clusters of C, but the same category of P. The number of objects in this case is represented by the letter c.

Case 4 x_i and x_j belong to different clusters of C, and different categories of P. The number of objects in this case is represented by the letter d.

The total of points $(n(n-1)/2)$, is denoted as M so that $M = a + b + c + d$. It is also considered that $m_1 = a + b$ and $m_2 = a + c$. Considering the aforementioned rules, the ARI index is defined as:

$$ARI = \frac{Ind - E(Ind)}{max(Ind) - E(Ind)} \tag{2}$$

where Ind is the Rand or Jaccard index, $E(Ind)$ is the expectation of Ind, and $max(Ind)$ is the maximum of Ind. The Rand [99] and Jaccard [100] indices are defined as:

$$R = \frac{(a+d)}{M} \tag{3}$$

$$J = \frac{a}{(a+b+c)} \tag{4}$$

Therefore, the closer the ARI index values are to 1, the higher the quality of the clustering, as opposed to a zero ARI value, which represents poor clustering quality.

4.3. Updating and stopping rule

After evaluating all particles, the velocities (for positions and number of groups) are updated following equation 1, thereafter updating the positions and the number of groups of all particles, by the sum of the new velocity values with the old positions and number of groups, respectively. If the new positions or the new number of groups exceed the acceptable limits, they are taken to the nearest limit.

The UFSPSO stops when it reaches $maxIter$ iterations.

5. UFSBRKGA-I

The traditional GA is inspired by natural evolutionary processes. The algorithm, which was created by Holland [101], is structured into seven basic steps: it creates a population of individuals (initial population), evaluates them (codification and fitness function), selects some of them for reproduction and mutation (selection, crossover, and mutation), and stops when a predetermined criterion is achieved (stopping rules). Thus, GA attempts to find and keep good solutions over the generations, ideally culminating in good results with respect to computational time and solution quality.

The UFSBRKGA-I follows the same structure of the GA; however, it makes some changes in specific steps. In the next subsections, we will describe each one, highlighting our contributions. First, the pseudocode of UFSBRKGA-I is shown in Algorithm 2.

Algorithm 2: UFSBRKGA-I

```
Initialize the parameters n_p, p_e, p_m, p_fs, ρ_s, k_max, maxIter;
Build the population randomly;
Evaluate each individual, considering p_fs to select the attributes;
while it does not reach the stopping criteria, do
    Divide the population into elite (p_e) and non-elite groups (1 − p_e);
    Build the next population, copying elite group (TOP), generating p_m ∗ n_p mutant individuals (BOT), and creating
    ((1 − p_e − p_m) ∗ n_p) new individuals by a crossover process;
    Evaluate each individual, considering p_fs to select the attributes;
end
```

5.1. Initial population, coding, decoding scheme, and fitness function

In this algorithm, the initial population is randomly created. Each individual is represented by a string with p_{attr} positions, where *attr* is the total number of attributes in the full dataset. Each position is filled randomly with a value taken from the uniform interval [0,1].

An example of a coded individual is s = (0.81 0.43 0.54 0.12 0.09 0.16 0.76 0.31), where each position refers to a position of the attribute in the dataset. The number of groups is defined randomly from the interval $[k_{min}, k_{max}]$, where k_{min} and k_{max} are the minimum and the maximum number of groups allowed, respectively.

The decoding process compares each value to a predefined number (ρ_s). The attribute is selected if the value corresponding to its position is greater than or equal to ρ_s. Otherwise, the attribute is not selected to compose the feature subset.

The decoder algorithm is very specific and highly dependent on the problem. Indeed, it is the only link to the problem. The other genetic procedure is independent (see more in Ref. [36]).

As a fitness function, the UFSBRKGA-I considers the same external quality index used in UFSPSO, the ARI index.

5.2. Selection, crossover, mutation, and stopping rule

The first step consists of classifying the individuals from the current population in descending order by their adaptation values, thereafter dividing the population into two groups, wherein the former, called the elite, receives the p_e (percentage) of the best individuals, and the second one, called non-elite, receives the remaining individuals.

Next, a new population is built in three steps by copying the full elite group from the old population to the newest one, renaming it as the TOP group, creating mutant individuals (p_m percent of the entire population), calling these individuals the BOT group, and generating the remaining individuals by the crossover procedure.

The mutant individuals are created as follows. The worst individual from the previous population is selected; thereafter, a random position of this individual is chosen to be substituted by a value generated randomly from the [0,1] interval. The same procedure is executed for all mutant individuals, updating the worst individual to the previous second-worst individual, and so on.

The individuals resulting from the crossover procedure are formed as follows. Two individuals (parent 1 and parent 2) are randomly selected from the previous population so that one comes from the TOP group (parent 1), and the other is selected from the entire population (parent 2). The new individual (son) is built with the help of an auxiliary string, which has the same size as an individual (*attr* positions). Each position (allele) of this string, which is filled with random numbers from the [0,1] interval, is compared to a predetermined parameter, called ρ_s. Thus, the parent 1 allele is selected to compose the child string if the corresponding allele of the auxiliary string is smaller than ρ_s. Otherwise, the corresponding parent 2 allele feeds the child sequence.

The UFSBRKGA-I stops when it achieves *maxIter* iterations.

6. UFSBRKGA-II

The UFSBRKGA-II follows the same steps as the UFSBRKGA-I, except for the way the initial population is created, which will be explained ahead. Algorithm 3 shows the pseudocode of the UFSBRKGA-II.

Algorithm 3: UFSBRKGA-I

```
Initialize the parameters n_p, p_e, p_m, p_fs, ρ_s, k_max, maxIter;
Build the population, with n_p particles, randomly and deterministically;
Evaluate each individual, considering p_fs to select the attributes;
while it does not reach the stopping criteria, do
    Divide the population into elite (p_e) and non-elite group (1 − p_e);
    Build the next population, copying elite group (TOP), generating p_m ∗ n_p mutant individuals (BOT), and creating
    ((1 − p_e − p_m) ∗ n_p) new individuals by a crossover process;
    Evaluate each individual, considering p_fs to select the attributes;
end
```

In this algorithm, approximately half of the initial population ($(n_p/2) - 1$) is randomly created.

Another half of the initial population is generated deterministically. Therefore, we considered three filter feature selection methods, the LS, the UDFS, and the VTFS. In UFSPSO, we also used these methods for the same objective; however, here they will be used in a different way because, in the UFSBRKGA-II, the number of selected attributes is not fixed.

First, we evaluated the clustering considering, the first attribute from the ranked list of attributes issued by the filter methods. Subsequently, in a repetitive procedure, we evaluate the clustering with the first two attributes of the list, and then with the first three, and so on. Thereafter, the selections are sorted in decreasing order. It is this list that contributes to the initial population creation as follows.

The population is divided into quartiles, the first quartile of individuals receives the sorted list of the LS method, one by one, following the order of this list. The second quartile of individuals also receives the selection from the ordered list, but now from the UDFS method. The first individual from the third quartile receives the selection suggested by the VTFS method. The rest of the individuals ($(n_p/2) - 1$) are randomly generated, as already mentioned.

7. Experiments

In this section, we describe the components of the computational experiments carried out here. Besides the software and computation set description, we also present the datasets, and all the methods employed in the comparison tests.

With respect to the data clustering algorithm, we used the *litekmeans* function from the toolbox of Matlab [102, 103, 104, 105] to run the K-means algorithm. It was done for all the methods and datasets. Additionally, all approaches were run 100 times and used the ARI (see in Equation 2) as an objective function.

To statistically compare the methods, we executed the nonparametric test Wilcoxon rank-sum [106] at a 5% significance level, using the *ranksum* function from the toolbox of Matlab (Statistics and Machine Learning Toolbox). The best performing algorithm in each dataset was used considered to carry out the test.

The computational experiments were carried out on Matlab R2015a, on a personal desktop computer with Intel Core i7-4790K CPU @ 4 GHz along with 16 GB of RAM under Windows 10.

7.1. Datasets

We simulated ten datasets to analyze the influence of three main situations, the presence of relevant and irrelevant attributes, and noisy and missing data. All these datasets were generated in RStudio software and are available on our research laboratory website [1] .

[1] http://www.laor.prod.eesc.usp.br/

To simulate datasets with relevant and irrelevant attributes, we created two sets of data with 1,000 records and 50 attributes each, the first having 25 relevant attributes and the second consisting of all relevant attributes. The number of groups (k) was defined as $k = 10$. The relevant attributes were issued by dividing the number of instances in k different Gaussians, which were spaced in 100 points (mean) with the standard deviation defined as 10 for them all. The irrelevant attributes were randomly generated from a uniform distribution, considering the total number of instances. These datasets were called sdr50R25 and sdr50R50, where sdr means simulated relevant dataset, followed by a number X, which means that the dataset has X attributes, of which RY are relevant.

