Strategic oscillation for the balanced minimum sum-of-squares clustering problem

R. Martín-Santamaría[a], J. Sánchez-Oro[a,*], S. Pérez-Peló[a], A. Duarte[a]

[a]*Dept. Computer Sciences, Universidad Rey Juan Carlos, Spain*
{*raul.martin, jesus.sanchezoro, sergio.perez.pelo, abraham.duarte*}*@urjc.es*

Abstract

In the age of connectivity, every person is constantly producing large amounts of data every minute: social networks, information about trips, work connections, etc. These data will only become useful information if we are able to analyze and extract the most relevant features from it, which depends on the field of analysis. This task is usually performed by clustering data into similar groups with the aim of finding similarities and differences among them. However, the vast amount of data available makes traditional analysis obsolete for real-life datasets. This paper addresses the problem of dividing a set of elements into a predefined number of equally-sized clusters. In order to do so, we propose a Strategic Oscillation approach combined with a Greedy Randomized Adaptive Search Procedure. The computational experiments section firstly tunes the parameters of the algorithm and studies the influence of the proposed strategies. Then, the best variant is compared with the current state-of-the-art method over the same set of instances. The obtained results show the superiority of the proposal using two different clustering metrics: MSE (Mean Square Error) and Davies–Bouldin index.

Keywords: balanced clustering, metaheuristics, strategic oscillation, GRASP, infeasibility

*Corresponding author

1. Introduction

The large amount of data available in several fields of research like economics or biology has made traditional analysis techniques impractical for real-life problems [35]. In fact, data used in most of these areas grows exponentially everyday. Therefore, the design of high-quality and high-performance algorithms has become a research field of interest for data analysts.

In order to get relevant information when using huge volume of data, two main approaches are typically followed: classification and clustering [20]. In the former, there is a complete previous knowledge of the available information (i.e., the group to which each element belongs to is known beforehand). Then, data analysis techniques can take advantage of this information in a supervised way. On the contrary, clustering techniques use this information in an unsupervised way, since it is not available *a priori*. The design of effective procedures for clustering is harder as the actual group for each element is not known [39].

Clustering problems consist in splitting data into groups (also known as clusters), which contain elements that share some specific features. In other words, it is expected that if two elements belong to the same cluster, then it is because they are related to each other by means of some specific features. Symmetrically, if two elements are in different clusters, it is because they are barely related. Therefore, clustering algorithms are designed to split a given dataset into several subsets maximizing the similarity of elements in the same subset, while minimizing the similarities between elements in different subsets. It is important to remark that the similarity metric used totally depends on the dataset and the scope of the clustering.

The minimum sum-of-squares clustering problem (MSSC) presents several practical applications in a wide variety of areas (biology, biometry, psychology, marketing, etc.), and it is also a useful technique to improve the performance of other techniques like pattern recognition, data mining, or image processing, among others [36]. This problem has been proven to be $\mathcal{NP}$-hard [24], even when considering only two clusters in the Euclidean space [2].

MSSC considers that the number of clusters k is known *a priori*. The objective of MSSC is to assign a set of n points $P = \{p_1, p_2, \ldots, p_n\}$ located in an s-dimensional Euclidean space $\mathbb{R}^s$ to a cluster $K_i \in \{K_1, K_2, \ldots, K_k\}$. MSSC then tries to find the assignment S of points to clusters with the minimum sum-of-squares distances from each point p_j of the cluster K_i to its corresponding centroid $\bar{p}_i$. Given a cluster K_i, its centroid is defined as the point whose distance to the other points of the cluster is minimum. More formally, the quality of the partition S, denoted as $f(S)$ can be evaluated as:

$$f(S) = \sum_{j=1}^{n} \sum_{i=1}^{k} x_{ij} ||p_j - \bar{p}_i||^2$$

where x_{ij} is a binary variable (with $1 \leq i \leq k$ and $1 \leq j \leq n$) that takes on the value 1 if point p_j is assigned to cluster K_i; otherwise, $x_{ij} = 0$. Naturally, this variable satisfies that $\sum_{i=1}^{k} x_{ij} = 1$ for $1 \leq j \leq n$.

The evaluation of the objective function can be improved by leveraging the results derived from the Huygens' theorem [13], as stated in [9]. Specifically, evaluating the sum-of-squared distances from all points of a given cluster to its centroid is equivalent to evaluating the sum-of-squared distances between each pair of points in the cluster divided by its cardinality. It can be formally defined as:

$$f(S) = \sum_{i=1}^{k} \sum_{j=1}^{n-1} \sum_{l=j+1}^{n} x_{ijl} \frac{||p_j - p_l||^2}{|K_i|}$$

where x_{ijl} is a binary variable (with $1 \leq i \leq k$ and $1 \leq j < l \leq n$) that takes on the value 1 if points p_j, p_l are assigned to cluster K_i; otherwise, $x_{ijl} = 0$.

It is worth mentioning that the use of the Huygens' theorem [13] allows us to design considerably efficient algorithms. Specifically, we compute the distances between each pair of elements only once (even before executing the algorithm) storing them in a matrix. Then, looking up the distance between two elements can be performed in $O(1)$ since it only requires an access to the distance matrix. Additionally, this approach makes the algorithms independent of the instance

dimensionality. See [4, 21] for further details about the issues related with the dimensionality course.

In this paper, we focus on the variant with the cardinality constraint, which is called Balanced MSSC (BMSSC) problem. Specifically, there will be $n \bmod k$ clusters of size $\lceil n/k \rceil$, and $k - (n \bmod k)$ clusters of size $\lfloor n/k \rfloor$. BMSSC has also been proven to be $\mathcal{NP}$-hard for $n/k \geq 3$. See [28] for further details.

Figure 1 shows an example of two possible clustering solutions for $k = 2$, where we consider the same set of points. On the one hand, Figure 1.a depicts a solution S_1 conformed with $K_1 = \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{F}, \mathtt{H}\}$ and $K_2 = \{\mathtt{D}, \mathtt{G}, \mathtt{E}, \mathtt{I}\}$, resulting in an objective function value of $f(S_1) = 245$. On the other hand, the solution S_2 presented in Figure 1.b contains clusters $K_1 = \{\mathtt{A}, \mathtt{C}, \mathtt{D}, \mathtt{G}, \mathtt{I}\}$ and $K_2 = \{\mathtt{B}, \mathtt{E}, \mathtt{F}, \mathtt{H}\}$, with an objective function value of $f(S_2) = 353$. Analyzing these values, we can conclude that S_1 is better than S_2 since the elements contained in each cluster are more similar among them (regarding the computation of the objective function).

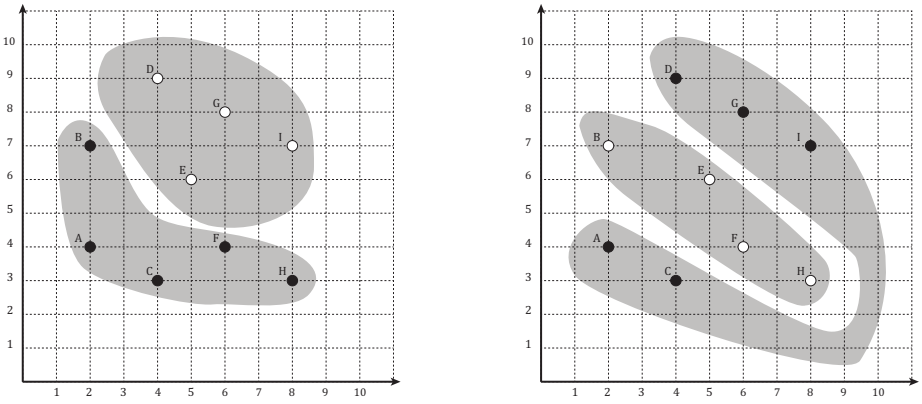

Figure 1: Two possible clustering solutions for a given set of points: (a) {A,B,C,F,H} and {D,G,E,I}, and (b) {A,C,D,G,I} and {B,E,F,H}

In this paper, we propose a Greedy Randomized Adaptive Search Procedure (GRASP) [15] combined with Strategic Oscillation (SO) [16]. Figure 2 shows

the complete scheme of the proposed algorithm, where we can identify two main blocks: GRASP and SO. The first one performs a predefined number of constructions followed by a local search method. Then, the best solution found is further improved in the SO block by alternatively considering feasible and unfeasible solutions. The SO block finally returns the best solution found during the search.

