Responsive threshold search based memetic algorithm for balanced minimum sum-of-squares clustering

Qing Zhou [a], Jin-Kao Hao [a,b], Qinghua Wu [c,*]

[a] *LERIA, Université d'Angers, 2 bd Lavoisier, 49045 Angers Cedex 01, France, email: qingzhouqz03@gmail.com; jin-kao.hao@univ-angers.fr*

[b] *Institut Universitaire de France, 1 rue Descartes, 75231 Paris, France*

[c] *School of Management, Huazhong University of Science and Technology, No. 1037, Luoyu Road, Wuhan, China, email: qinghuawu1005@gmail.com*

Accepted to ***Information Sciences***, April 2021

Abstract

Clustering is a common task in data mining for constructing well-separated groups (clusters) from a large set of data points. The balanced minimum sum-of-squares clustering problem is a variant of the classic minimum sum-of-squares clustering (MSSC) problem and arises from broad real-life applications where the cardinalities of any two clusters differ by at most one. This study presents the first memetic algorithm for solving the balanced MSSC problem. The proposed algorithm combines a backbone-based crossover operator for generating offspring solutions and a responsive threshold search that alternates between a threshold-based exploration procedure and a descent-based improvement procedure for improving new offspring solutions. Numerical results on 16 real-life datasets show that the proposed algorithm competes very favorably with several state-of-the-art methods from the literature. Key components of the proposed algorithm are investigated to understand their effects on the performance of the algorithm.

Keywords: Balanced clustering; minimum sum-of-squares; memetic algorithm; responsive threshold search; heuristics.

* Corresponding author.

1 Introduction

The minimum sum-of-squares clustering (MSSC) is among the most important tasks in data science and has numerous practical applications in diverse domains. Given a set of n data points $N = \{p_1, p_2, ..., p_n\}$ in a d-dimensional Euclidean space $\mathbb{R}^d$ where the value of each dimension of the data point is a real number that can be positive, negative, or 0, MSSC is to partition N into k (k is given) disjoint subsets C_1, C_2, ..., C_k called clusters such that the sum of the squared distances between each data point to its cluster centroid is minimized. Let w_{ig} be a binary decision variable such that $w_{ig} = 1$ if point p_i is allocated to cluster g ($i \in \{1, 2, ..., n\}$, $g \in \{1, 2, ..., k\}$) and 0 otherwise. The MSSC problem can be formulated as the following optimization problem [4]:

$$\begin{cases} \textbf{min} & \sum_{i=1}^{n} \sum_{g=1}^{k} w_{ig} \left\| p_i - \overline{z}_g \right\|^2 \\ \textbf{s.t.} & \sum_{g=1}^{k} w_{ig} = 1, \quad i \in \{1, 2, ..., n\} \\ & w_{ig} \in \{0, 1\}, \quad i \in \{1, 2, ..., n\}, \quad g \in \{1, 2, ..., k\} \end{cases} \tag{1}$$

where $\overline{z}_g$ is the center (or centroid) of cluster g, given by $\overline{z}_g = \frac{\sum_{i=1}^{n} w_{ig} p_i}{\sum_{i=1}^{n} w_{ig}}$ ($g \in \{1, 2, ..., k\}$); and the norm $\parallel x - y \parallel^2$ designates the squared Euclidean distance between the two points $x \in \mathbb{R}^d$ and $y \in \mathbb{R}^d$, i.e., $\parallel x - y \parallel^2 = \sum_{r=1}^{d} (x_r - y_r)^2$.

MSSC is known to be NP-hard [8]. Accordingly, the problem is computationally challenging. To practically solve MSSC, a variety of heuristic and metaheuristic algorithms have been proposed to find good approximate solutions, as shown in the recent review by Pereira et al. [38]. The popular heuristics for MSSC include the k-means algorithm [23] and its variants, such as global k-means [30], modified global k-means [5], J-means [19] and HG-means [17]. Representative metaheuristic algorithms include tabu search [31], genetic algorithm [27] and variable neighborhood search [38] along with recent incremental algorithm [25], and convex optimization methods [3].

However, in many applications, it is desirable that the data points are evenly distributed among the clusters. Practical examples arise from problems, such as cloud computing [43] and working groups composition [13], where the working groups are required to have approximately the same number of members. This requirement can be formulated by the following balance constraint:

$$\sum_{i=1}^{n} w_{ig} - \sum_{i=1}^{n} w_{ih} \leq 1, g \neq h, 1 \leq g, h \leq k \qquad (2)$$

which states that the sizes of any two clusters differ by at most one.

The balanced MSSC (BMSSC) problem is obtained by adding the balance constraint to the MSSC model. Similar to MSSC, BMSSC is known to be NP-hard for $\frac{n}{k} \geq 3$ in general dimensions [39] and remains NP-hard even for the particular case of two equal sized clusters ($k = 2$) [7].

Given the intrinsic hardness and computational challenge of BMSSC, several heuristic approaches have been used to find sub-optimal solutions within a limited time frame. Malinen et al. [34] developed a k-means-based clustering approach that minimizes the mean square error for given cluster sizes. The algorithm uses the Hungarian method [26] with a time complexity of $O(n^3)$ in the k-means assignment phase, which makes the proposed algorithm more efficient than the previous $O(n^{3.5}k^{3.5})$ time linear programming method used in constrained k-means. This algorithm can deal with datasets with size up to 5000 data points. Costa et al. [12] proposed a variable neighborhood search algorithm for BMSSC based on the nested swap neighborhood. The algorithm showed competitive performance compared with the state-of-the-art algorithms.