To simulate noisy data, we created four datasets, replacing 30% and 70% of the records of the relevant attributes of the sdr50R25 and sdr50R50 datasets by values from a uniform distribution. Thus, we called them sdn50R25N30, sdn50R25N70, sdn50R50N30, and sdn50R50N70, where sdn means simulated noisy dataset, followed by a number X, which means that the dataset has X attributes, of which RY are relevant, followed by NZ, which means that Z percent of the records of the relevant attributes (Y) is composed by noisy data.

To simulate the missing dataset, we created four datasets, replacing 30% and 70% of the records of the relevant attributes of the sdr50R25 and sdr50R50 datasets by zero. Thus, we called them sdn50R25M30, sdn50R25M70, sdn50R50M30, and sdn50R50M70, where sdn means simulated noisy dataset, followed by a number X, which means that the dataset has X attributes, of which RY are relevant, followed by NZ, which means that Z percent of the records of the relevant attributes (Y) is composed by missing data.

In the generation of all the datasets mentioned above, we shuffled the attributes so that relevant and irrelevant attributes were scattered throughout the dataset. The same procedure was applied to noisy and absent datasets.

All dataset information is shown in Table 1.

7.2. Methods

All methods of comparison tests considered in this paper are detailed below. We selected the first two approaches to be our baseline methods, while the third and sixth ones were selected because they were recently proposed in the literature and worked on subjects similar to the purpose of this article. The GA and PSO methods were used because they are the traditional versions of the meta-heuristics we used in the proposed methods.

No feature selection (NOFS) This is the first baseline method, which selects all features, that is, considers the full dataset.

Variance Thresholding for Feature Selection (VTFS) This is the second baseline method, which selects only the variables that have a variance greater than a predefined threshold (pt).

Table 1: datasets

			instances			features		
dataset	chacacteristic	#groups	#total	#noise	#missing	#total	# relevante	# irrelevante
sdr50R25	25% relevant	10	1000	0	-	50	25	25
sdn50R25N30	30% noisy	10	1000	300	-	50	25	25
sdn50R25N70	70% noisy	10	1000	700	-	50	25	25
sdm50R25M30	30% missing	10	1000	0	300	50	25	25
sdm50R25M70	70% missing	10	1000	0	700	50	25	25
sdr50R50	100% relevant	10	1000	0	-	50	50	0
sdn50R50N30	30% noisy	10	1000	300	-	50	50	0
sdn50R50N70	70% noisy	10	1000	700	-	50	50	0
sdm50R50M30	30% missing	10	1000	0	300	50	50	0
sdm50R50M70	70% missing	10	1000	0	700	50	50	0

Embedded Unsupervised Feature Selection (EUFS) This algorithm, which was proposed by Wang, Tang and Liu [60], is a recent embedded unsupervised feature selection method that embeds feature selection into a clustering algorithm via sparse learning without transformation. We used the implementation of the EUFS offered by the authors[2].

Particle Swarm Optimization (PSO) This traditional version was modeled for the unsupervised feature selection problem, considering a fixed and previously defined size of the relevant feature subset, as was modeled in UFSPSO (see in Section 4).

Unsupervised Feature Selection Particle Swarm Optimization (UFSPSO) This is the first proposed algorithm, which considers the construction of the initial population in a partially deterministic way (see more in Section 4).

Genetic Algorithm (GA) This traditional version was modeled for the unsupervised feature selection problem to make the comparison possible. The individuals were encoded by binary code, where each position was randomly generated. In the crossover step, the parents and the two cut points were randomly selected. Regarding the mutation step, we chose one cut-point method. The parameter *ppm* defines the percentage of mutant individuals in the population, so that its value is defined for each dataset (see in Section 7.3).

Constructive Genetic Algorithm (CGA) This algorithm, recently proposed by Martarelli and Nagano

[2]http://www.public.asu.edu/swang187/

[22], is a constructive version of GA. In this paper, we defined $f(s_i)$ and $g(s_i)$ as:

$$f(s_i) = \frac{1}{ARI_{max} * 100} \tag{5}$$

$$g(s_i) = \frac{1}{ARI * 100} \tag{6}$$

where ARI_{max} is one since it is considered the maximum value of this index. These choices were made because $g(s_i)$ must be less than $f(s_i)$ for every s_i. Thus, for every s_i, the inequality $g(s_i) \geq f(s_i)$ is valid.

Unsupervised Feature Selection Biased Random-Key Genetic Algorithm (UFSBRKGA-I) This is the second proposed algorithm, in which we modeled the BRKGA for the unsupervised feature selection problem. In this algorithm, we built the initial population in a random way (see more in Section 5).

Unsupervised Feature Selection Biased Random-Key Genetic Algorithm (UFSBRKGA-II) Finally, this is the third proposed algorithm, which was modeled in the same way as we did in the UFSBRKGA-I, except for the way the initial population is created; that is, here the initial population was partially generated in a deterministic manner (see more in Section 6).

7.3. Parameters

The parameters were defined exhaustively, which means we defined a range of values for each one, and then we evaluated all the possibilities, choosing the ones that produced the best results.

For all algorithms that require the number of individuals in the initial population and the number of iterations, we set $n_p = 30$ and $maxIter = 15$. In all algorithms and datasets, the number of groups (k) was generated randomly, from the interval [k_{min}, k_{max}], where k_{min} is 2 and k_{max} is the number of different labels in the dataset.

The thresholding parameter for the VTFS was set after evaluating all the possibilities from the interval pt = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]. Table 2 brings the best thresholding value for each dataset.

For the GA, we set the following parameters for all datasets, $p_c = 0.8$, $p_m = 0.2$, and $ppm = 0.2$.

The CGA received the following parameters, for all datasets, $p_{base} = 0.3$, $p_k = 0.8$, $p_{mut} = 1$, $alw_{sC} = yes$, $p_(Cht) = 0.3$, $corr_{min} = 0.2$, $corr_{max} = 0.7$, $p_{bl} = 0.4$ (except for the sdm50R50M30 dataset, in which it is 0.2), $p_{bm} = 0.6$ (except for the sdm50R50M30 dataset, in which it was 0.5), and $p_{bh} = 0.8$. The other parameters are in the Table 2. All the mentioned parameters are properly explained and defined in Martarelli and Nagano [22].

To define the two parameters of the EUFS (λ and α), we set two intervals, which contained values that were suggested by the authors. Both intervals are $\lambda = [10^{-6}, 10^{-4}, 10^{-2}, 10, 10^2, 10^4, 10^6]$, and $\alpha = [10^{-6}, 10^{-4}, 10^{-2}, 10, 10^2, 10^4, 10^6]$. After all the possibilities tested, we defined the value for each parameter, as can be seen in Table 2.

The PSO and the UFSPSO requests three main parameters (w, pcc, psc), which were defined after all the possibilities tested, following the values from the intervals: $w = [0.5, 0.8, 1.0, 1.5, 2.0]$, $pcc = [1.5, 1.8, 2.0, 2.3, 2.5]$, $psc = [1.5, 1.8, 2.0, 2.3, 2.5]$. Table 2 shows the best parameters for both algorithms.

Regarding the parameters for the LS and UDFS methods, for all datasets, we set the k-nearest neighbor to construct the graph equal to 5, and use the cosine dissimilarity to assign weights for each edge in the graph. To carry out the UDFS, we used the UDFS toolbox [102, 103, 105, 104], and we used the LS for feature selection provided by He, Cai and Niyogi[55]. The parameter $nClass$ is 5 for all datasets.

For UFSBRKGA-I and UFSBRKGA-II, we set the following parameters, for all datasets: $p_e = 0.3$, $p_m = 0.3$, $\rho_s = 0.8$, and $p_{fs} = 0.5$.

Table 2: The parameter settings

dataset	VTFS	EUFS		CGA			PSO			UFSPSO		
	pt	λ	α	ϵ	$\rho_{g_{max}}$	d	w	pcc	psc	w	pcc	psc
sdr50R25	0.20	10^6	10^{-6}	0.10	0.60	0.60	0.80	1.50	1.50	0.50	1.50	1.50
sdn50R25N30	0.60	10^1	10^1	0.10	0.40	0.60	2.00	2.30	2.30	2.00	2.50	1.80
sdn50R25N70	0.50	10^2	10^1	0.10	0.40	0.60	2.00	2.30	1.80	0.80	2.30	2.00
sdm50R25M30	0.40	10^{-2}	10^2	0.05	0.50	0.60	1.50	2.30	2.30	0.50	2.30	2.30
sdm50R25M70	0.30	10^4	10^{-2}	0.05	0.20	0.40	0.50	2.30	1.50	1.50	1.50	2.30
sdr50R50	0.80	10^6	10^{-6}	0.10	0.60	0.60	0.80	2.3,	2.30	0.50	1.50	1.50
sdn50R50N30	0.50	10^2	10^{-4}	0.10	0.40	0.60	1.00	2.00	2.30	1.00	2.30	2.00
sdn50R50N70	0.40	10^{-4}	10^{-6}	0.10	0.40	0.60	1.00	1.50	2.00	1.00	1.80	2.00
sdm50R50M30	0.40	10^{-2}	10^2	0.05	0.50	0.60	1.50	2.30	2.30	0.50	2.30	2.30
sdm50R50M70	0.30	10^4	10^{-2}	0.05	0.20	0.40	0.50	2.30	1.50	1.50	1.50	2.30

8. Results and discussion

As a result of 100 repetitions, we obtained the ARI values, execution times, number of relevant features selected, and number of groups selected for each algorithm and dataset. To summarize these results, we calculated the average and standard deviations for ARI values and execution time values, as can be seen in

Tables 3 and 4, respectively. We also provided Figures 1 and 2 to show graphically the average of number of relevant features and groups selected, respectively.