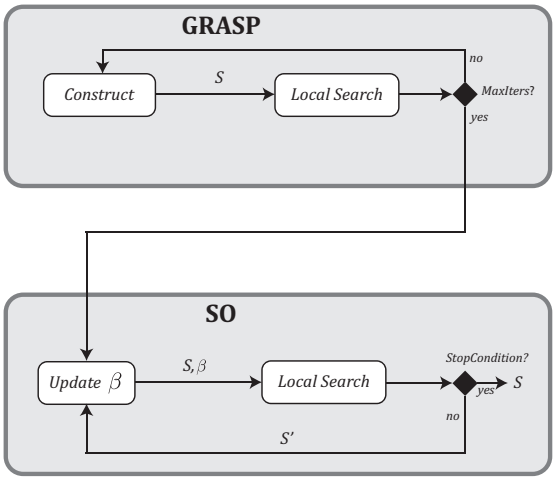

Figure 2: Complete scheme of the proposed algorithm.

This procedure presents a remarkable performance in both, quality and computing time, as we will show in the computational experience. The remaining of the paper is organized as follows: Section 3 presents the metaheuristic algorithm developed for the BMSSC, Section 4 describes a post-processing method in order to improve the quality of the obtained solutions, Section 5 thoroughly describes the experiments performed to test our proposal and, finally, Section 7 draws some conclusions over the problem and the proposed algorithms.

2. Literature review

Most of the clustering problems have been proven to be $\mathcal{NP}$-hard [17]. Therefore, the majority of the algorithms found in the related literature are

approximate procedures, where the main goal is to find a high quality partition in reasonable computing times. However, these algorithms are not able to guarantee the optimality of the provided solution. Clustering algorithms can be classified in: partitional clustering [29], in which data must be divided into disjoint sets and each element is assigned to one set; hierarchical clustering [5], where clusters are created following a top-down or bottom-up approach; density-based clustering [27], where elements are grouped following a density function; or grid-based clustering [19], in which data is divided into grids with different granularity level. We refer the reader to [34] for a detailed taxonomy of basic and advanced clustering techniques.

In this paper, we focus on partitional clustering whose objective is to generate non-overlapping subsets of elements where each element is assigned to exactly one cluster. The concept of similarity can be defined in a large variety of criteria, but usually they do not coincide. However, there is a common criterion when considering the clustering of elements that can be located in an Euclidean space. Specifically, it is widely accepted that minimizing the sum of squares between elements in the same cluster is a good criterion for clustering analysis. Furthermore, it is equivalent to maximizing the sum of squares between elements in different clusters, resulting in a criterion for increasing both similarity in the same cluster and separation among different clusters [36].

The k-means procedure has been widely used mainly due to its simplicity and its computational efficiency [31, 37]. However, it totally relies on the initial random centroid selection. Therefore, if that selection results in a bad initial set of centroids, the method will not be able to obtain a good partition. A new method named k-means++ [3] was designed to mitigate this behavior and perform a better initial centroid selection. Specifically, it consists in selecting the most diverse centroid; that is, the one with the smallest similarity measure among them. This idea reduces the harmful effect that a random centroid selection can produce in the performance of the traditional k-means algorithm. Recently, there have been another approaches proposed to even reduce this effect. In particular, the power k-means algorithm is proposed to avoid poor

local minima by annealing through a family of smoother surfaces [38]. This algorithm is further improved in [6] when dealing with high dimensions. Finally, in [8] is described a different approach, denoted as convex clustering, where a penalty function is introduced to guarantee the convexity of the derived problem.

The classical Minimum Sum-of-Squares Clustering (MSSC) problem does not have any constraint about the number of elements that can be assigned to each cluster. However, several problems require generating clusters of similar sizes. Desrosiers et al. [11] proposed a Variable Neighborhood Search algorithm for dividing students from school and universities in teams, considering that each team must provide a good representation of the population. They propose two different functions for evaluating the balance among teams and test the efficiency of the algorithm over actual data from an MBA program. This problem also has applications in Very Large Scale Integration (VLSI) design. Specifically, Hagen et al. [18] indicates that the second smallest eigenvalue of a matrix derived from the corresponding netlist provides a good approximation to the optimal ratio cut partition cost. As is stated in this paper, balance clustering problems are equivalent to the second eigenvector computation. Consequently, an effective and efficient method for the former is useful for the later. Finally, Su et al. [33] present an algorithm for balancing tenant placement in cloud computing, with the aim of improving the performance and maximizing the resource utilization of complex multi-tenant architectures.

First attempts to solve the Balanced Minimum Sum-of-Squares Clustering (BMSSC) problem consider the well-known k-means procedure [32]. This algorithm is a classical clustering method that consists of two main steps: it firstly selects the elements that will become the centroids of the clusters and then assigns each one of the remaining elements to their nearest centroid. We refer the reader to [22] for an efficient implementation of the method.

The best heuristic algorithm identified in the related literature is presented in [9]. The procedure is based on the Variable Neighborhood Search methodology, which relies on the idea of combining stochastic and deterministic changes of neighborhood to escape from local optimality. Therefore, we include this

algorithm in the computational testing. In order to complement the experimentation, we include k-means [3] with the Hungarian algorithm [23] to satisfy the balance constraint among clusters. Finally, we include two of the most performing procedures for the MSSC problem. Specifically, the hierarchical Clustering with Optimal Transport (HCOT) method [5] and the Continuous GRASP [29]. In both methods, a post-processing method is executed to guarantee the balance constraint (i.e., oversized clusters are repaired by reassigning their elements to the best available cluster).

3. Greedy Randomized Adaptive Search Procedure

The term Greedy Randomized Adaptive Search Procedure (GRASP) refers to a metaheuristic algorithm originally introduced in the late 1980s [14] but it was not formally defined until 1994 [15]. GRASP is an iterative algorithm where each iteration can be divided into two stages: generating an initial solution and then locally improving it. The first phase starts from an empty solution building the Candidate List (CL) of elements to be added to the solution under construction. The first element is usually selected at random from the CL. The remaining elements to be included in the solution are selected following a greedy criterion. Specifically, a Restricted Candidate List (RCL) is conformed with those elements that surpass the greedy criterion established. Then, the next element to be added is selected at random from the RCL. This random selection allows GRASP methodology to explore diverse regions of the search space, thus increasing the possibility of finding better solutions.

The random part of the initial solution construction is able to generate diverse solutions, but it is not designed to find a local optimum with respect to the constructed solution. Therefore, a local improvement method is required in order to find a local optimum with respect to the constructed solution. The versatility of the GRASP methodology allows us to use different algorithms in this stage, from traditional local search methods to more complex implementations such as hybridizations with other metaheuristics that has lead to successful

research: Tabu Search [26], Path Relinking [12], Ejection Chains [30], among others.

Finally, this two stages are iteratively repeated until a stopping criterion is reached, which is usually a maximum number of generated solutions or a maximum allowed running time. The method returns the best solution found during the search. The remaining of this section is devoted to presenting the specific design of the GRASP algorithm for the BMSSC.