Despite the importance of the BMSSC model, few practical algorithms exist for solving BMSSC, and bio-inspired metaheuristics have not been investigated in the context of BMSSC. Many bio-inspired algorithms have been proposed to solve various problems, such as multimodal biomedical image registration [11], large scale feature selection [29], hardware/software co-design [22], multimodal optimization [46], and environmental/economic dispatch problems [40]. More examples can be found in [41,47]. Among the numerous bio-inspired approaches, memetic algorithms (MAs) are known to be particularly suitable for solving difficult problems, including discrete optimization [21,33,35,45], continuous optimization [49], constrained optimization [48], and multi-objective optimization [2,28,42]. However, investigation on the use of the memetic approach to solve BMSSC is still missing in the literature.

This work fills the gap by presenting the first population-based MA for solving BMSSC. The main contributions are summarized as follows:

- The proposed algorithm MA has two original features. First, MA uses a backbone-based crossover operator for offspring solution generation. This crossover operator is specially designed for BMSSC, which generates offspring solutions by inheriting good properties from parent solutions to guide the search towards promising search regions. Second, MA applies the responsive threshold search (RTS) that alternates between a threshold-

based exploration (TBE) procedure and a descent-based improvement (DBI) procedure for local improvement. The TBE procedure explores two neighborhoods and accepts any encountering neighboring solution as long as its objective value is not worse than a responsive threshold value. By contrast the DBI procedure accepts only improved neighboring solutions in the two neighborhoods. The RTS is able to find better solutions in a larger area of the search space without being easily trapped into local optima by alternating between the two complementary procedures.
- Extensive computational assessments are carried out on 16 datasets of 160 real-life benchmark instances, demonstrating the high competitiveness of the proposed MA approach compared with two state-of-the-art algorithms. The results of the proposed MA approach provide a reference for performance assessment of other BMSSC methods for future studies.

The rest of the paper is organized as follows. Section 2 presents the general framework and main components of the proposed algorithm. Section 3 reports the computational evaluation of the algorithm. Section 4 analyzes key algorithmic ingredients, and Section 5 concludes the work.

2 Memetic algorithm for BMSSC

Algorithm 1: Memetic algorithm for BMSSC problem

```
   Input: $I$: a BMSSC instance, $|P|$: population size
   Output: The best solution $S^*$ found
1  $P = \{S^1, S^2, ..., S^{|P|}\} \leftarrow PopulationInitial()$ /* Section 2.2 */
2  $S^* \leftarrow Best(P)$ /* Record the best solution found so far */
3  while Stopping condition is not met do
4      Select two parents $S^a$, $S^b$ from $P$ at random
5      $S \leftarrow CrossOver(S^a, S^b)$ /* Section 2.3, generate an offspring solution */
6      $S \leftarrow ResponsiveThresholdSearch(S)$ /* Section 2.5, improve the solution */
7      if $f(S) < f(S^*)$ then
8          $S^* \leftarrow S$ /* Update the best solution found so far */
9      $P \leftarrow PopulationUpdate(P, S)$ /* Section 2.4, update the population */
```

MA is a general framework that combines the exploration capacity of population-based search and the exploitation ability of local optimization [20,37]. Exploration underlines the capacity to diversify the search to seek promising search regions, whereas exploitation emphasizes the ability to search specific regions in depth. An approach solely focusing on exploration can miss high quality solutions, while a method heavily depending on exploitation will confine itself in a limited search region and be unable to visit other promising regions of the search space. When the method can provide a good balance between exploitation and exploration, a high search performance could be expected. The proposed MA for BMSSC follows the general memetic methodology and offers an appropriate conciliation between exploration and exploitation by combining the backbone-based crossover operator (Section 2.3) and the RTS procedure (Section 2.5) to constitute an effective method. Algorithm 1 shows the general scheme of MA. The algorithm starts with an initial population (line 1) with $|P|$ individuals ($|P|$ is the population size) where each individual is generated by the random construction procedure described in Section 2.2. Then the algorithm repeats an iterative process to improve the population until a stopping condition is met (typically a time limit or a fixed number of generations). At each iteration (generation), two parent solutions are randomly selected from the population (line 4) and recombined by the crossover operator to generate an offspring solution (line 5). The newly generated offspring solution is further improved by the RTS procedure (line 6). Finally, a simple population updating strategy is applied to update the population (line 9, Section 2.4). The main components of MA are presented in the following subsections.

2.1 Search space and evaluation function

Given a BMSSC instance with a set of n data points $N = \{p_1, p_2, ..., p_n\}$ and an integer k, the search space Ω explored by the proposed MA is all partitions of n points into k disjoint clusters C_1, C_2, ..., C_k, such that the cardinalities of any two clusters differ by at most one (i.e., $\Omega = \{\{C_1, C_2, ..., C_k\}, \cup_{g=1}^{k} C_g = N, C_g \cap C_h = \emptyset, |C_g| - |C_h| \leq 1\}$ where $|C_g|$ denotes the cardinality of cluster C_g). According to Huygens' theorem [14], which states that the sum of squares of the Euclidean distances from all data points of a given cluster to their centroid, is equal to the sum of squares of the distances between each pair of points of this cluster divided by its cardinality. Then, the objective function of (1) can be rewritten as follows [12]:

$$\sum_{g=1}^{k} \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ig} w_{jg} \left\| p_i - p_j \right\|^2}{\sum_{i=1}^{n} w_{ig}} \tag{3}$$

The objective value $f(S)$ for any candidate solution S in Ω is evaluated by Eq. (3).