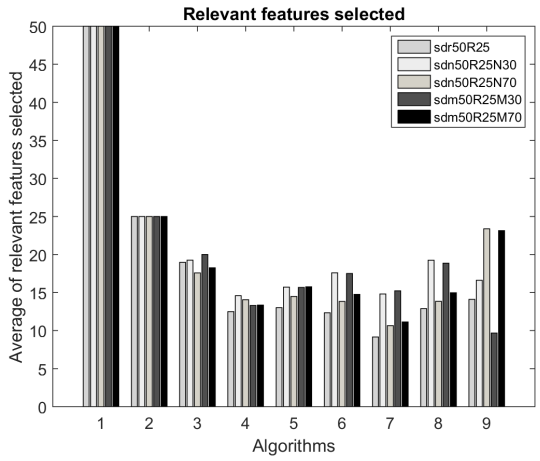
Moreover, to obtain a better understanding with respect to the convergence of the algorithms over the 15 iterations, we created the convergence curves (Figures 3 - 5) for all those methods that worked with interactions, which are all of them except NOFS, VTFS, and EUFS. Note the vertical axis is on the logarithmic scale.

Table 3: Average and the standard deviation (avg±std) of ARI value of all methods, in all datasets. The best results are highlighted in bold (the higher the better)

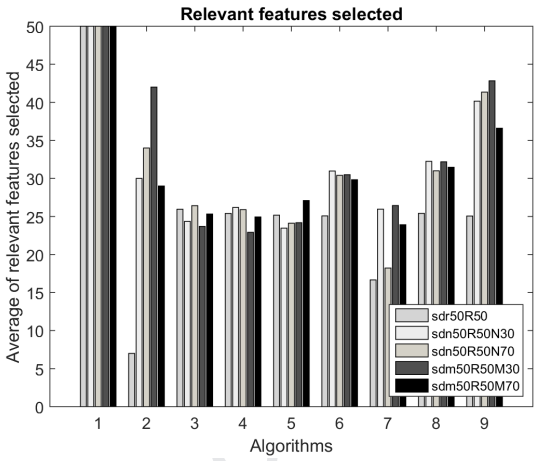
dataset	Algorithms								
	NOFS	VTFS	EUFS	PSO	UFSPSO	GA	CGA	UFSBRKGA-I	UFSBRKGA-II
sdr50R25	0.551±0.210	0.569±0.193	0.558±0.218	0.991±0.030	**1.000±0.000**	0.987±0.037	0.983±0.039	0.987±0.035	0.990±0.031
sdn50R25N30	0.303±0.064	0.304±0.051	0.272±0.072	0.349±0.047	0.362±0.049	0.325±0.009	0.301±0.013	0.343±0.010	**0.395±0.023**
sdn50R25N70	0.116±0.023	0.115±0.022	0.102±0.029	0.139±0.026	0.150±0.015	0.142±0.004	0.133±0.006	0.150±0.004	**0.163±0.005**
sdm50R25M30	0.289±0.061	0.298±0.060	0.267±0.068	0.344±0.046	0.354±0.040	0.326±0.010	0.306±0.014	0.340±0.009	**0.400±0.028**
sdm50R25M70	0.117±0.020	0.112±0.019	0.106±0.031	0.138±0.031	0.151±0.026	0.145±0.005	0.143±0.004	0.153±0.005	**0.168±0.007**
sdr50R50	0.601±0.199	0.602±0.199	0.538±0.210	0.989±0.032	**1.000±0.000**	0.988±0.034	0.986±0.036	0.990±0.031	0.983±0.041
sdn50R50N30	0.353±0.089	0.312±0.067	0.284±0.090	0.390±0.089	0.383±0.092	0.414±0.014	0.372±0.014	0.432±0.015	**0.465±0.023**
sdn50R50N70	0.150±0.030	0.129±0.022	0.116±0.036	0.163±0.051	0.162±0.052	0.174±0.007	0.132±0.020	0.181±0.008	**0.199±0.011**
sdm50R50M30	0.368±0.067	0.352±0.077	0.273±0.085	0.377±0.086	0.393±0.089	0.411±0.014	0.376±0.017	0.437±0.019	**0.492±0.012**
sdm50R50M70	0.156±0.036	0.124±0.027	0.115±0.044	0.167±0.044	0.177±0.046	0.179±0.008	0.155±0.009	0.189±0.008	**0.203±0.008**

Table 4: Average and the standard deviation (avg±std) of execution time (in seconds) of all methods, in all datasets. The best results are highlighted in bold (the smaller the better)

dataset	Algorithms								
	NOFS	VTFS	EUFS	PSO	UFSPSO	GA	CGA	UFSBRKGA-I	UFSBRKGA-II
sdr50R25	**0.009±0.003**	**0.009±0.003**	0.143±0.027	3.954±0.309	5.251±0.153	3.932±0.292	5.251±0.580	4.439±0.183	4.645±0.207
sdn50R25N30	0.015±0.006	**0.014±0.005**	0.186±0.052	5.678±0.713	8.454±0.745	5.971±0.418	2.186±0.252	6.437±0.365	6.431±0.404
sdn50R25N70	0.014±0.005	**0.013±0.005**	0.196±0.049	5.693±0.625	8.179±0.502	5.866±0.368	1.695±0.217	6.434±0.201	6.712±0.267
sdm50R25M30	**0.014±0.006**	**0.014±0.006**	0.180±0.053	5.558±0.674	8.414±0.659	5.975±0.444	1.737±0.502	6.381±0.408	6.339±0.413
sdm50R25M70	0.014±0.005	**0.013±0.005**	0.199±0.054	5.559±0.638	7.191±0.545	6.045±0.451	6.846±0.925	6.459±0.221	6.882±0.281
sdr50R50	**0.009±0.003**	**0.009±0.003**	0.134±0.034	3.714±0.257	6.071±0.316	4.139±0.328	5.240±0.688	4.415±0.230	4.402±0.225
sdn50R50N30	**0.012±0.005**	**0.012±0.005**	0.190±0.062	5.580±0.558	8.032±0.585	5.844±0.462	2.090±0.285	5.990±0.361	5.935±0.403
sdn50R50N70	**0.012±0.004**	**0.012±0.004**	0.221±0.061	5.536±0.532	7.716±0.686	5.924±0.400	0.621±0.121	6.390±0.348	6.503±0.371
sdm50R50M30	**0.012±0.004**	**0.012±0.004**	0.198±0.065	5.621±0.631	7.924±0.627	5.654±0.400	2.316±0.641	5.982±0.366	5.965±0.373
sdm50R50M70	0.013±0.005	**0.011±0.004**	0.232±0.069	5.783±0.562	6.989±0.469	5.912±0.376	3.409±0.550	6.270±0.277	6.173±0.321

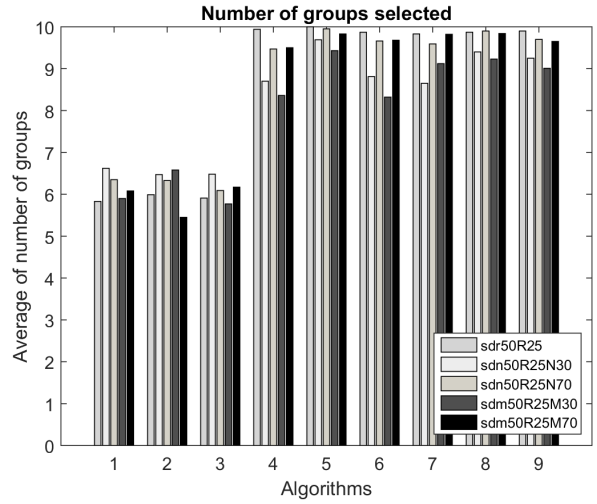


(a) All datasets with 25 relevant attributes

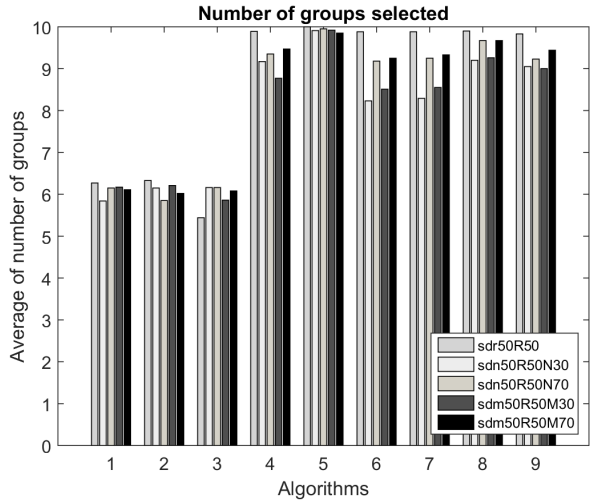


(b) All datasets with 50 relevant attributes

Figure 1: Average of the number of relevant features selected by each algorithm in all datasets, where 1-NOFS, 2-VTFS, 3-EUFS, 4-PSO, 5-UFSPSO, 6-GA, 7-CGA, 8-UFSBRKGA-I and 9-UFSBRKGA-II