3.1. Initialization

Classical GRASP algorithms usually select the first element to be added to the solution under construction at random. However, as stated in Section 1, the random selection of the initial elements in a clustering problem can determine the quality of the obtained results. Therefore, we propose a new initialization criterion which tries to guide the initial solution to promising regions of the search space by inserting a single vertex in each cluster. The method starts by selecting the first element at random and inserting it in the first cluster. Then, the distance from the remaining elements to the one already selected is evaluated, selecting the element that presents the largest one (i.e., the one that is furthest from the element already clustered).

Once two elements have been assigned to two different clusters, the next element to be inserted should be far away from both elements already assigned. For this purpose, we define the distance from an element p to a given cluster K_i as follows:

$$d(p, K_i) = \sum_{p_j \in K_i} ||p - p_j||^2$$

where K_i is the set of points $p_j \in P$ that have been assigned to cluster i.

Then, the distance from an element p to a given solution $S = \{K_1, K_2, \ldots, K_k\}$ can be defined as:

$$d(p, S) = \min_{K_i \in S} d(p, K_i)$$

Then, the next element to be added, $p^\star$, will be the one with the largest distance to the already clustered elements. More formally:

$$p^\star = \arg\max_{p \in P \setminus S} d(p, S)$$

This criterion allows us to insert, in each cluster, the furthest element with respect to the already clustered elements, thus reducing the similarity of the elements in different clusters and, therefore, increasing the similarity between elements in the same cluster.

The initialization stage ends when each cluster has exactly one element in it, using this partial solution as input for the constructive method.

3.2. Constructive method

The constructive algorithm proposed in this work, named *JoinClosest*, follows a traditional GRASP approach where each element is added to the cluster that minimizes the value of the objective function. *JoinClosest*, as a GRASP constructive method, requires from a greedy function that evaluates the relevance of adding an element in a given step of the construction.

The greedy function for each element is evaluated as the increase in the objective function value if the element is inserted in the closest cluster. It is worth mentioning that only the clusters which are not completed yet are considered in this step, in order to maintain the feasibility of the solution. More formally,

$$g(v, S) = \min_{1 \leq i \leq k} \sum_{p \in K_i} d(p, v)$$

Algorithm 1 depicts the pseudocode of the *JoinClosest* constructive procedure. The algorithm receives three input parameters: P, the set of points that needs to be clustered; S, the set of clusters created in the initialization step; and α, a parameter that controls the greediness/randomness of the method (discussed later).

Algorithm 1 *JoinClosest*$(P, S = \{K_1, K_2, \dots K_k\}, \alpha)$

1: $CL \leftarrow P \setminus S$
2: **while** $CL \neq \emptyset$ **do**
3: $\quad g_{\min} \leftarrow \min_{c \in CL} g(c, S)$
4: $\quad g_{\max} \leftarrow \max_{c \in CL} g(c, S)$
5: $\quad \mu \leftarrow g_{\min} + \alpha \cdot (g_{\max} - g_{\min})$
6: $\quad RCL \leftarrow \{c \in CL \,:\, g(c, S) \leq \mu\}$
7: $\quad c^\star \leftarrow Random(RCL)$
8: $\quad CL \leftarrow CL \setminus \{c^\star\}$
9: $\quad K^\star \leftarrow \arg\min_{K_i \in S} d(c^\star, K_i)$
10: $\quad K^\star \leftarrow K^\star \cup \{c^\star\}$
11: **end while**
12: **return** S

The method starts by creating the Candidate List (CL) with the set of elements in P that has not been assigned to any cluster in the initialization phase (step 1). Then, it iterates until all the elements have been clustered (steps 2-11), adding a new element to a cluster in each iteration.

Specifically, *JoinClosest* evaluates the minimum and maximum value for the greedy function previously defined (steps 3-4) and then evaluates a threshold μ (step 5) that depends on the value of the parameter $\alpha \in [0, 1]$ whose function is to limit the elements that are allowed to enter the Restricted Candidate List (RCL). On the one hand, $\alpha = 0$ indicates that only the elements with the best greedy function value are selected, which results in a totally greedy algorithm. On the other hand, $\alpha = 1$ considers all the elements in the CL to be added to the RCL, which results in a totally random algorithm. Therefore, parameter α is able to control the greediness/randomness of the constructive procedure.

Once the RCL is created with the elements whose objective function value is smaller than the threshold (step 6), an element $c^\star$ is selected at random from the RCL (step 7). Then, the algorithm selects the closest cluster $K^\star$ (step 9) and inserts the element in the cluster (step 10). The method ends when all the

elements have been assigned to a cluster, returning the solution created.

3.3. Local optimization

The solution created in the construction phase tries to find a balance between diversification and intensification in order to explore a wider portion of the search space. However, this behavior complicates finding a local optimum in the construction stage. The second phase of a GRASP procedure consists of an improvement strategy that is able to find a local optimum of the initial solution with respect to a given neighborhood.

In this work, we consider a neighborhood based on interchanging two elements of different clusters. More formally, given a solution

$$S = \{K_1, \ldots, K_a, \ldots, K_i, \ldots, K_k\}$$

the interchange of elements $p_j \in K_a$ and $p_l \in K_i$, produces a new solution $S' = \{K_1, \ldots, K_a \setminus \{p_j\} \cup \{p_l\}, \ldots, K_i \setminus \{p_l\} \cup \{p_j\}, \ldots, K_k\}$. For the sake of clarity, we represent this move as $S' \leftarrow move(S, p_j, K_i)$.

We propose a local search method for the BMSSC based on this neighborhood. It evaluates the interchange of every pair of elements in the solution, executing in each step the first interchange that results in a better solution (first improvement, FI). We additionally propose a different approach in which the interchange performed is not the first element that leads to a better solution but the one that leads to the best solution in the neighborhood (best improvement, BI). The performance of both local search methods will be later discussed in Section 5.

4. Strategic oscillation

Reaching a better solution during the search might prove difficult in some cases, since the constraint on the size of each cluster in the BMSSC problem limits the number of available movements that can be performed. Specifically, in order to maintain the same size in each cluster it is only possible to perform symmetrical movements (mostly based on interchanges).

With the aim of increasing the portion of the search space explored, we propose to consider unfeasible solutions during the search that can eventually lead the algorithm to better feasible solutions.

Strategic oscillation (SO) is a methodology originally proposed for being used in combination with Tabu Search [16]. It is based on allowing the algorithm to surpass the boundaries of its search space, usually by consider the exploration of unfeasible solutions. When the search gets stuck in a deep basin of attraction, SO modifies the rules of the search, allowing the algorithm to continue its exploration considering the set of unfeasible solutions. Every time a promising unfeasible solution is reached, the algorithm must repair it in order to transform it into a feasible solution that would eventually be a high-quality one.

The first step in SO is the definition of the boundary to be surpassed. In the context of BMSSC, we consider the increment of the size of each cluster. This modification is performed by increasing each cluster size by a percentage defined by a parameter $\beta \in [0, 1]$ that controls how far the explored solutions are from being feasible. Specifically, given a cluster K_i, if the original cluster size is $|K_i|$, considering the relaxation of the feasibility, the new cluster size is now limited by $|K_i| \cdot (1 + \beta)$. This relaxation allows the algorithm to include more points in each cluster, which may lead the procedure to find better solutions that can be later repaired. A search algorithm that considers small values of β will explore solutions that are almost feasible, while considering larger values will explore rather unfeasible solutions.

SO allows us to define a new neighborhood to be explored, which consists in moving a given element from one cluster to another. Notice that this movement cannot be considered in the feasible solution space since any move violates the size constraint. Given a point p_j that belongs to cluster K_a which is moved to cluster K_i in a certain solution $S = \{K_1, \ldots, K_a, \ldots, K_i, \ldots, K_k\}$, the movement generates a new solution $S' = \{K_1, \ldots, K_a \setminus \{p_j\}, \ldots, K_i \cup \{p_j\}, \ldots, K_k\}$.