2.2 Population initialization

The $|P|$ individuals of the initial population are constructed as follows. For each individual, a random solution is generated by performing a series of insertion operations until all points are allocated to a cluster. Specifically, a point p is randomly selected for each insertion operation from the set of unallocated points, and a cluster g with a minimum number of allocated points is selected from the set of clusters (ties are broken randomly). Then, p is allocated to cluster g. From an initial solution that each cluster is an empty set, the insertion operation, which allocates a point p to a cluster g having the smallest size, can guarantee that the sizes of any two clusters differ by at most one unity at each time. This generated solution is then improved by the DBI procedure (Section 2.5.2). The improved solution will be inserted into the population if it is not identical to any existing solution from the population. This procedure repeats until the size of the population reaches $|P|$.

2.3 Crossover operator

Algorithm 2: Backbone-based crossover operator

```
   Input: Two parent solutions $S^1 = \{C_1^1, C_2^1, ..., C_k^1\}$ and
          $S^2 = \{C_1^2, C_2^2, ..., C_k^2\}$
   Output: A feasible offspring solution $S^o = \{C_1^o, C_2^o, ..., C_k^o\}$
1  /* Step 1: Construct a partial solution based on backbone */
2  Initializing $C_t^o = \emptyset$ $(t \in \{1, ..., k\})$
3  for $t := 1$ to $k$ do
4      Identify a cluster $C_g^1$ of $S^1$ and a cluster $C_h^2$ of $S^2$ $(g, h \in \{1, ..., k\})$ to
         ensure that $C_g^1$ and $C_h^2$ have the maximum number of identical points
5      $C_t^o \leftarrow C_g^1 \cap C_h^2$
6      Remove all points in $C_t^o$ from all clusters of $S^1$ and $S^2$
7  /* Step 2: Complete the partial offspring solution */
8  $U \leftarrow \{p_1, ..., p_n\} \setminus \{C_1^o \cup ... \cup C_k^o\}$ /* $U$ denotes the set of unallocated points
     */
9  repeat
10     Select a point $p$ from $U$ at random
11     Select a cluster $t$ from $\{C_1^o, C_2^o, ..., C_k^o\}$ with a minimum number of
         points (ties are broken randomly)
12     $C_t^o \leftarrow C_t^o \cup \{p\}$
13     $U \leftarrow U \setminus \{p\}$
14 until $U$ is an empty set
15 return $S^o$
```

The proposed MA algorithm uses a backbone-based crossover operator (denoted by BCX) to generate promising offspring solutions from existing solutions in the population. To ensure that the crossover plays its role, it should be designed in a way that it can transmit good properties from parents to offspring solutions with respect to the given optimization objective [20]. As presented in Section 2.1, a solution of BMSSC is a partition of the set of n data points into k disjoint clusters. As such, it is more meaningful for a crossover to manipulate groups of points rather than single point. Crossovers using such an idea have been successfully applied to solve a number of “grouping” problems, such as graph coloring [24] and graph partitioning [6].

The BCX generates an offspring solution by inheriting good properties from two randomly selected parent solutions to guide the search to promising search regions in accordance with the general principle presented above.

Definition 1 (backbone): Given two parent solutions $S^1 = \{C_1^1, C_2^1, ..., C_k^1\}$ and $S^2 = \{C_1^2, C_2^2, ..., C_k^2\}$, the backbone of S^1 and S^2 (denoted as B) is the set of k subsets of points $\{B^1, B^2, ..., B^k\}$, such that B^t ($t \in \{1, 2, ..., k\}$) is the subset of points that are grouped together in both S^1 and S^2, i.e., $B^t = C_g^1 \cap C_h^2$ ($g, h \in \{1, 2, ..., k\}$), while $B^1 \cup B^2 \cup ... \cup B^k$ is as large as possible.

The proposed crossover BCX generates an offspring solution in two steps, as illustrated in Algorithm 2. BCX first constructs a partial solution by preserving the backbone of the two selected parent solutions and completes the partial solution in a random way.

Construct a partial solution based on backbone: To produce a partial solution based on the backbone, a cluster matching procedure is needed to identify an optimal matching between clusters of two parent solutions such that it preserves as many common points in both parent solutions as possible. This task amounts to the maximum weight matching problem in an edge weight complete bipartite graph $H = (V', E')$, where V' consists of k left vertices and k right vertices, which corresponds to the clusters of the parent solutions S^1 and S^2, respectively; each edge $(C_g^1, C_h^2) \in E'$ is associated with a weight $w_{C_g^1, C_h^2}$, which is the number of identical points in C_g^1 of S^1 and C_h^2 of S^2. This maximum weight matching problem can be optimally solved by the classic Hungarian algorithm [26]. Nevertheless, running the Hungarian algorithm for each crossover will be too time consuming. Therefore, a fast greedy matching procedure (lines 3 to 6) is applied to find a near-optimal matching between the clusters of the parent solutions. This matching procedure creates the partial solution S^o in k iterations, and at each iteration t, it builds the cluster C_t^o of S^o as follows. The procedure first identifies a cluster C_g^1 of S^1 and a cluster C_h^2 of S^2 ($g, h \in \{1, ..., k\}$) such that C_g^1 and C_h^2 have the maximum number of identical points across all cluster combinations of S^1 and S^2, i.e., $max_{g,h \in \{1,...,k\}} |C_g^1 \cap C_h^2|$. Then, the backbone (i.e., all identical points) for each identified pair of matched clusters is preserved to the corresponding cluster of the offspring solution, i.e., $C_t^o = C_g^1 \cap C_h^2$. At the end of iteration t, the points that belong to the cluster C_t^o are removed from the clusters of S^1 and S^2.