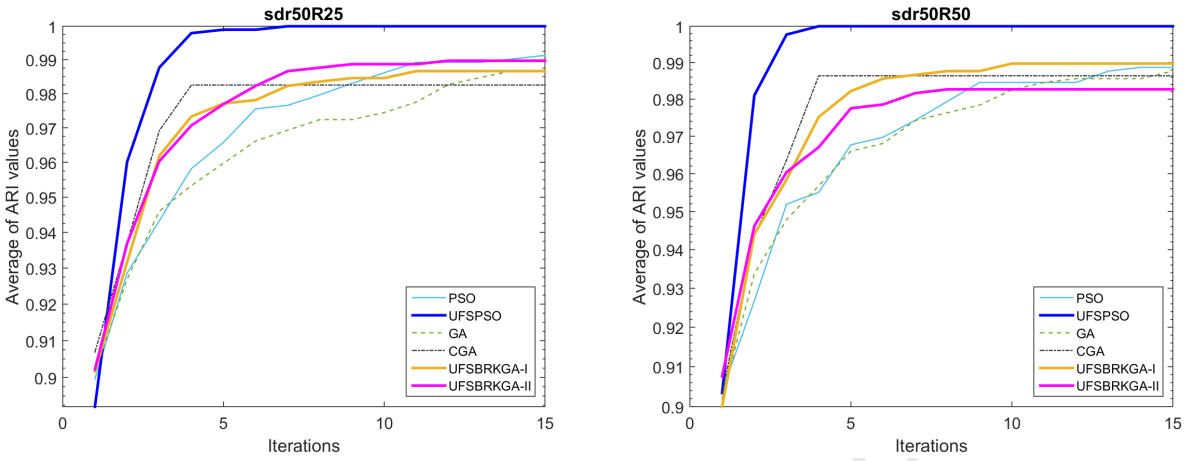


(a) All datasets with 25 relevant attributes



(b) All datasets with 50 relevant attributes

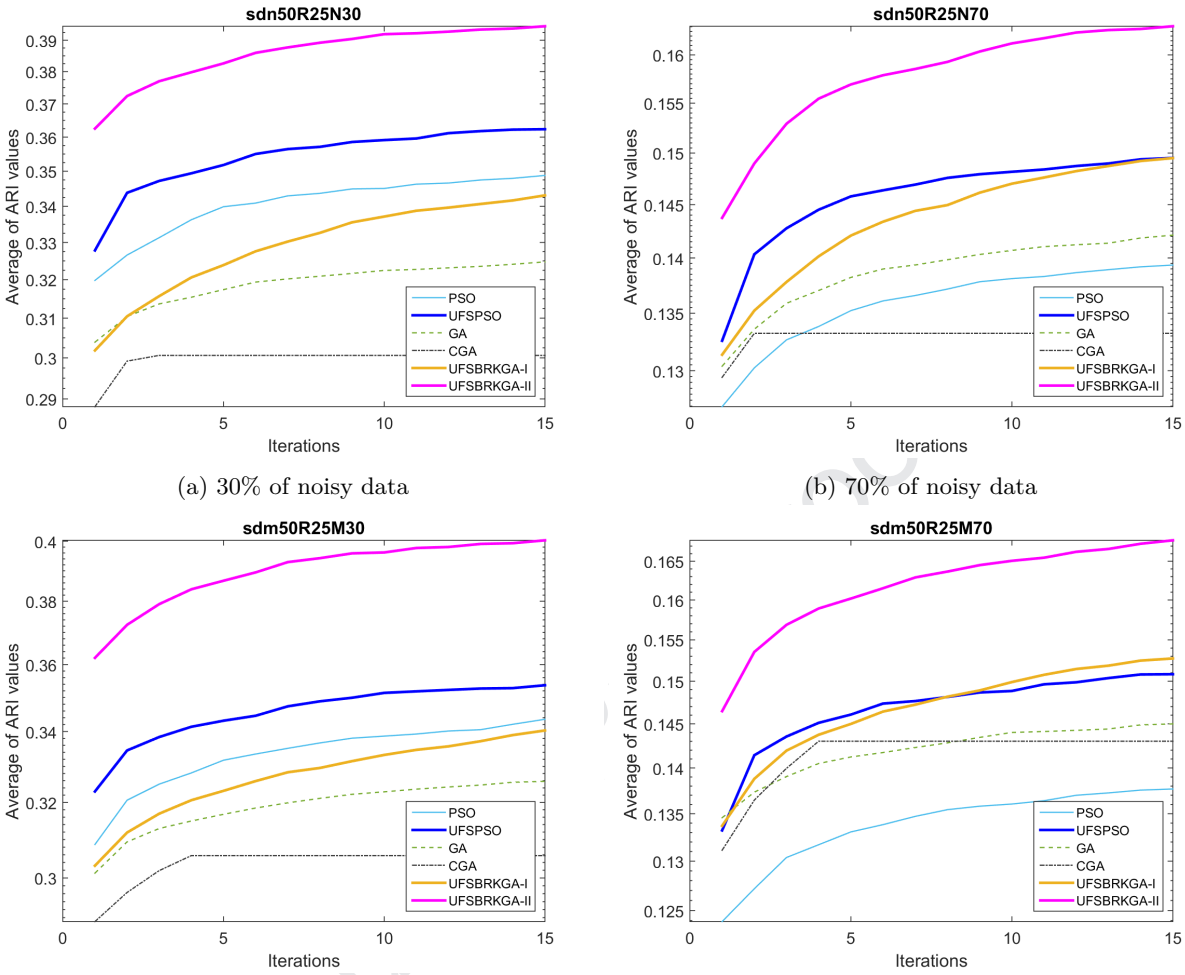
Figure 2: Average of the number of groups selected by each algorithm in all datasets, where 1-NOFS, 2-VTFS, 3-EUFS, 4-PSO, 5-UFSPSO, 6-GA, 7-CGA, 8-UFSBRKGA-I and 9-UFSBRKGA-II



(a) 25 relevant attributes

(b) 50 relevant attributes

Figure 3: Convergence curves for datasets with relevant and irrelevant attributes