The SO methodology can be divided into two phases: the first one is a local search devoted to exploring the unfeasible region while the second one tries to repair every promising solution.

The local search phase considers the *move* neighborhood previously defined as follows. The method evaluates the move of each element p_j (with $1 \leq j \leq n$) of the solution $S = \{K_1, K_2, \ldots, K_k\}$ to every cluster K_i (with $1 \leq i \leq k$). Notice that, in the context of SO, the size of each cluster K_i is now limited by $|K_i| \cdot (1 + \beta)$. The method then selects the cluster $K^\star$ that minimizes the objective function value f. More formally,

$$K^\star = \underset{1 \leq i \leq k; 1 \leq j \leq n}{\arg\min} f\left(move(S, p_j, K_i)\right)$$

It is worth mentioning that the local search moves each element to the cluster that minimizes the value of the objective function, resulting in a best improvement method.

The repair phase is applied to every unfeasible solution whose objective function value outperforms the best solution found so far. This phase is intended to reduce the number of elements of each oversized cluster as follows. For each element p_j belonging to an oversized cluster, the method finds the cluster K_i that minimizes the value of the objective function of those whose size satisfies the original size constraint. Then, it applies the operation $move(S, p_j, K_i)$ in order to insert p_j in cluster K_i. The method ends when every cluster satisfies the original size constraint, returning the best solution found during the search.

5. Computational experiments

This section is intended to evaluate the quality of the proposed algorithms and compare it with the best previous approach. All algorithms have been implemented using Java 8 and the experiments were conducted in an Intel Core i5-4210U 1.7GHz and 8GB RAM. The source code and the full experimental results are available at the following URL: `https://grafo.etsii.urjc.es/BMSSC`.

In order to have a fair comparison, we have considered the set of 16 instances used in the best previous work found in the literature. We have additionally incorporated 9 instances to have a larger benchmark. All these instances have

been taken from the machine learning repository of the University of California[1]. Table 1 shows, for each single instance, the number of points (n), dimensions (s), and clusters (k). Notice that we first show the original instances and then the new instances, separated by an horizontal line. It is worth mentioning that all instances come from real data. In particular, benchmarks consider information derived from different real world scenarios: breast cancer, phone sensors, water treatment, among others. We refer the reader to [9] to find a deep analysis and description of these instances.

We divide the experiments performed into two different subsets: preliminary experimentation and final experimentation. The former refers to those experiments designed to find the best parameters for the proposed algorithms and to evaluate the relevance of the proposed strategies (constructive method, local search procedure, and Strategic Oscillation algorithm), while the aim of the latter is to perform a comparison between the best algorithm designed and the best previous method found in the state of the art. Notice that the proposed algorithm requires just one run to obtain the presented results.

All the experiments report the following metrics: Dev. (%), the average deviation with respect to the best solution found in the experiment; #Best, the number of times that an algorithm reaches the best solution; and Time (s), the average computing time in seconds.

5.1. Preliminary experimentation

This section is intended to find the best values for the input parameters of the proposed algorithms. Specifically, the algorithms proposed require to find the best value for α and β parameters (constructive procedure and Strategic Oscillation, respectively), as well as selecting the best local search method. We have selected a subset of 4 representative instances (Vehicle, Yeast, Multiple Features, and Image Segmentation) for tuning the parameters of the algorithm in order to avoid overfitting.

[1] `http://www.ics.uci.edu/~mlearn/MLRepository.html`

Instance Name	n	s	k
Body	507	5	2
Breast Cancer	569	30	2
Glass	214	9	7
Image Segmentation	2310	19	7
Ionosphere	351	34	2
Iris	150	4	3
Libra	360	90	15
Multiple Features Reduced	2000	240	7
Synthetic Control	600	60	6
Thyroid	215	5	3
User Knowledge	403	5	4
Vehicle	846	18	6
Vowel	871	3	3
Water	527	38	13
Wine	178	13	3
Yeast	1484	8	10
Cardiotopography	2126	24	10
MobileKSD	2855	71	56
Ozone	2536	73	21
Seismic Bumps	2584	15	19
Internet Ads	3279	1555	2
PhonesAcc	2000	3	4
PhonesGyro	2000	3	4
WatchAcc	2000	3	4
WatchGyro	2000	3	4

Table 1: Individual description of each instance considered in this work. For each instance, the following parameters are provided: n, total number of points; s, number of dimensions; k, number of clusters to create.

The first experiment is devoted to evaluating the performance of constructive method *JoinClosest* when varying the α parameter value. In particular, we have considered $\alpha = \{0.25, 0.50, 0.75, RND\}$, where *RND* indicates that the value is selected at random in the range 0–1 in each construction.

Algorithm	**Dev.(%)**	**#Best**	**Time (s)**
JoinClosest(0.25)	1.10	1	8.99
JoinClosest(0.50)	20.51	2	9.89
JoinClosest(0.75)	22.77	0	9.85
JoinClosest(*RND*)	13.94	1	9.26

Table 2: Performance of the constructive method *JoinClosest* when varying the α parameter value.

Table 2 shows the effect of the α parameter in the performance of *JoinClosest* when constructing 100 solutions. In particular, the best results are achieved with *JoinClosest*(0.25), which is the closest variant to a totally greedy algorithm. It is worth mentioning that a pure greedy configuration of this constructive method produces worse results than the constructive procedure configured with $\alpha = 0.25$. We can then conclude from this experiment that the larger the randomness, the lower the quality of the constructive procedure. Notice that these results are in line with those presented in [3], which indicates that a random initialization in the well-known k-means algorithm usually produces worse quality solutions than those obtained with the greedy initialization of the k-means++ version.

The main aim of the second experiment is to analyze the ability of each proposed improvement method for finding a local optimum starting the search from each constructed solution. Specifically, we combine the best constructive method (i.e., $\alpha = 0.25$) with the two local search methods proposed in Section 3.3: First Improvement (FI) and Best Improvement (BI).

Table 3 compares the results obtained when combining the best variant of the constructive procedure coupled with both local search algorithms, resulting

Algorithm	Dev.(%)	#Best	Time (s)
JoinClosest(0.25)	31.41	0	8.99
JoinClosest(0.25) + BI	0.24	2	15.24
JoinClosest(0.25) + FI	0.07	2	10.57

Table 3: Comparison of the two local search procedures, Best Improvement (BI) and First Improvement (FI),

and the best constructive method.

into two GRASP variants: *JoinClosest*(0.25) + FI and *JoinClosest*(0.25) + BI. We also include the results obtained with *JoinClosest* isolated, in order to analyze the contribution of the improvement strategies. Notice that the results reported in Table 3 are averaged over the 4 instances used in the preliminary experiments. As can be seen in this table, both local search approaches obtain similar results in terms of quality, which can be seen in the average deviation with respect to best found value (0.24% and 0.07%, respectively) and number of best solutions found (both methods matches 2 best solutions). It is also important to remark that avoiding exploring the neighborhood exhaustively with the First Improvement approach lead us to marginally increase the computing time with respect to the constructive method (10.57 versus 8.99 seconds on average). However, the Best Improvement approach requires almost twice the computing time (15.24 versus 8.99), without significantly improving the obtained results. Therefore, we select the First Improvement approach as the local search procedure for the final algorithm.

The next preliminary experiment is intended to evaluate the contribution of using Strategic Oscillation to further improve the best solution obtained with the GRASP procedure. As stated in Section 4, SO method requires only one parameter that controls how far from feasibility are the solutions explored during the search. In this experiment we consider the values $\beta = \{0.1, 0.25, 0.50, 0.75\}$. These values indicates the increase of each cluster size in a 10%, 25%, 50%, and 75% of the original cluster size. We do not consider larger values of β since

values that exceed 100% will evaluate clusters with more than twice the original size, which is rather distant from feasibility. If so, it would be equivalent to start the search from a totally new solution.