Complete the partial solution: To complete the partial offspring solution, the unallocated points are assigned to the clusters on the basis of a random construction strategy. Specifically, a series of insertion operations are performed until all unallocated points are assigned. For each insertion operation, an unallocated point p is randomly selected and allocated to a cluster C_t^o of S^o with the minimum number of points (ties are broken randomly). From a partial offspring solution, the random construction strategy, which allocates an unallocated point to a cluster with the minimum size each time until each unallocated point is allocated, ensures that the completed offspring solution satisfies the balance constraint.

2.4 Population updating rule

Once the offspring solution is generated by the backbone-based crossover operator, it is first submitted for improvement by the RTS procedure presented in Section 2.5. Then, a classic population updating rule is used to update the population as follows. If the offspring solution is not the same as any existing solution in the population and has a better quality than the worst solution of the population, then the offspring replaces the worst solution in the population; otherwise the offspring is ignored, and the population is kept unchanged.

2.5 Responsive threshold search

Algorithm 3: Responsive threshold search algorithm

```
   Input: $S$: an input solution, $R_{iter}$: search depth of RTS
   Output: The best solution $S^*$ found
1  $ic \leftarrow 0$ /* Iteration counter */
2  $S^* \leftarrow S$ /* Record the best solution found so far */
3  $f_b \leftarrow f(S^*)$ /* $f_b$ records the objective value of the best local optimum */
4  $tr \leftarrow compute\_threshold\_ratio(f_b)$ /* $tr$ indicates the threshold ratio,
     Section 2.5.1 */
5  $tvT \leftarrow compute\_threshold\_value(f_b, tr)$ /* $tvT$ denotes the threshold value,
     Section 2.5.1 */
6  while $ic < R_{iter}$ do
7      $(S, S^*) \leftarrow threshold\_based\_exploration(S, tvT)$ /* Section 2.5.1 */
8      $(S, S^*) \leftarrow descent\_based\_improvement(S)$ /* Section 2.5.2 */
9      if $f(S^*) < f_b$ then
10         $f_b \leftarrow f(S^*)$
11         $tr \leftarrow compute\_threshold\_ratio(f_b)$
12         $tvT \leftarrow compute\_threshold\_value(f_b, tr)$
13     $ic \leftarrow ic + 1$
14 return $S^*$
```

The proposed MA algorithm uses a responsive threshold search (RTS) method as its local optimization component. RTS is designed to ensure intensification and diversification of the search space.

Basically, RTS alternates between a threshold-based exploration (TBE) procedure and a descent-based improvement (DBI) procedure. TBE aims to explore in a large zone of the search space by accepting any encountered neighboring solution as long as it satisfies a quality threshold value, which is dynamically adjusted relying on the best local optimum found so far by jointly applying two neighborhoods induced by two move operators (Section 2.5.1). To complement this exploratory procedure, DBI is applied to ensure a more directed and focused search by accepting only improving neighboring solutions. Thus, TBE and DBI rely on the same neighborhoods, but they use different criteria to accept neighboring solutions. RTS enlarges its search scope to find better solutions and avoid being easily trapped into local optima by alternating between TBE and DBI.

Algorithm 3 summarizes the general scheme of the RTS approach. Starting from an initial solution S, RTS first calculates the threshold ratio tr and responsive threshold value tvT (Section 2.5.1), which are dynamically updated by the objective value f_b of the best local optimum. Afterward, the search enters into the main loop. For each iteration, RTS first applies the TBE procedure to explore a large area of the search space by accepting neighboring solutions with objective values not worse than the given threshold tvT. Then, RTS switches to the DBI procedure to intensify the search by accepting only improving neighboring solutions. The DBI procedure stops when a local optimum is reached. If this local optimum is better than the recorded best local optimum, then the recorded best objective value f_b is updated. In this case, the threshold ratio tr and the responsive threshold value tvT are also updated accordingly before moving to the next round of Exploration-Improvement searches. The above process is repeated until a prefixed number of iterations R_{iter} is reached, where R_{iter} is a parameter called the search depth of RTS.

2.5.1 Threshold-based exploration procedure

The TBE procedure systematically exploits two neighborhoods, i.e., the insertion neighborhood N_i induced by the $OneMove$ operator and the swap neighborhood N_s induced by the $SwapMove$ operator. The $SwapMove$ operator has been successfully applied in previous work [12] while the $OneMove$ operator is introduced for the first time in this study for BMSSC. In the succeeding sections, these two move operators are introduced in detail.

OneMove operator: Given a solution $S = \{C_1, C_2, ..., C_k\}$, the $OneMove$ operator transfers a point p from its current cluster C_g to another cluster C_h such that $|C_g| > |C_h|$. To rapidly evaluate the move gain induced by an $OneMove$ move, which indicates how much a solution is improved in terms of the objective value, this study adopts the fast increment evaluation technique.