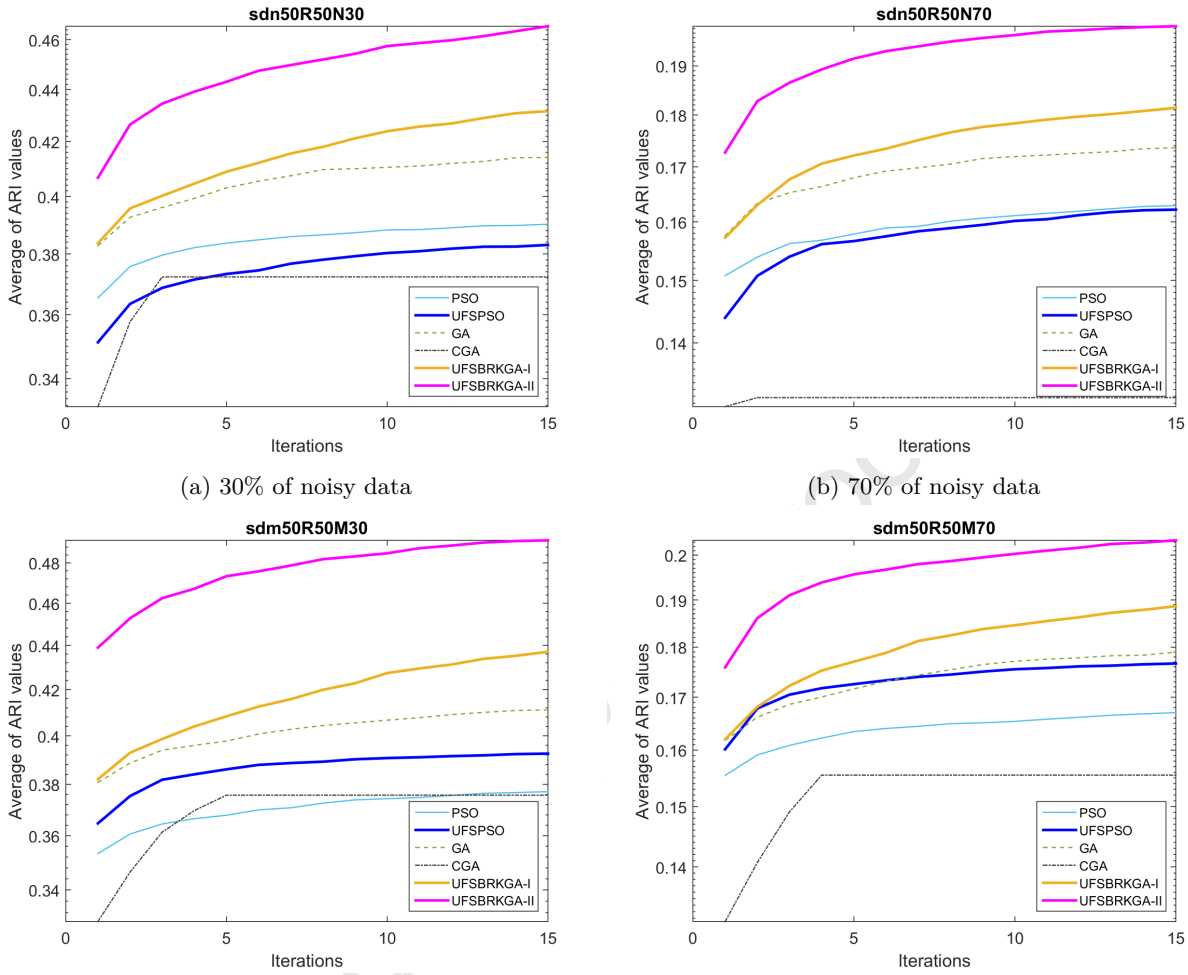
(a) 30% of noisy data

(b) 70% of noisy data

(c) 30% of missing data

(d) 70% of missing data

Figure 4: Convergence curves for datasets with 25 relevant attributes with different levels of noisy and missing data



(a) 30% of noisy data

(b) 70% of noisy data

(c) 30% of missing data

(d) 70% of missing data

Figure 5: Convergence curves for datasets with 50 relevant attributes with different levels of noisy and missing data

In order to verify the results statistically, we used the Wilcoxon rank-sum test, at a 5% of significance level, on the best algorithm in each dataset. Table 5 shows the results.

Table 5: The p-values of the Wilcoxon rank-sum test ($p \geq 0.05$ are not significant). The best algorithm (highlighted in bold) for each dataset was tested against other algorithms

dataset	**Algorithms**								
	NOFS	VTFS	EUFS	PSO	UFSPSO	GA	CGA	UFSBRKGA-I	UFSBRKGA-II
sdr50R25	2.54E-39	2.65E-39	9.70E-39	2.02E-03	**1.00E+00**	1.85E-04	8.65E-06	1.01E-04	6.12E-04
sdn50R25N30	6.93E-33	8.81E-34	1.72E-34	1.63E-16	7.06E-11	1.23E-34	1.24E-34	1.35E-34	**5.00E-01**
sdn50R25N70	1.10E-33	3.40E-34	1.37E-34	4.68E-25	8.86E-15	3.94E-34	1.25E-34	8.53E-31	**5.00E-01**
sdm50R25M30	4.24E-34	1.54E-33	2.45E-34	3.27E-18	1.08E-17	1.24E-34	1.26E-34	1.25E-34	**5.00E-01**
sdm50R25M70	1.79E-33	8.07E-34	3.59E-34	1.94E-23	2.20E-10	1.29E-33	2.85E-34	1.23E-27	**5.00E-01**
sdr50R50	2.64E-39	2.67E-39	2.56E-39	3.37E-04	**1.00E+00**	1.85E-04	1.01E-04	6.12E-04	1.62E-05
sdn50R50N30	2.04E-26	1.78E-34	3.30E-32	9.01E-09	3.47E-09	2.51E-30	1.34E-34	7.63E-21	**5.00E-01**
sdn50R50N70	4.81E-24	2.44E-34	4.43E-34	8.09E-08	2.49E-06	3.57E-31	1.24E-34	6.23E-23	**5.00E-01**
sdm50R50M30	1.45E-32	1.92E-34	1.34E-34	8.61E-22	1.01E-16	1.35E-34	1.25E-34	2.29E-33	**5.00E-01**
sdm50R50M70	6.04E-18	1.27E-34	1.57E-31	1.17E-08	4.29E-05	2.16E-32	1.24E-34	3.45E-23	**5.00E-01**

The statistical tests confirmed that the performance of the algorithms is statistically different. With respect to the ARI values shown in Table 3, we can see that the UFSPSO achieved the best ARI value in all repetitions (standard deviation is zero) in datasets with relevant and irrelevant attributes (sdr50R25 and sdr50R50). However, when noisy and missing data were present, the UFSBRKGA-II achieved the best ARI values.

It is interesting to note that the CGA did not perform well in the datasets with noisy and missing data. In our view, this happened for two reasons. The first is that the way individuals are constructed in the initial population of CGA does not meet noisy and missing dataset characteristics, causing the algorithm to begin the search at bad points. The other reason is the population is quickly zeroed (influenced by the bad initial points) because in this approach the population size is dynamic and controlled by a parameter of evolution that is eliminating individuals with ARI values less than an expected value for that iteration.

Thus, the CGA starts the search at a disadvantage, and thereafter the population is emptied quickly. This behavior can be seen in the convergence curves (Figures 3-5), where the performance of the CGA is stagnant after the first iterations. We defined that if populations were zeroed, the ARI value used would be the last ARI value calculated when the population still had at least one individual.

Regarding the number of relevant features selected (Figure 1), it is possible to observe that, in general, the algorithms converged to the true number of relevant features in datasets with 25 relevant attributes, (Figure 1a) except the NOFS that always works with the full dataset. However, in datasets with 50 relevant attributes (Figure 1b), only UFSBRKGA-II converged to the true number.

With respect to the number of groups, all algorithms for all datasets achieved closer values of 10, the true number of groups in all datasets, as can be seen in Figure 2.

8.1. BRKGA for unsupervised feature selection problem

The use of BRKGA for the unsupervised feature selection problem showed good results. As can be seen in Table 3 and in Figures 3-5 UFSBRKGA-I and II obtained good ARI values, standing out in the comparison tests.

In our opinion, the good performance of UFSBRKGA methods is primarily related to the crossover procedure and the way the populations are created over generations. First, the crossover process analyzes allele-to-allele from the individual, allowing the promising ones to compose future individuals, unlike what occurs in the traditional GA version (or even in CGA), in which a block of alleles from the parents builds a new individual. In additional, populations are built by maintaining the best individuals over the generations (TOP group) and adding a predefined number of mutant individuals (BOT group), offering the optimizer a balance between diversification and intensification in all generations (iterations).

Regarding the execution time, the two BRKGA-based algorithms took less computational time than the UFSPSO, although they achieved similar times to that achieved by GA and longer than those of the other approaches. However, it can be stated that, in comparison to the evolutionary and swarm algorithms, the UFSBRKGA-I and II obtained execution times close to these approaches, with the exception of the CGA, which obtained a runtime far below the others because it had its population zeroed since the first iterations.

8.2. The influence of the initial population

The results indicated that there is a positive influence on the performance of the algorithms when the initial population is partially constructed in a deterministic way. As imagined, individuals created by filtering methods offered the optimizer a competitive advantage by starting the search from promising points. Future work is needed to determine how different numbers of deterministic and random individuals in the population affect the performance of the algorithm.

It is interesting to note that, in general, UFSPSO took about 2 seconds longer than PSO to execute the algorithm. However, the runtime of UFSBRKGA-I and UFSBRKGA-II was very similar. We believe this occurred because the PSO modeling considered a predetermined size for the subset of relevant attributes, and thereby, the operation to create the individuals by filter methods was more costly because more tests were necessary to ensure that the individual was filled correctly.

8.3. The influence of relevant and irrelevant attributes

As shown in Figure 3, all algorithms performed well on both sets of data that simulated relevant and irrelevant attributes, although the UFSPSO stood out, as confirmed in the statistical tests (see in Table 5).

It is important to note that UFSBRKGA-II ranked last in the sdn50R50 dataset, suggesting that the number of individuals generated deterministically in the initial population may have impaired the algorithm, because in UFSBRKGA-II, half of the initial population is composed of individuals deterministically generated, and in UFSPSO only 3 individuals (out of 30) are generated in this way. More studies are needed to confirm this suspicion.