Algorithm	Dev.(%)	#Best	Time (s)
JoinClosest(0.25) + FI	0.81	0	10.57
SO(0.10)	0.24	3	10.94
SO(0.25)	0.66	1	10.69
SO(0.50)	0.22	2	10.82
SO(0.75)	0.09	3	11.21

Table 4: Results obtained by Strategic Oscillation (SO) with different increments in the cluster size.

Table 4 presents the results obtained with the aforementioned different values of β. Additionally, we have included the best GRASP variant for measuring the contribution of the Strategic Oscillation to the quality of the algorithm. As can be easily seen, every variant of the Strategic Oscillation algorithm outperforms the GRASP procedure in all metrics, barely increasing the computing time. Among SO variants, the one that increases the size of the cluster in a 75% obtains the best results. This can be mainly due to its ability to explore further regions of the search space, most of them intractable for the remaining variants. Therefore, we select $\beta = 0.75$ as the best variant of the SO algorithm.

5.2. Comparison with the state-of-the-art procedures

The next experiment is devoted to evaluating the contribution of our best proposal by comparing it with the best previous method identified in the state of the art, which is VNS-LIMA [9]. This method is a Variable Neighborhood Search algorithm that follows the "Less Is More Approach". We additionally include a third algorithm in the comparison to verify the superiority of the proposal. In particular, we have considered an adaptation of the traditional k-means for balanced clustering [25]. This work follows the well-known k-means clustering

algorithm, which is one of the most successful algorithms for clustering. Instead of selecting the closest centroid, it considers a set of clusters in which a point can be assigned, in order to satisfy the size constraint. This assignment is performed by following the Hungarian algorithm [23].

Additionally, we have executed two state-of-the-art algorithms for the classic clustering problem. Specifically, the Hierarchical Clustering with Optimal Transport (HCOT) [5] and the Continuous Greedy Randomized Adaptive Search Procedure (CGRASP) [29]. Notice that these methods do not consider the balance constraint. Therefore, once an algorithm obtains a solution, a post processing method is applied in order to make it feasible. In order to do so, each point belonging to an overloaded cluster is moved to the best cluster not yet completed.

With the aim of facilitating the comparison among algorithms, we report in Table 5 the same information than the one reported in [9]. Specifically, we show the Mean Squared Error of (MSE) of every single instance achieved with the proposed Strategic Oscillation algorithm (SO), when comparing with those obtained with HCOT, CGRASP, k-means with Hungarian algorithm (KMH), and VNS-LIMA. All the algorithms have been executed in the same computer and the same time per instance to have a fair comparison (last column of Table 5). We additionally consider in these experiments the whole set of 25 instances. Instances where a procedure is not able to produce a feasible solution are marked with an asterisk symbol $'*'$.

As we can observe, instance dimensions are rather different so it is hard to compare directly the MSE value. Therefore, we consider the average deviation with respect to the best known value since this metric is dimensionless. In particular, SO presents an average deviation of 0.38% with respect to the best known value, while the deviation of VNS-LIMA rises up to 4.84%. Finally, the HCOT, CGRASP, and KMH algorithms have a deviation of 23.70%, 54.35%, and 23.20%, respectively. Summarizing, SO is able to reach the best known solution in most of the instances (19 out of 25). Symmetrically, in the 6 instances in which SO does not obtain the best value, the corresponding result remains

Instance Name	HCOT	CGRASP	KMH	VNS	OS	Time (s)
Body	1.14E+05	1.14E+05	2.34E+05	**1.14E+05**	1.14E+05	1.08
Breast cancer	*	**1.38E+08**	1.38E+08	**1.38E+08**	**1.38E+08**	0.90
Glass	9.70E+02	9.36E+02	7.29E+02	**5.08E+02**	5.25E+02	0.27
Image Segmentation	*	2.24E+07	2.75E+07	2.14E+07	**2.14E+07**	19.24
Ionosphere	2.52E+03	2.44E+03	2.52E+03	**2.43E+03**	**2.43E+03**	0.32
Iris	**8.14E+01**	**8.14E+01**	8.90E+01	**8.14E+01**	**8.14E+01**	0.06
Libra	*	6.66E+07	6.70E+07	6.42E+07	**6.41E+07**	0.57
Multiple Features	2.04E+06	2.00E+06	2.10E+06	1.99E+06	**1.96E+06**	16.29
Synthetic Control	1.64E+06	1.09E+06	1.28E+06	1.14E+06	**1.01E+06**	0.97
Thyroid	3.82E+04	3.70E+04	3.87E+04	**3.44E+04**	**3.44E+04**	0.12
User Knowledge	8.31E+01	7.17E+01	8.06E+01	7.09E+01	**7.03E+01**	0.49
Vehicle	*	6.32E+06	4.76E+06	2.93E+06	**2.90E+06**	2.13
Vowel	*	6.47E+07	1.58E+08	7.53E+07	**6.45E+07**	2.18
Water	*	2.73E+10	**7.88E+09**	7.94E+09	7.93E+09	0.96
Wine	*	5.55E+06	3.77E+06	3.83E+06	**3.77E+06**	0.08
Yeast	6.74E+01	5.82E+01	6.01E+01	5.41E+01	**5.35E+01**	8.84
Cardiotopography	1.19E+07	2.72E+07	1.10E+07	**8.50E+06**	8.60E+06	23.24
MobileKSD	*	3.85E+10	3.28E+10	3.12E+10	**3.11E+10**	141.65
Ozone	*	3.61E+09	2.56E+09	**2.46E+09**	2.56E+09	61.93
Seismic Bumps	*	1.07E+14	2.96E+13	3.05E+13	**2.94E+13**	56.93
Internet Ads	3.76E+04	*	*	**3.74E+04**	3.75E+04	24.41
Phones Acc	4.16E+02	*	3.35E+02	3.11E+02	**2.63E+02**	20.14
Phones Gyro	3.65E+01	*	3.42E+01	3.47E+01	**3.37E+01**	13.72
Watch Acc	2.08E+03	*	1.80E+03	2.65E+03	**1.68E+03**	30.86
Watch Gyro	1.64E+01	*	1.58E+01	1.54E+01	**1.54E+01**	22.86

Table 5: Final comparison among SO, VNS-LIMA, KMH, CGRASP and HCOT considering the MSE metric. Best values found with each method are highlighted with bold font.

very close to the best known.

With the aim of studying the adaptability of the algorithms to different metrics, we evaluate the five procedures when considering an alternative metric not used during the optimization phase. In particular, the well-known Davies-Bouldin [10] index (DB) is analyzed. This index was proposed for evaluating the quality of clustering algorithms by reducing the inter-cluster similarity while increasing the intra-cluster similarity. The smaller the index value, the better the clustering. Table 6 shows the individual results of each algorithm over each instance evaluated with the Davies-Bouldin index.

These results are in line with those reported in Table 5, obtaining our method the best DB index in 16 out of 25 instances, followed by VNS-LIMA (11 out of 25). As expected, algorithms not explicitly designed for the BMSSC problem present a moderate performance. We can then conclude that the proposed Strategic Oscillation algorithm emerges as the best variant even considering this new metric.

We complement these experiments by conducting a Friedman test to determine whether there exists statistically significant differences among the compared methods or not. The resulting p-value smaller than 0.0001, in both MSE and DB metrics, indicates that the proposed algorithm is statistically better than the competitors. Table 7 reports the associated rank values for both MSE and DB for the five compared algorithms. These results confirm the superiority of the proposal when considering short computing times, which is specially relevant when considering applications where clustering is just a small part of the whole process that must be performed several times.