Algorithm 4: Threshold-based exploration search algorithm

```
   Input: $S$: an input solution; $tvT$: threshold value; $LL$: exploration strength
          of TBE
   Output: A final solution $S$ and the best solution $S^*$ found
 1 $NH \leftarrow \{N_i,\ N_s\}$ /* $NH$ is composed of the two considered neighborhoods
   */
 2 $S^* \leftarrow S$ /* $S^*$ records the best solution found so far */
 3 for $ic := 1$ to $LL$ do
 4     for each neighborhood $nh \in NH$ do
 5         Randomly shuffle all neighboring solutions of $nh$ in the set of
           solutions $NB\_SET(S, nh)$
 6         for each neighboring solution $S' \in NB\_SET(S, nh)$ do
 7             if $f(S') < tvT$ then
 8                 $S \leftarrow S'$
 9             if $f(S') < f(S^*)$ then
10                 $S^* \leftarrow S'$
11 return $S$ and $S^*$
```

The main idea is to maintain a $n \times k$ matrix γ where its element $\gamma[p][g]$ records the summation of squared Euclidean distances between a point p to all points in cluster C_g, i.e., $\gamma[p][g] = \sum_{q \in C_g} \|p - q\|^2$. Another k-dimensional vector β is maintained where its element $\beta[g]$ denotes the summation of squared distances between each pair of points in cluster C_g, i.e., $\beta[g] = \frac{\sum_{p \in C_g} \gamma[p][g]}{2}$. With these two memories, the move gain of an *OneMove* operation denoted by $OneMove(p, C_g, C_h)$ can be efficiently calculated by:

$$\Delta_f(OneMove(p, C_g, C_h)) = \frac{\beta[g] - \gamma[p][g]}{|C_g| - 1} + \frac{\gamma[p][h] + \beta[h]}{|C_h| + 1} - (\frac{\beta[g]}{|C_g|} + \frac{\beta[h]}{|C_h|})$$

After each $OneMove(p, C_g, C_h)$ operation, a subset of values in γ and β affected by the move is conveniently updated as follows: $\beta[g] = \beta[g] - \gamma[p][g], \beta[h] = \beta[h] + \gamma[p][h], \gamma[q][g] = \gamma[q][g] - \|p - q\|^2, \gamma[q][h] = \gamma[q][h] + \|p - q\|^2, \forall q \in N$. The time complexity of updating β and γ is bounded by $O(1)$ and $O(n)$, respectively.

SwapMove operator: The *SwapMove* operator exchanges two points p and q from two different clusters C_g and C_h. The move gain of a *SwapMove* operation denoted as $SwapMove(p, q)$ can be efficiently computed by:

$$\Delta_f(SwapMove(p, q)) = \frac{\gamma[q][g] - \gamma[p][g] - \|p - q\|^2}{|C_g|} + \frac{\gamma[p][h] - \gamma[q][h] - \|p - q\|^2}{|C_h|}$$

Given that a *SwapMove* operation can be decomposed into two consecutive *OneMove* moves, the matrix γ and the vector β are consecutively updated two times according to the corresponding *OneMove* operation.

When several neighborhoods are available, one important issue is how to integrate these neighborhoods to efficiently explore the search space. Several ways can be used to combine different neighborhoods proposed in the literature, such as token-ring and neighborhood union [32]. A key motivation for considering a combination of multiple neighborhoods is to allow the search to go beyond local optima and continue its exploration toward still better solutions. In the TBE procedure, the two neighborhoods N_i and N_s induced by the $OneMove$ and the $SwapMove$ operators, respectively are examined in a token-ring way, i.e., $N_i \rightarrow N_s \rightarrow N_i \rightarrow ....$ The general scheme of the TBE procedure is summarized in Algorithm 4. Starting from an initial solution S, TBE explores the two neighborhoods N_i and N_s in turn. For each considered neighborhood, every move in this neighborhood is examined in a random order, and the resulting neighboring solution S' is accepted if its objective value is not worse than the given threshold value tvT, i.e., $f(S') \leq tvT$. The improving and deteriorating solutions are allowed in the TBE procedure as long as its solution quality meets the threshold accepting rule. In this way, a high number of moves become available, enabling a large exploration of the search space and thus enhancing the exploration capability of the approach.

The responsive threshold value tvT is critical to the performance of the TBE procedure. A particularly large threshold value ($tvT - f_b$ is very large) may lead the TBE procedure to a purely random approach, while an extremely small threshold value ($tvT - f_b$ is very small) can greatly weaken the exploration power of the TBE procedure. Following [10], the threshold value tvT is dynamically tuned according to the objective value of the recorded best local optimum f_b and the threshold ratio tr, i.e., $tvT = (1 + tr) \times f_b$. Thus, the responsive threshold value tvT dynamically evolves with the best solution found during the search process. In general, as the RTS proceeds, the local optimum of better quality will be found, leading to a stricter acceptance criterion of better candidate solutions in the subsequent search.

To determine a proper value for the threshold ratio tr and inspired by the work of [10], this study adopts a proportional function where tr strictly decreases when f_b decreases: $tr = \frac{1}{\frac{ta \times 10000}{fb} + tb} + tc$, where ta, tb, and tc are three fixed coefficients determined empirically, i.e., $ta = 16.98, tb = 76.81, tc = 0.0031$. To identify appropriate values for ta, tb, and tc, this study chooses three instances from three datasets (Iris, Glass, and Vehicle), whose best objective values vary from small to large, and keeps a fixed value of tr for each of the three selected instances. For each instance, the obtained best objective value is used as f_b, and then three pairs of (f_b, tr) are obtained. The values of ta, tb and tc are obtained by solving the three equations. During the search process of the RTS, tr is dynamically recomputed each time f_b is updated.