8.4. The influence of noisy and missing data

The presence of noisy and missing data in datasets significantly impairs the performance of most algorithms, especially PSO, UFSPSO, and CGA. However, both BRKGA-based algorithms stand out significantly as the level of noisy and missing data increases.

It is important to note that the more relevant attributes there are in a dataset, the more noisy and missing data will be in it. This is because noisy and missing data are only inserted into relevant attributes.

To note this, let us look at the convergence curves of PSO and BRKGA-based algorithms in noisy datasets, following the sequence of Figures 3a, 3b, 4a, 4b, 5a, 5b.

From this analysis, we can identify how sensitive PSO-based approaches are to these types of data, while we can also see the BRKGA-based algorithms become powerful as the level of noisy and missing data increases.

We believe that this behavior is related to the characteristics of BRKGA, such as the crossover step, construction of new populations, and the fact that half of the initial population of UFSBRKGA-II is composed of deterministically created individuals.

In respect to runtime, Table 4 shows that the computational time slightly increases as noisy and missing data level rises.

9. Conclusion

In this paper, three unsupervised feature selection algorithms were proposed. The first, called UFSBRKGA-I, aimed to model the BRKGA approach for this problem. The second one, called UFSBRKGA-II, implemented a modification in the building of the initial population of UFSBRKGA-I. Finally, the third one, called UFSPSO, applied the same idea of the initial population construction in the PSO algorithm.

Moreover, this work simulated ten datasets to investigate the influence of relevant and irrelevant attributes, as well as the influence of noisy and missing data in relevant attributes.

Our findings showed that the BRKGA-based algorithms achieved excellent results with noisy and missing data, specifically UFSBRKGA-II, which created part of the individuals of the initial population deterministically, was the best algorithm among the competitors that obtained the best values for the external quality index of the clustering, ARI.

UFSPSO achieved the best ARI value (1) in all repetitions (standard deviation was zero) in datasets with relevant and irrelevant attributes, without any noisy and missing data.

This work evidences the importance of the preprocessing task, called dimensionality reduction, not only to improve the performance of pattern recognition algorithms but also to provide an opportunity to analyze the results generated by the knowledge extraction process in databases more correctly, since irrelevant attributes can distort such results.

As future work, we suggest applying other clustering quality indices, such as the Davies–Bouldin index, and silhouette index. Additionally, it would be interesting to develop a hybrid method considering UFSPSO and UFSBRKGA-II for this problem, as well as analyze the proportion of deterministic and random individuals in the initial population.

10. Acknowledgments

We would like to thank the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) of Brazil for the financial support received (grant numbers 448161/2014-1, 308047/2014-1, 1633193/2016-2, and 306075/2017-2) and express our gratitude to the anonymous reviewers for their rich contributions.

References

[1] A. K. Jain, R. C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Upper Saddle River, NJ, USA, 1988.

[2] H.-H. Bock, Origins and extensions of the k-means algorithm in cluster analysis., Journal Électronique d'Histoire des Probabilités et de la Statistique [electronic only] 4.

[3] A. Saxena, M. Prasad, A. Gupta, N. Bharill, O. P. Patel, A. Tiwari, M. J. Er, W. Ding, C.-T. Lin, A review of clustering techniques and developments, Neurocomputing 267 (6) (2017) 664–681. doi:10.1016/j.neucom.2017.06.053.

[4] R. Xu, D. Wunsch, Clustering, Wiley-IEEE Press, New Jersey, USA, 2009.

[5] V. Roth, T. Lange, Feature selection in clustering problems, in: S. Thrun, L. Saul, L. Schölkopf (Eds.), Advances in Neural Information Processing Systems 16, MIT Press, Cambridge, MA, 2003, pp. 1–8.

[6] S. Alelyani, J. Tang, H. Liu, Feature selection for clustering: A review, in: Data Clustering: Algorithms and Applications, 2013.

[7] V. Kumar, S. Minz, Feature selection: A literature review, Smart Computing Review 4 (2014) 211–229.

[8] B. Xue, M. Zhang, W. N. Browne, X. Yao, A survey on evolutionary computation approaches to feature selection, IEEE Transactions on Evolutionary Computation 20 (4) (2016) 606–626. doi:10.1109/TEVC.2015.2504420.

[9] R. Kohavi, G. H. John, Wrappers for feature subset selection, Artificial Intelligence 97 (1-2) (1997) 273–324. doi:10.1016/S0004-3702(97)00043-X.

[10] U. Stańczyk, Feature Evaluation by Filter, Wrapper, and Embedded Approaches, Springer Berlin Heidelberg, Berlin, Heidelberg, 2015, Ch. Feature Evaluation by Filter, Wrapper, and Embedded Approaches, pp. 29–44. doi:10.1007/978-3-662-45620-0-3.

[11] L. Wang, Y. Wang, Q. Chang, Feature selection methods for big data bioinformatics: A survey from the search perspective, Methods 111 (2016) 21–31. doi:10.1016/j.ymeth.2016.08.014.

[12] A. L. Blum, P. Langley, Selection of relevant features and examples in machine learning, Artificial Intelligence 97 (1997) 245–271.

[13] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, Journal of Machine Learning Research 3 (2003) 1157–1182.

[14] G. Chandrashekar, F. Sahin, A survey on feature selection methods, Computers and Electrical Engineering 40 (1) (2014) 16–28. doi:10.1016/j.compeleceng.2013.11.024.

[15] S. Solorio-Fernández, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, A review of unsupervised feature selection methods, Artificial Intelligence Review 1 (2019) 1573–7462. doi:10.1007/s10462-019-09682-y.

[16] W. Duch, Filter Methods, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, Ch. Filter Methods, pp. 89–117. doi:10.1007/978-3-540-35488-8-4.

[17] N. Sánchez-Maroño, A. Alonso-Betanzos, M. Tombilla-Sanromán, Filter Methods for Feature Selection - A Comparative Study, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, Ch. Filter Methods for Feature Selection - A Comparative Study, pp. 178–187. doi:10.1007/978-3-540-77226-2-19.

[18] P. Shamsinejadbabki, M. Saraee, A new unsupervised feature selection method for text clustering based on genetic algorithms, Journal of Intelligent Information Systems 38 (3) (2012) 669–684. doi:10.1007/s10844-011-0172-5.

[19] L. M. Abualigah, A. T. Khader, M. A. Al-Betar, Unsupervised feature selection technique based on genetic algorithm for improving the text clustering, in: 7th International Conference on Computer Science and Information Technology (CSIT), 2016, pp. 1–6. doi:10.1109/CSIT.2016.7549453.

[20] T.-P. Hong, C.-H. Chen, F.-S. Lin, Using group genetic algorithm to improve performance of attribute clustering, Applied Soft Computing 29 (2015) 371–378. doi:10.1016/j.asoc.2015.01.001.

[21] K. Kouser, A. Priyam, Feature selection using genetic algorithm for clustering high dimensional data, International Journal of Engineering and Technology 7 (2.11) (2018) 27–30. doi:10.14419/ijet.v7i2.11.11001.

[22] N. J. Martarelli, M. S. Nagano, A constructive evolutionary approach for feature selection in unsupervised learning, Swarm and Evolutionary Computation 42 (1) (2018) 125–137.

[23] M. Anusha, J. G. R. Sathiaseelan, Feature selection using k-means genetic algorithm for multi-objective optimization, Procedia Computer Science 57 (1) (2015) 1074–1080. doi:10.1016/j.procs.2015.07.387.

[24] H. Hannah Inbarani, P. K. Nizar Banu, A. T. Azar, Feature selection using swarm-based relative reduct technique for fetal heart rate, Neural Computing and Applications 25 (3) (2014) 793–806. doi:10.1007/s00521-014-1552-x.

[25] M. Yuwono, Y. Guo, J. Wall, J. Li, S. West, G. Platt, S. W. Su, Unsupervised feature selection using swarm intelligence and consensus clustering for automatic fault detection and diagnosis in heating ventilation and air conditioning systems, Applied Soft Computing 34 (2015) 402–425. doi:https://doi.org/10.1016/j.asoc.2015.05.030.

[26] P. S. Deepthi, S. M. Thampi, Unsupervised gene selection using particle swarm optimization and k-means, in: Proceedings of the Second ACM IKDD Conference on Data Sciences, 2015, pp. 134–135. doi:10.1145/2732587.2732615.

[27] N. Kushwaha, M. Pant, Link based bpso for feature selection in big data text clustering, Future Generation Computer Systems 82 (2018) 190–199. doi:10.1016/j.future.2017.12.005.

[28] L. M. Abualigah, A. T. Khader, E. S. Hanandeh, A new feature selection method to improve the document clustering using particle swarm optimization algorithm, Journal of Computational Science 25 (2018) 456–466. doi:10.1016/j.jocs.2017.07.018.