6. Experimental Analysis

This section is devoted to deeply analyzing the parameters selected for the proposed algorithm. Specifically, we first test the robustness and reliability of the proposed algorithm by executing 30 times the SO algorithm over the 4 instances of the preliminary experimentation (Multiple Features, Vehicles,

Instance Name	HCOT	CGRASP	KMH	VNS	OS	Time (s)
Body	9.25E+01	9.24E+01	8.35E+02	**9.23E+01**	9.24E+01	1.08
Breast cancer	*	**1.64E+02**	1.65E+02	**1.64E+02**	**1.64E+02**	0.90
Glass	3.02E+02	5.80E+02	9.12E+01	**3.67E+01**	6.14E+01	0.27
Image Segmentation	*	1.11E+03	9.43E+02	**8.33E+02**	1.15E+03	19.24
Ionosphere	3.07E+02	2.65E+02	3.08E+02	**2.64E+02**	**2.64E+02**	0.32
Iris	**1.64E+01**	**1.64E+01**	2.02E+01	**1.64E+01**	**1.64E+01**	0.06
Libra	*	3.24E+02	4.38E+02	**2.88E+02**	3.16E+02	0.57
Multiple Features	1.18E+03	9.83E+02	1.25E+03	9.50E+02	**8.40E+02**	16.29
Synthetic Control	1.53E+03	2.90E+02	3.67E+02	4.19E+02	**2.36E+02**	0.97
Thyroid	1.16E+02	9.86E+01	1.75E+02	**9.00E+01**	**9.00E+01**	0.12
User Knowledge	2.55E+02	1.68E+02	2.02E+02	1.56E+02	**1.42E+02**	0.49
Vehicle	*	2.64E+02	9.01E+02	1.40E+02	**1.05E+02**	2.13
Vowel	*	1.47E+02	1.08E+03	1.71E+02	**1.47E+02**	2.18
Water	*	1.58E+03	4.72E+01	2.71E+01	**2.53E+01**	0.96
Wine	*	2.72E+01	1.49E+01	1.51E+01	**1.49E+01**	0.08
Yeast	7.00E+02	5.89E+02	3.65E+02	3.65E+02	**3.63E+02**	8.84
Cardiotopography	1.41E+03	3.61E+03	7.75E+02	5.02E+02	**4.85E+02**	23.24
MobileKSD	*	1.95E+03	3.92E+02	**3.89E+02**	3.94E+02	141.65
Ozone	*	5.16E+03	2.91E+02	1.50E+02	**1.46E+02**	61.93
Seismic Bumps	*	3.58E+04	**2.11E+02**	4.23E+03	2.14E+02	56.93
Internet Ads	3.12E+04	*	*	**2.57E+04**	3.01E+04	24.41
Phones Acc	2.10E+03	*	6.98E+02	3.84E+03	**5.22E+02**	20.14
Phones Gyro	1.23E+04	*	1.81E+03	1.87E+03	**1.66E+03**	13.72
Watch Acc	1.46E+03	*	**1.19E+03**	2.30E+05	1.36E+03	30.86
Watch Gyro	1.54E+04	*	6.60E+03	**6.09E+03**	6.14E+03	22.86

Table 6: Final comparison among SO, VNS-LIMA, KMH, CGRASP and HCOT considering the Davies-Bouldin metric. Best values found with each method are highlighted with bold font.

Metric	HCOT	CGRASP	KMH	VNS	OS	$p < 0.01$
MSE	4.38	3.62	3.50	2.10	**1.40**	YES
DB	4.34	3.62	3.30	2.10	**1.64**	YES

Table 7: Friedman test for both MSE (Mean Squared Error) and DB (Davies–Bouldin) metrics.

Yeast, and Image Segmentation). Considering that differences in the objective function are rather large, we use the average deviation with respect to the best value found in the 30 independent executions, since it is a dimensionless metric. We depict in Figure 3 the associated box and whisker plot, reporting for each instance quartiles, median, minimum, and maximum values (excluding outliers).

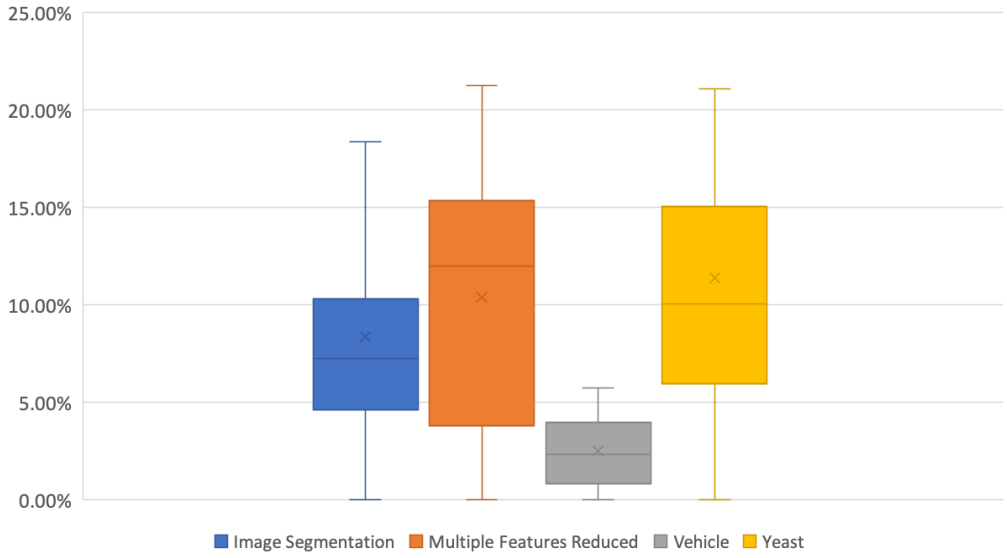

Figure 3: Box and whisker plot for 30 independent executions.

As it can be derived from the results, the proposed algorithm presents a robust behavior when considering 30 independent executions in the instances considered for the preliminary experiments. As expected, the combination of GRASP with SO tries to find a balance between diversification and intensification. Specifically, in all the instances the mean (represented with an x) and median (represented with an horizontal value) values are very close, with the median under the mean in most of the cases. This result indicates that the algorithm is able to diversify the search to explore a larger portion of the search space but the intensification phase is able to lead the algorithm to high-quality

solutions.

We additionally conduct a sensitivity analysis. In particular, for each parameter of the SO algorithm, namely α (the balance between greediness/randomness of the constructive method) and β (the percentage of cluster size increment inside Strategic Oscillation), we evaluate different possibilities while fixing the remaining parameters to the best values found in the preliminary experimentation. In particular, we test values $\alpha = \{RND, 0.25, 0.50, 0.75\}$ and $\beta = \{0.1, 0.25, 0.5, 0.75\}$. For each instance and each parameter setting, we execute 30 independent iterations of each algorithm.

α	0.25	0.50	0.75
0.50	0.686		
0.75	0.181	0.581	
RND	0.196	0.123	0.742

β	0.10	0.25	0.50
0.25	0.012		
0.50	0.002	0.000	
0.75	0.133	0.019	0.002

Table 8: Sensitivity analysis for α and β parameters.

We consider the Wilcoxon signed rank test to determine if there exists significant statistical differences among variants (in terms of the average objective function value) when we only vary a single parameter as mentioned above. The corresponding Wilcoxon test shows that the different α configurations do not significantly affect the performance of the proposal. Specifically, the associated p-values range from 0.181 to 0.742 which are considerably larger than the customary 0.05 threshold. This experiment shows that the proposed algorithm does not present a particular sensitivity with respect to this parameter.