Algorithm 5: Descent-based improvement algorithm

```
   Input: $S$: an input solution
   Output: A final solution $S$ and the best solution $S^*$ found
 1 $NH \leftarrow \{N_i,\ N_s\}$ /* $NH$ is composed of the two considered neighborhoods */
 2 $S^* \leftarrow S$
 3 $flag\_imp \leftarrow true$
 4 while $flag\_imp = true$ do
 5     $flag\_imp \leftarrow false$
 6     for each neighborhood $nh \in NH$ do
 7         Randomly shuffle all neighboring solutions of $nh$ in the set of solutions $NB\_SET(S, nh)$
 8         for each neighboring solution $S' \in NB\_SET(S, nh)$ do
 9             if $f(S') < f(S)$ then
10                 $S \leftarrow S'$
11                 $flag\_imp \leftarrow true$
12             if $f(S') < f(S^*)$ then
13                 $S^* \leftarrow S'$
14 return $S$ and $S^*$
```

2.5.2 Descent-based improvement procedure

After the TBE procedure, RTS continues its search with the descent-based improvement (DBI) procedure (Algorithm 5) that aims at finding new local optima of increasing quality. Starting with a solution returned by the TBE procedure, DBI iteratively explores the two neighborhoods N_i and N_s in a token-ring manner $N_i \rightarrow N_s \rightarrow N_i \rightarrow$ .... For each considered neighborhood, a neighboring solution S' is selected at random and replaces the current solution S if S' is better than S, i.e., $f(S') < f(S)$. This procedure stops when no improving solutions can be found in neighborhoods N_i and N_s. In this case, a local optimum is achieved.

3 Computational experiments

This section reports the computational results of the proposed MA algorithm and makes comparisons with state-of-the-art approaches to further evaluate the effectiveness of the proposed MA algorithm.

Table 1
List of datasets.

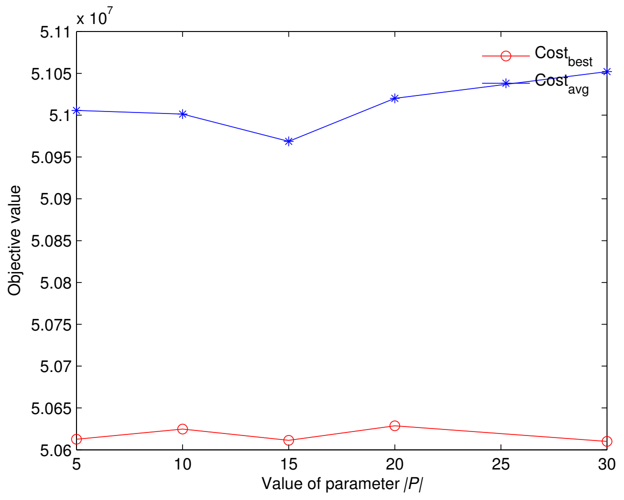
Dataset	n	d
Iris	150	4
Wine	178	13
Glass	214	10
Thyroid	215	5
Ionosphere	351	34
Libra	360	90
User knowledge	403	5
Body measurements	507	5
Water treatment plant	527	38
Breast cancer	569	30
Synthetic control	600	60
Vehicle	846	18
Vowel recognition	990	10
Yeast	1484	8
Multiple features	2000	240
Image segmentation	2310	19

Table 2
Parameter levels of the 2-level full factorial experiment.

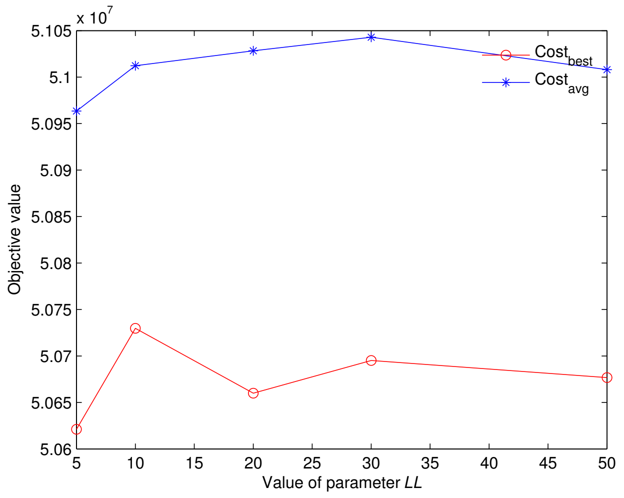
Parameter	low level	high level
$\|P\|$	10	15
LL	5	20
R_{iter}	50	100

3.1 Benchmark instances and experimental protocol

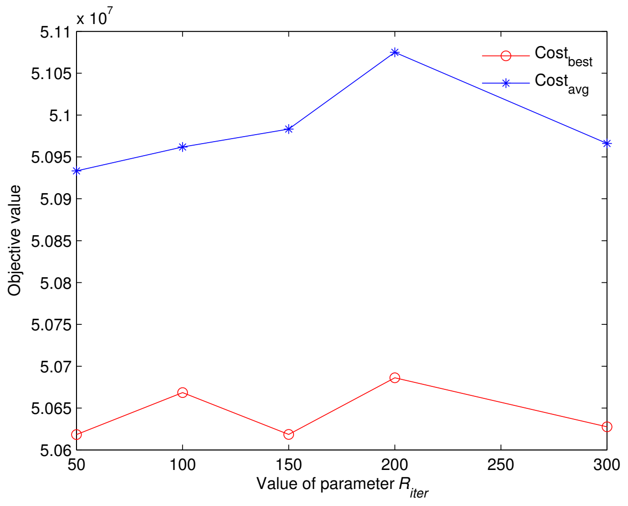
The MA algorithm is assessed on 16 datasets from the UCI machine learning repository[1]. Table 1 provides details of these datasets. Column “Dataset” indicates name of the dataset. Columns “n” and “d” refer to the number of points contained in the dataset and its dimensions, respectively. Following [3,17], this study generates for each dataset 10 benchmark instances with 10 different number of clusters $k \in \{2, 3, 4, 6, 7, 10, 11, 13, 15, 20\}$, leading to various BMSSC test set-ups. Thus, a total of 160 instances is obtained. The proposed algorithm was programmed in the C/C++ language[2] and compiled with GNU g++ 4.8.3 compiler, running on a computing platform with an Intel I7-4790 processor (3.6 GHz) and 8 GB RAM.