[29] H. B. Nguyen, B. Xue, P. Andreae, Pso with surrogate models for feature selection: static and dynamic clustering-based methods, Memetic Computing 10 (3) (2018) 291–300. doi:10.1007/s12293-018-0254-9.

[30] A. S. S. Rani, R. R. Rajalaxmi, Unsupervised feature selection using binary bat algorithm, in: 2nd International Conference on Electronics and Communication Systems (ICECS), 2015, pp. 451–456. doi:10.1109/ECS.2015.7124945.

[31] V. Kumar, J. K. Chhabra, D. Kumar, Automatic unsupervised feature selection using gravitational search algorithm, IETE Journal of Research 61 (1) (2015) 22–31. doi:10.1080/03772063.2014.987702.

[32] S. Saha, R. Spandana, A. Ekbal, S. Bandyopadhyay, Simultaneous feature selection and symmetry based clustering using

multiobjective framework, Applied Soft Computing 29 (2015) 479–486. doi:10.1016/j.asoc.2014.12.009.

[33] R. R. Ramasamy, S. S. A. Rani, Modified binary bat algorithm for feature selection in unsupervised learning, The International Arab Journal of Information Technology 15 (2018) 1060–1067.

[34] S. Jameel, S.-U. Rehman, An optimal feature selection method using a modified wrapper-based ant colony optimisation, Journal of the National Science Foundation of Sri Lanka 46 (2) (2018) 143–151. doi:10.4038/jnsfsr.v46i2.8414.

[35] J. Prakash, P. K. Singh, Gravitational search algorithm and k-means for simultaneous feature selection and data clustering: a multi-objective approach, Soft Computing 23 (6) (2019) 2083–2100. doi:10.1007/s00500-017-2923-x.

[36] J. F. GonÇalves, M. G. C. Resende, Biased random-key genetic algorithms for combinatorial optimization, Journal of Heuristics 17 (2011) 487–525. doi:10.1007/s10732-010-9143-1.

[37] P. Moradi, M. Rostami, Integration of graph clustering with ant colony optimization for feature selection, Knowledge-Based Systems 84 (1).

[38] I. Guyon, A. Elisseeff, An Introduction to Feature Extraction, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, Ch. An Introduction to Feature Extraction, pp. 1–25. doi:10.1007/978-3-540-35488-8-1.

[39] M. S. Nixon, A. S. Aguado, Chapter 4 - low-level feature extraction, in: M. S. Nixon, A. S. Aguado (Eds.), Feature Extraction and Image Processing for Computer Vision, third edition Edition, Academic Press, Oxford, 2012, pp. 137–216. doi:10.1016/B978-0-12-396549-3.00004-5.

[40] I. T. Jolliffe, Principal Component Analysis, 1986.

[41] R. L. Gorsuch, Factor Analysis, L. Erlbaum Associates, 1983.

[42] P. J. Huber, Projection pursuit, The Annals of Statistics 13 (2) (1985) 435–475. doi:10.1214/aos/1176349519.

[43] J. H. Friedman, Exploratory projection pursuit, Journal of the American Statistical Association 82 (397).

[44] G. H. Golub, C. Reinsch, Singular value decomposition and least squares solutions, Numerische Mathematik 14 (5) (1970) 403–420. doi:10.1007/BF02163027.

[45] R. A. Fisher, The use of multiple measurements in taxonomic problems, Annals of Eugenics.

[46] N. Kwak, C.-H. Choi, Input feature selection for classification problems, IEEE Transactions on Neural Networks 13 (1) (2002) 143–159. doi:10.1109/72.977291.

[47] C. Lazar, J. Taminau, S. Meganck, D. Steenhoff, A. Coletta, C. Molter, V. d. Schaetzen, R. Duque, H. Bersini, A. Nowe, A survey on filter techniques for feature selection in gene expression microarray analysis, IEEE/ACM Transactions on Computational Biology and Bioinformatics 9 (4) (2012) 1106–1119. doi:10.1109/TCBB.2012.33.

[48] R. Battiti, Using mutual information for selecting features in supervised neural net learning, IEEE Transactions on Neural Networks 5.

[49] P. Pudil, J. Novovičová, J. Kittler, Floating search methods in feature selection, Pattern Recognition Letters 15 (1994) 1119–1125. doi:10.1016/0167-8655(94)90127-9.

[50] T. Rückstieß, C. Osendorfer, P. Van Der Smagt, Sequential Feature Selection for Classification, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, Ch. Sequential Feature Selection for Classification, pp. 132–141. doi:10.1007/978-3-642-25832-9-14.

[51] Z. Chen, C. Wu, Y. Zhang, Z. Huang, B. Ran, M. Zhong, N. Lyu, Feature selection with redundancy-complementariness dispersion, Knowledge-Based Systems 89 (1) (2015) 203–217. doi:10.1016/j.knosys.2015.07.004.

[52] E. Hancer, B. Xue, M. Zhang, Differential evolution for filter feature selection based on information theory and feature ranking, Knowledge-Based Systems 140 (15) (2017) 103–119. doi:10.1016/j.knosys.2017.10.028.

[53] S. Maldonado, J. López, Synchronized feature selection for support vector machines with twin hyperplanes, Knowledge-Based Systems 132 (2017) 119–28. doi:10.1016/j.knosys.2017.06.025.

[54] J. G. Dy, C. E. Brodley, Feature selection for unsupervised learning, Journal of Machine Learning Research 5 (2004) 845–889.

[55] D. He, D. Cai, P. Niyogi, Laplacian score for feature selection, in: Advances in Neural Information Processing Systems 18, 2005.

[56] P. Mitra, C. A. Murthy, S. K. Pal, Unsupervised feature selection using feature similarity, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (3).

[57] D. Cai, C. Zhang, X. He, Unsupervised feature selection for multi-cluster data, in: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, New York, NY, USA, 2010, pp. 333–342. doi:10.1145/1835804.1835848.

[58] Y. Yang, H. T. Shen, Z. Ma, Z. Huang, X. Zhou, L2,1-norm regularized discriminative feature selection for unsupervised learning, in: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, AAAI Press, 2011, pp. 1589–1594. doi:10.5591/978-1-57735-516-8/IJCAI11-267.

[59] S. Tabakhi, P. Moradi, F. Akhlaghian, An unsupervised feature selection algorithm based on ant colony optimization, Engineering Applications of Artificial Intelligence 32 (1) (2014) 112–123. doi:10.1016/j.engappai.2014.03.007.

[60] S. Wang, J. Tang, H. Liu, Embedded unsupervised feature selection, in: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI Press, 2015, pp. 470–476.

[61] Y. Yi, W. Zhou, C. Bi, G. Luo, Y. Cao, Y. Shi, Inner product regularized nonnegative self representation for image classification and clustering, IEEE Access 5 (2017) 14165–14176.

[62] S. Wang, H. Wang, Unsupervised feature selection via low-rank approximation and structure learning, Knowledge-Based Systems 124 (2017) 70–79. doi:10.1016/j.knosys.2017.03.002.

[63] J. Wang, R. Zhao, Y. Wang, C. Zheng, J. Kong, Y. Yi, Locality constrained graph optimization for dimensionality reduction, Neurocomputing 245 (2017) 55–67.

[64] Y. Yi, S. Qiao, W. Zhou, C. Zheng, Q. Liu, J. Wang, Adaptive multiple graph regularized semi-supervised extreme learning machine, Soft Computing 22 (11) (2018) 3545–3562. doi:10.1007/s00500-018-3109-x.

[65] Y. Yi, W. Zhou, Q. Liu, G. Luo, J. Wang, Y. Fang, C. Zheng, Ordinal preserving matrix factorization for unsupervised feature selection, Signal Processing: Image Communication 67 (2018) 118–131. doi:10.1016/j.image.2018.06.005.

[66] M. Qi, T. Wang, F. Liu, B. Zhang, J. Wang, Y. Yi, Unsupervised feature selection by regularized matrix factorization, Neurocomputing 273 (2018) 593–610. doi:10.1016/j.neucom.2017.08.047.

[67] Y. Yi, J. Wang, W. Zhou, C. Zheng, J. Kong, S. Qiao, Non-negative matrix factorization with locality constrained adaptive graph, IEEE Transactions on Circuits and Systems for Video Technology 29. doi:10.1016/j.image.2018.06.005.

[68] W. Siedlecki, J. Sklansky, A note on genetic algorithms for large-scale feature selection, Pattern recognition letters 10 (1989) 335–347. doi:10.1016/0167-8655(89)90037-8.

[69] R. Lleti, M. C. Ortiz, L. A. Sarabia, M. S. Sanchez, Selecting variables for k-means cluster analysis by using a genetic algorithm that optimises the silhouettes, Analytica Chimica Acta 515 (2004) 87–100. doi:10.1016/j.aca.2003.12.020.

[70] H.-H. Liu, C.-S. Ong, Variable selection in clustering for marketing segmentation using genetic algorithms, Expert Systems with Applications 34 (2008) 502–510. doi:10.1016/j.eswa.2006.09.039.