On the other hand, observing the β parameter, there are statistical differences in performance. Therefore, this experiment justifies the election of the $\beta = 0.75$, as shown in Table 4.

To further investigate the performance of the proposed SO procedure, we conduct a convergence analysis by considering time-to-target plots (TTTPlot), which is essentially a run-time distribution [1]. The experimental hypothesis in TTTPlots is that running times fit a two parameter, or shifted, exponential

distribution. Then, for a particular instance, the execution time needed to find an objective function value at least as good as a given target value is recorded. In the context of the heuristic optimization, the algorithm is determined a pre-established number of times on the selected instance and using the given target solution. For each of run, the random number generator is initialized with a different seed and therefore the executions are assumed to be independent. To compare the empirical and the theoretical distributions, we follow a standard graphical methodology for data analysis [7], execute our algorithm 30 times, and recording for each instance/target pair the corresponding running time. Figure 4 shows the TTTPlot for those instances in the set of preliminary experiments. In these figures, each value in the abscissa axis represents a running time, while each value in the ordinate axis, reports the probability of obtaining the best-known value. This experiment confirms the expected exponential run-time distribution of our SO algorithm. If we analyze the instances Multiple Features, Vehicles, and Yeast, we can observe that the probability of SO to find a solution at least at good as the target value in less than a second is close to 100%. However, regarding Image Segmentation, which is a more complex instance, this probability is near 50% when considering 10 seconds, requiring about 15 seconds to rise the probability to 100%.

7. Conclusions

We proposed a Greedy Randomized Adaptive Search Procedure (GRASP) coupled with Strategic Oscillation (SO) algorithm for the Balanced Minimum Sum-of-Squares Clustering Problem (BMSSC), which consists in grouping a set of s-dimensional points into k clusters maximizing the similarity among them. The experiments performed showed that SO is able to modify the search space, allowing us to explore solutions that are unattainable by using traditional heuristic procedures. This behavior leads our proposal to obtain better results than the best previous method found in the state of the art. The simplicity of the method and its speed is essential when considering large amounts of data

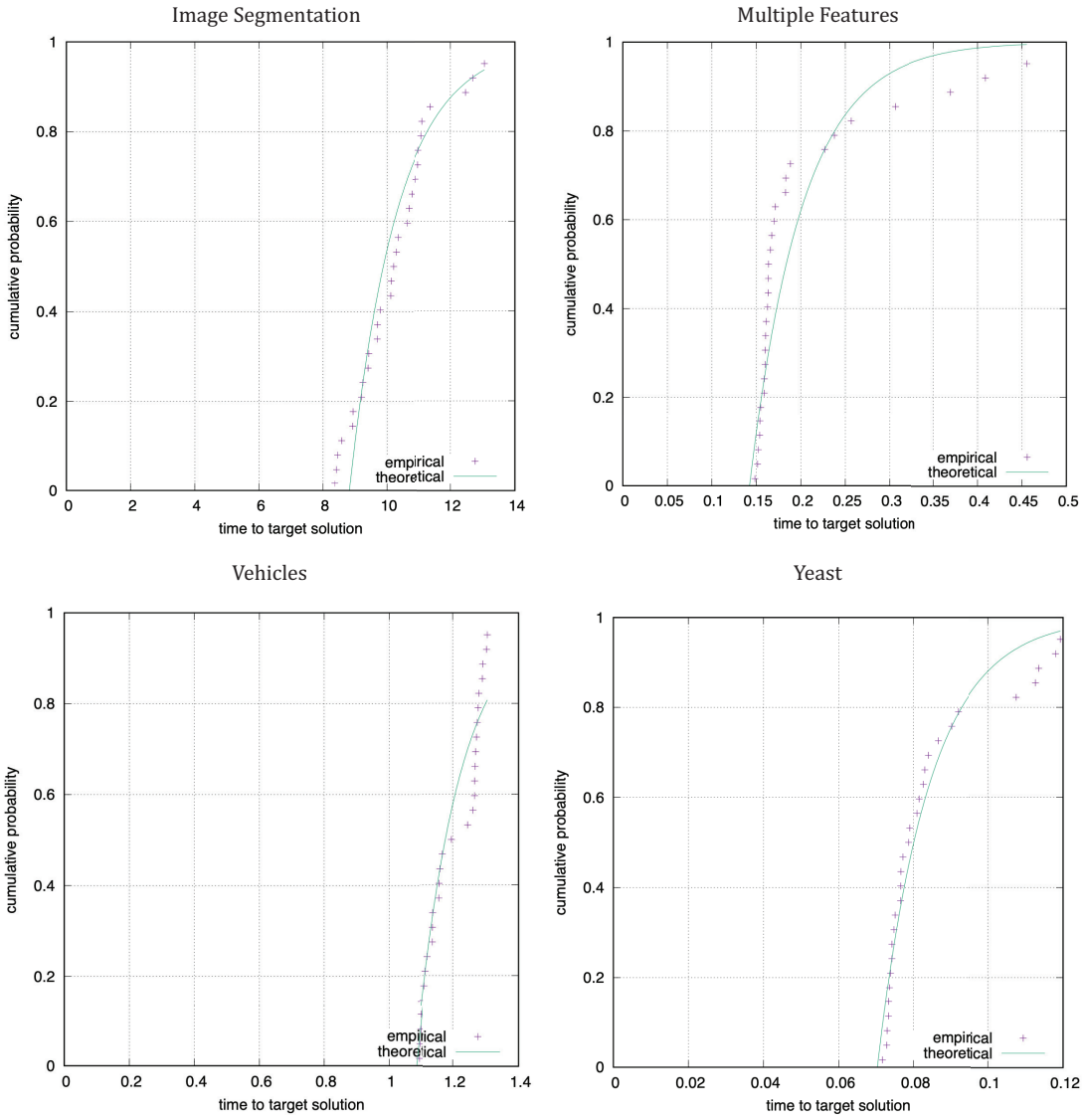

Figure 4: Time to target plots for the preliminary instances.

that are continuously generated (i.e., data derived from the stock exchange, social networks, etc.) and must be quickly analyzed.

The proposed algorithm presents the best results in the literature for the BMSSC problem, requiring small computing times. As we follow the GRASP methodology, the algorithm is easily scalable to a distributed system, in order to further reduce the computing times. The main limitation of our algorithm emerges when dealing with a variant in which the size of the clusters is not fixed and can vary during the execution. In that case, a deep redesign of the algorithm should be performed to adapt the method to this new problem. Furthermore, the algorithm is designed to work with a single neighborhood structure. In order to include more neighborhoods, one shall consider a more complex local search method, such as Variable Neighborhood Descent, which embeds several neighborhood structures in the same algorithm.

In order to summarize the main features of our procedure, we report in Table 9 its main advantages and disadvantages.

Advantages	**Disadvantages**
Best results in the literature	Not suitable if the size of the clusters is not fixed
Fast method	Does not guarantee optimality
Easily adaptable to new objective functions	Works with just one neighborhood structure
Scalable to a distributed architecture	
Independent of the dataset	

Table 9: Advantages and disadvantages of the proposed algorithm

Focusing on the first disadvantage mentioned, this paper deals with the BMSSC, in which all the clusters share the same size. The proposed algorithm is designed to obtain high quality solutions when the cluster size is constrained. This proposal can be easily adapted to an unconstrained problem by modifying

the feasibility constraint and the greedy criterion of the algorithm, in order to consider alternative moves inside the local search and constructive procedure.

As a metaheuristic algorithm, GRASP does not guarantee optimality. However, it must be born in mind that, in most real world applications, finding the optimum value is not feasible, mainly due to the complexity of the problem under consideration. Nonetheless, the proposed GRASP algorithm tries to balance solution quality and computational cost, being able to reach high quality solutions in reasonable computing time. Due to the size of the instances under consideration, using an exact solver cannot be considered.