(a) parameter $|P|$



(b) parameter LL



(c) parameter R_{iter}

Fig. 1. Analysis of influence of the key parameters ($|P|$, LL, R_{iter}).

Table 3
p-values for the variance analysis with the significance level of 0.05.

Source of variance	$\|P\| * LL$	$\|P\| * R_{iter}$	$LL * R_{iter}$	$\|P\| * LL * R_{iter}$
p-value	0.83	0.35	0.32	0.64

Table 4
Settings of parameters.

Parameter	Section	Description	Final value
$\|P\|$	2, 2.2	size of population	15
LL	2.5.1	exploration strength of the TBE procedure	5
R_{iter}	2.5	search depth of RTS	50

3.2 Parameter settings

MA is controlled by three parameters: the population size $|P|$, the exploration strength of the TBE procedure LL, and the search depth of the RTS R_{iter}. To determine the appropriate parameter settings for MA, a two-level full factorial experiment [36] is first conducted to observe the interaction effects among the three parameters with the levels of each parameter shown in Table 2. Given that each parameter has two levels, there are eight ($2^3 = 8$) combinations for the three parameters. For the experiment, 30 instances were used from three randomly selected datasets (Wine, Breast cancer, and Vehicle), and each instance was independently solved 10 times for each combination of the parameters. Then, the average result of the best objective values obtained on the 30 instances is considered for each execution. Table 3 shows the analysis of the variances of the considered results, and the last four columns indicate that the interaction effects among the parameters are not statistically significant (p-values > 0.05). Therefore, a one-at-a-time tuning experiment [18] is carried out to calibrate these parameters and analyze the influence of a parameter on the algorithm's performance. The best configuration of this tuning experiment is shown in Table 4. For this experiment, a set of potential values are tested for each parameter while fixing other parameters to their default values as shown in Table 4. Specifically, the three parameters are tested in the following three sets of potential values: $|P| \in \{5, 10, 15, 20, 30\}$, $LL \in \{5, 10, 20, 30, 50\}$, and $R_{iter} \in \{50, 100, 150, 200, 300\}$.

The behavior of MA on each parameter is summarized in Fig. 1, where the X-axis indicates the tested parameter values and the Y-axis shows the average value over all the 30 selected instances in terms of the best/average objective value across 10 independent runs. Fig. 1 demonstrates that the value of each

[1] http://www.ics.uci.edu/ mlearn/MLRepository.html
[2] The code of the proposed algorithm will be publicly available at: `http://www.info.univ-angers.fr/pub/hao/MABMSSC.html`

parameter affects the algorithm's performance. For the parameter $|P|$, the value of 15 is the best choice, while a larger or a smaller value could weaken the performance of MA. This notion can be explained by the fact that a large value of $|P|$ (a large population size) makes the algorithm slowly converge within a given cutoff time, and a small value of $|P|$ (a small population size) cannot maintain a suitable population diversity during the search process, which makes the algorithm prematurely converge. MA obtains the best performance with the value of five for LL and larger values decrease its performance. In addition, $R_{iter} = 50$ is the best choice for MA. This experiment justifies the parameter setting in Table 4, and these parameter values were consistently used in all the following experiments.

3.3 Computational comparison with reference algorithms and managerial insights

According to the computational results reported in one of the latest studies for BMSSC [12], the balanced k-means algorithm (bk-means) [34] and the variable neighborhood search based on the "Less Is More Approach" (VNS-LIMA) [12] exhibit an overall best performance among all the existing heuristics for BMSSC. Thus, these two heuristic approaches can be considered as the state-of-the-art algorithms for BMSSC, and they are used as the reference algorithms in the computational comparisons.

To ensure a fair comparison between MA and the two reference algorithms bk-means and VNS-LIMA, this study used the source codes of these two algorithms and directly ran them under the computing platform, as described in Section 3.1. The source code of the bk-means algorithm (in Matlab) is available at `http://www2.uef.fi/en/sipu/data-and-software`. The source code (in C++) of the VNS-LIMA algorithm was kindly provided by the corresponding author of [12]. In accordance with the same experimental protocol as used in [12], each compared algorithm was performed 10 independent runs for each instance. The running time elapsed by the bk-means algorithm for each run was used as the time limit for VNS-LIMA and MA.

Table 5 presents the comparative results of these three compared heuristic approaches. Column "k" indicates the number of clusters used in each dataset. Columns "*cost*" and "$time(s)$" refer to the best objective values obtained by the three compared algorithms and the average CPU times in seconds across 10 independent runs in executing the compared algorithms, respectively. Columns "$Best_{Dev}$" and "Avg_{Dev}" show the percentage deviation of the best objective value and the average objective value over 10 independent executions from the best solution value given in column "*cost*" for the compared algorithms, respectively. For an instance, the percentage deviation of the best and the

Table 5
Comparative results between MA and the two reference algorithms bk-means and VNS-LIMA in terms of percentage deviation from the best solution values obtained by all the three compared algorithms. The best results are marked in bold (part I).