[71] C.-F. Tsai, W. Eberle, C.-Y. Chu, Genetic algorithms in feature and instance selection, Knowledge-Based Systems 39 (2013) 240–47. doi:10.1016/j.knosys.2012.11.005.

[72] A. K. Das, S. Das, A. Ghosh, Ensemble feature selection using bi-objective genetic algorithm, Knowledge-Based Systems 123 (1) (2017) 116–127. doi:10.1016/j.knosys.2017.02.013.

[73] M. R. G. Raman, N. Somu, K. Kirthivasan, R. Liscano, V. S. S. Sriram, An efficient intrusion detection system based on hypergraph - genetic algorithm for parameter optimization and feature selection in support vector machine, Knowledge-Based Systems 134 (1) (2017) 1–12. doi:10.1016/j.knosys.2017.07.005.

[74] L. A. N. Lorena, J. C. Furtado, Constructive genetic algorithm for clustering problems, Evolutionary Computation 9 (2001) 309–328.

[75] A. C. M. d. Oliveira, L. A. N. Lorena, 2-Opt Population Training for Minimization of Open Stack Problem, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, Ch. 2-Opt Population Training for Minimization of Open Stack Problem, pp. 313–323. doi:10.10073-54036127830.

[76] A. C. M. Oliveira, L. A. N. Lorena, A constructive genetic algorithm for gate matrix layout problems, IEEE transactions on computer-aided design of integrated circuits and systems 21 (2002) 969–974. doi:10.1109/TCAD.2002.800454.

[77] D.-F. Shiau, S.-C. Cheng, Y.-M. Huang, Proportionate flexible flow shop scheduling via a hybrid constructive genetic algorithm, Expert System with Application 34 (2008) 1133–1143. doi:10.1016/j.eswa.2006.12.002.

[78] M. S. Nagano, R. Ruiz, L. A. N. Lorena, A constructive genetic algorithm for permutation flowshop scheduling, Computers and industrial engineering 55 (2008) 195–207. doi:10.1016/j.cie.2007.11.018.

[79] P. A.-P., A constructive genetic algorithm for the p- median location problem of typhoon emergency shelter in china coastal rural areas, in: Y. W. Wu (Ed.), Materials engineering for advanced technologies, pts 1 and 2, Vol. 480-481 of Key Engineering Materials, 2011, pp. 1215–1220. doi:10.4028/www.scientific.net/KEM.480-481.1215.

[80] S. P. Gomes, L. A. N. Lorena, G. M. Ribeiro, A constructive genetic algorithm for discrete dispersion on point feature cartographic label placement problems, Geographical analysis 48 (2016) 43–58. doi:10.1111/gean.12082.

[81] M. Owais, M. K. Osman, G. Moussa, Multi-objective transit route network design as set covering problem, IEEE Transactions on intelligent transportation systems 17 (2016) 670–679. doi:10.1109/TITS.2015.2480885.

[82] J. C. Bean, Genetic algorithms and random keys for sequencing and optimization., Journal on computing 6 (1994) 154–160. doi:10.1287/ijoc.6.2.154.

[83] J. F. GonÇalves, J. J. M. Mendes, M. G. C. Resende, A hybrid genetic algorithm for the job shop scheduling problem, European Journal of Operational Research 167 (2005) 77–95. doi:10.1016/j.ejor.2004.03.012.

[84] J. F. GonÇalves, M. G. C. Resende, J. J. M. Mendes, A biased random-key genetic algorithm with forward-backward improvement for the resource constrained project scheduling problem, Journal of Heuristics 17 (2011) 467–486. doi:10.1007/s10732-010-9142-2.

[85] J. F. GonÇalves, A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem, European Journal of Operational Research 183 (2007) 1212–1229. doi:10.1016/j.ejor.2005.11.062.

[86] J. F. GonÇalves, M. G. C. Resende, A parallel multi-population genetic algorithm for a constrained two-dimensional orthogonal packing problem., Journal of Combinatorial Optimization 22 (2011) 180–201. doi:10.1007/s10878-009-9282-1.

[87] J. A. Goncalves, J. F. Almeida, J. Raimundo, A hybrid genetic algorithm for assembly line balancing, Journal of heuristics 8 (2002) 629–642. doi:10.1023/A:1020377910258.

[88] J. F. GonÇalves, J. J. M. Mendes, An evolutionary algorithm for manufacturing cell formation, Computer 47 (2004) 247–273. doi:10.1016/j.cie.2004.07.003.

[89] J. F. GonÇalves, M. G. C. Resende, A biased random-key genetic algorithm for 2d and 3d bin packing problems, International journal of production economics 145 (2013) 500–510. doi:10.1016/j.ijpe.2013.04.019.

[90] C. E. Andrade, F. K. Miyazawa, M. G. C. Resende, Evolutionary algorithm for the k-interconnected multi-depot multi-traveling salesmen problem, in: Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference, 2013, pp. 463–470.

[91] J. F. GonÇalves, M. G. C. Resende, An extended akers graphical method with a biased random-key genetic algorithm for job-shop scheduling, International transactions in operational research 21 (2014) 215–246. doi:10.1111/itor.12044.

[92] P. Festa, A biased random-key genetic algorithm for data clustering, Mathematical Biosciences 245 (2013) 76–85. doi:10.1016/j.mbs.2013.07.011.

[93] C. E. Andrade, M. G. C. Resende, H. J. Karloff, F. K. Miyazawa, Evolutionary algorithms for overlapping correlation clustering, in: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, 2014, pp. 405–412.

[94] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural

Networks, Vol. 4, 1995, pp. 1942–1948. doi:10.1109/ICNN.1995.488968.

[95] L. J. Hubert, J. R. Levin, A General Statistical Framework for Assessing Categorical Clustering in Free Recall, Theoretical papers, Wisconsin Research and Development Center for Cognitive Learning, 1975.

[96] M. J. Brusco, J. D. Cradit, A variable-selection heuristic for k-means clustering, Psychometrika 66 (2) (2001) 249–270. doi:10.1007/BF02294838.

[97] L. I. Kuncheva, D. P. Vetrov, Evaluation of stability of k-means cluster ensembles with respect to random initialization, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (11) (2006) 1798–1808. doi:10.1109/TPAMI.2006.226.

[98] S.-S. Kim, Variable selection and outlier detection for automated k-means clustering, Communications for Statistical Applications and Methods 22 (1) (2015) 55–67. doi:10.5351/CSAM.2015.22.1.055.

[99] W. M. Rand, Objective criteria for the evaluation of clustering methods, Journal of the American Statistical Association 66 (336) (1971) 846–850. doi:10.2307/2284239.

[100] P. Jaccard, Nouvelles recherches sur la distribution florale, Bulletin de la Sociéte Vaudense des Sciences Naturelles 44 (163) (1908) 223–270. doi:10.5169/seals-268384.

[101] J. H. Holland, Adaptation in natural and artificial systems, University of Michigan press, Ann Arbor, MI, USA, 1975.

[102] G. Roffo, S. Melzi, M. Cristani, Infinite feature selection, in: 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 4202–4210. doi:10.1109/ICCV.2015.478.

[103] G. Roffo, S. Melzi, Ranking to learn: Feature ranking and selection via eigenvector centrality, in: New Frontiers in Mining Complex Patterns: 5th International Workshop, NFMCP, 2017, pp. 19–35. doi:10.1007/798-3-319-61461-8_2.

[104] G. Roffo, S. Melzi, Ranking to Learn, Springer International Publishing, 2017, Ch. Ranking to Learn, pp. 19–35. doi:10.1007/978-3-319-61461-8_2.

[105] G. Roffo, S. Melzi, U. Castellani, A. Vinciarelli, Infinite latent feature selection: A probabilistic latent graph-based ranking approach, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 1–9.

[106] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics Bulletin.

Title: Evaluation of genetic algorithms performance applied to dimensionality reduction in data clustering.

Authors: Marcelo Seido Nagano and Nadia Junqueira Martarelli

HIGHLIGHTS

- An unprecedented comparison among genetic algorithms and state-of-the-art methods of unsupervised feature selection is done.
- Biased random-key genetic algorithm and constructive genetic algorithm are modeled.
- Nine data sets are used, five real-world and four simulated.
- An exhaustive computational and statistical evaluation is carried out.
- The genetic algorithm branches outperformed other approaches.

Conflict of Interest and Authorship Conformation Form

Please check the following as appropriate:

- ✓ All authors have participated in (a) conception and design, or analysis and interpretation of the data; (b) drafting the article or revising it critically for important intellectual content; and (c) approval of the final version.

- ✓ This manuscript has not been submitted to, nor is under review at, another journal or other publishing venue.

- ✓ The authors have no affiliation with any organization with a direct or indirect financial interest in the subject matter discussed in the manuscript.