This algorithm considers a single neighborhood structure since the results obtained are excellent, and it is not necessary to increase the computational time by including additional neighborhoods. However, the algorithm can be easily modified to consider new neighborhood structures, by extending the proposed local search procedure.

Acknowledgments

We would like to thank professors Costa, Aloise, and Mladenovic for kindly sending us the executable version of their algorithm and for the help configuring it. This work has been partially supported by the "Ministerio de Ciencia, Innovación y Universidades" under grant ref. PGC2018-095322-B-C22 and "Comunidad de Madrid" and "Fondos Estructurales" of European Union with grant ref. S2018/TCS-4566.

References

[1] Aiex, R. M., Resende, M. G., and Ribeiro, C. C. (2007). Ttt plots: a perl program to create time-to-target plots. *Optimization Letters*, 1(4):355–366.

[2] Aloise, D. J., Aloise, D., Rocha, C. T. M., Ribeiro, C., Filho, J. R., and Moura, L. S. (2006). Scheduling workover rigs for onshore oil production. *Discrete Applied Mathematics*, 154(5):695–702.

[3] Arthur, D. and Vassilvitskii, S. (2007). k-means++: the advantages of careful seeding. pages 1027–1035, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

[4] Chakraborty, S. and Das, S. (2020). Detecting meaningful clusters from high-dimensional data: A strongly consistent sparse center-based clustering approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[5] Chakraborty, S., Paul, D., and Das, S. (2020a). Hierarchical clustering with optimal transport. *Statistics & Probability Letters*, 163:108781.

[6] Chakraborty, S., Paul, D., Das, S., and Xu, J. (2020b). Entropy weighted power k-means clustering. In *International Conference on Artificial Intelligence and Statistics*, pages 691–701. PMLR.

[7] Chambers, J. M. (2017). *Graphical methods for data analysis: 0*. Chapman and Hall/CRC.

[8] Chi, E. C. and Lange, K. (2015). Splitting methods for convex clustering. *Journal of Computational and Graphical Statistics*, 24(4):994–1013.

[9] Costa, L. R., Aloise, D., and Mladenović, N. (2017). Less is more: basic variable neighborhood search heuristic for balanced minimum sum-of-squares clustering. *Inf. Sci.*, 415:247–253.

[10] Davies, D. L. and Bouldin, D. W. (1979). A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227.

[11] Desrosiers, J., Mladenović, N., and Villeneuve, D. (2005). Design of balanced mba student teams. *Journal of the Operational Research Society*, 56(1):60–66.

[12] Duarte, A., Sánchez-Oro, J., Resende, M. G. C., Glover, F., and Martí, R. (2015). Greedy randomized adaptive search procedure with exterior path relinking for differential dispersion minimization. *Inf. Sci.*, 296:46–60.

[13] Edwards, A. W. F. and Cavalli-Sforza, L. (1965). A Method for Cluster Analysis. *Biometrics*, 21(2):362–375.

[14] Feo, T. A. and Resende, M. G. C. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2):67 – 71.

[15] Feo, T. A., Resende, M. G. C., and Smith, S. H. (1994). A Greedy Randomized Adaptive Search Procedure for Maximum Independent Set. *Operations Research*, 42(5):860–878.

[16] Glover, F. and Hao, J.-K. (2011). The case for strategic oscillation. *Annals of Operations Research*, 183(1):163–173.

[17] González, T. F. (1982). On the computational complexity of clustering and related problems. In Drenick, R. F. and Kozin, F., editors, *System Modeling and Optimization*, pages 174–182, Berlin, Heidelberg. Springer Berlin Heidelberg.

[18] Hagen, L. W. and Kahng, A. B. (1992). New spectral methods for ratio cut partitioning and clustering. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 11(9):1074–1085.

[19] Hinneburg, A. and Keim, D. A. (1999). Optimal grid-clustering : Towards breaking the curse of dimensionality in high-dimensional clustering. In *Proceedings of the 25 th International Conference on Very Large Databases, 1999*, pages 506–517.

[20] Jajuga, K., Sokolowski, A., and Bock, H.-H. (2012). *Classification, clustering, and data analysis: recent advances and applications*. Springer Science & Business Media.

[21] Jin, J., Wang, W., et al. (2016). Influential features pca for high dimensional clustering. *Annals of Statistics*, 44(6):2323–2359.

[22] Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, A. Y. (2002). An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892.

[23] Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

[24] Mahajan, M., Nimbhorkar, P., and Varadarajan, K. (2009). The Planar k-Means Problem is NP-Hard. In Das, S. and Uehara, R., editors, *WALCOM: Algorithms and Computation*, pages 274–285, Berlin, Heidelberg. Springer Berlin Heidelberg.

[25] Malinen, M. I. and Fränti, P. (2014). Balanced k-means for clustering. In Fränti, P., Brown, G., Loog, M., Escolano, F., and Pelillo, M., editors, *Structural, Syntactic, and Statistical Pattern Recognition*, pages 32–41, Berlin, Heidelberg. Springer Berlin Heidelberg.

[26] Martí, R., Martínez-Gavara, A., Sánchez-Oro, J., and Duarte, A. (2018). Tabu search for the dynamic Bipartite Drawing Problem. *Computers & OR*, 91:1–12.

[27] Mohammed, S. M., Jacksi, K., and Zeebaree, S. (2021). A state-of-the-art survey on semantic similarity for document clustering using glove and density-based algorithms. *Indonesian Journal of Electrical Engineering and Computer Science*, 22(1):552–562.

[28] Pyatkin, A., Aloise, D., and Mladenović, N. (2017). NP-Hardness of balanced minimum sum-of-squares clustering. *Pattern Recognition Letters*, 97:44–45.

[29] Queiroga, E., Subramanian, A., and dos Anjos F. Cabral, L. (2018). Continuous greedy randomized adaptive search procedure for data clustering. *Applied Soft Computing*, 72:43–55.

[30] Sevaux, M., Rossi, A., Soto, M., Duarte, A., and Martí, R. (2014). GRASP with ejection chains for the dynamic memory allocation in embedded systems. *Soft Comput.*, 18(8):1515–1527.

[31] Shapiro, L. and Stockman, G. (2001). *Computer Vision*. Prentice-Hall, Upper Saddle River, NJ.

[32] Steinhaus, H. (1956). Sur la division des corps matériels en parties. *Bull. Acad. Polon. Sci. Cl. III.*, 4:801–804.

[33] Su, W., Hu, J., Lin, C., and Shen, S. X. (2015). SLA-Aware Tenant Placement and Dynamic Resource Provision in SaaS. In Miller, J. A. and Zhu, H., editors, *ICWS*, pages 615–622. IEEE Computer Society.

[34] Wong, K. (2015). A Short Survey on Data Clustering Algorithms. In *2015 Second International Conference on Soft Computing and Machine Intelligence (ISCMI)*, pages 64–68.

[35] Wong, K. and Zhang, Z. (2014). SNPdryad: predicting deleterious non-synonymous human SNPs using only orthologous protein sequences. *Bioinformatics*, 30(8):1112–1119.

[36] Xavier, A. E. and Xavier, V. L. (2011). Solving the minimum sum-of-squares clustering problem by hyperbolic smoothing and partition into boundary and gravitational regions. *Pattern Recognition*, 44(1):70–77.

[37] Xiong, H., Wu, J., and Chen, J. (2009). K-Means Clustering Versus Validation Measures: A Data-Distribution Perspective. *IEEE Trans. Systems, Man, and Cybernetics, Part B*, 39(2):318–331.

[38] Xu, J. and Lange, K. (2019). Power k-means clustering. In *International Conference on Machine Learning*, pages 6921–6931. PMLR.

[39] Xu, R. and Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678.