			bk-means		VNS-LIMA		MA	
k	*cost*	*time(s)*	$Best_{Dev}$	Avg_{Dev}	$Best_{Dev}$	Avg_{Dev}	$Best_{Dev}$	Avg_{Dev}
Iris								
2	2.228128e+002	0.58	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
3	8.136720e+001	0.46	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
4	1.112496e+002	0.69	0.03	1.45	**0.00**	1.80	**0.00**	**0.00**
6	4.320800e+001	0.73	0.03	1.03	**0.00**	**0.00**	**0.00**	**0.00**
7	6.192013e+001	0.67	2.07	3.89	2.58	4.79	**0.00**	**1.49**
10	4.488400e+001	0.90	0.53	1.13	**0.00**	0.06	**0.00**	**0.02**
11	3.473445e+001	0.84	8.90	13.71	1.90	8.35	**0.00**	**2.75**
13	3.025152e+001	0.79	11.89	16.33	7.36	10.92	**0.00**	**3.32**
15	2.190800e+001	0.57	2.99	6.17	**0.00**	0.72	**0.00**	**0.69**
20	1.797000e+001	0.64	16.99	**22.44**	**0.00**	23.23	21.96	25.22
Wine								
2	6.507529e+006	0.61	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
3	2.962226e+006	0.48	**0.00**	0.42	**0.00**	0.62	**0.00**	**0.00**
4	1.904950e+006	0.59	**0.00**	0.18	**0.00**	0.38	**0.00**	**0.00**
6	1.008776e+006	0.51	0.36	2.14	0.36	1.98	**0.00**	**0.00**
7	7.345635e+005	0.55	2.34	3.21	0.49	4.00	**0.00**	**0.00**
10	5.061534e+005	0.63	1.34	1.90	0.58	1.57	**0.00**	**0.06**
11	4.327903e+005	0.71	0.33	0.87	**0.00**	2.76	**0.00**	**0.00**
13	3.601952e+005	0.72	3.37	6.39	2.59	7.12	**0.00**	**0.01**
15	2.764871e+005	0.82	7.36	9.63	0.89	7.01	**0.00**	**0.00**
20	1.737925e+005	0.90	0.43	6.27	0.53	12.21	**0.00**	**0.03**
Glass								
2	1.036726e+003	3.26	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
3	8.325955e+002	2.14	**0.00**	3.66	**0.00**	0.35	**0.00**	**0.00**
4	7.142026e+002	2.16	0.37	3.99	**0.00**	0.20	**0.00**	**0.01**
6	5.829703e+002	3.26	1.16	2.54	1.17	1.71	**0.00**	**0.00**
7	5.046425e+002	1.96	1.36	2.61	**0.00**	0.91	**0.00**	**0.08**
10	4.185386e+002	1.89	1.28	2.83	0.30	1.60	**0.00**	**0.47**
11	4.038911e+002	2.08	1.06	3.48	0.03	0.90	**0.00**	**0.45**
13	3.724880e+002	2.07	1.64	4.58	1.08	2.64	**0.00**	**0.56**
15	3.478221e+002	1.96	3.93	6.43	1.17	1.88	**0.00**	**0.60**
20	2.805971e+002	1.73	5.31	12.31	2.13	4.33	**0.00**	**2.64**
Thyroid								
2	4.150596e+004	0.93	0.33	0.35	0.33	0.34	**0.00**	**0.30**
3	3.441325e+004	1.60	**0.00**	0.10	**0.00**	0.07	**0.00**	**0.00**
4	3.004778e+004	1.77	**0.00**	0.38	**0.00**	0.29	**0.00**	**0.00**
6	2.295611e+004	2.76	2.70	3.32	2.70	3.61	**0.00**	**1.96**
7	2.084779e+004	1.65	1.41	1.96	0.41	1.82	**0.00**	**0.00**
10	1.582070e+004	2.50	0.20	2.62	0.28	1.26	**0.00**	**0.09**
11	1.380010e+004	1.57	6.47	8.55	5.00	7.04	**0.00**	**4.47**
13	1.272262e+004	1.78	0.25	2.19	0.16	1.45	**0.00**	**0.06**
15	1.177890e+004	1.82	1.49	4.10	0.37	1.47	**0.00**	**0.34**
20	8.687001e+003	1.49	5.05	7.30	3.17	5.19	**0.00**	**1.35**
Ionosphere								
2	2.425624e+003	5.30	0.34	0.41	0.34	0.38	**0.00**	**0.17**
3	2.295170e+003	9.21	**0.00**	0.24	**0.00**	**0.00**	**0.00**	**0.00**
4	2.131964e+003	9.69	0.42	0.48	0.33	0.39	**0.00**	**0.30**
6	1.938683e+003	7.64	0.93	1.90	0.02	0.24	**0.00**	**0.03**
7	1.861279e+003	8.53	1.22	2.02	**0.00**	0.16	**0.00**	**0.04**
10	1.689036e+003	7.62	1.01	1.76	0.08	0.24	**0.00**	**0.12**
11	1.644647e+003	7.22	1.73	2.18	0.18	0.51	**0.00**	**0.27**
13	1.569937e+003	7.13	1.62	2.66	0.08	0.52	**0.00**	**0.39**
15	1.505582e+003	7.71	1.95	2.48	0.32	0.66	**0.00**	**0.18**
20	1.397595e+003	6.57	3.19	4.90	0.70	1.09	**0.00**	**0.41**

Table 5
Comparative results between MA and the two reference algorithms bk-means and VNS-LIMA in terms of percentage deviation from the best solution values obtained by all the three compared algorithms. The best results are marked in bold (part II).

			bk-means		VNS-LIMA		MA	
k	$cost$	$time(s)$	$Best_{Dev}$	Avg_{Dev}	$Best_{Dev}$	Avg_{Dev}	$Best_{Dev}$	Avg_{Dev}
Libra								
2	8.345947e+002	12.10	0.01	0.91	**0.00**	**0.00**	**0.00**	**0.00**
3	7.076289e+002	10.45	0.02	0.70	**0.00**	**0.00**	**0.00**	**0.00**
4	6.276321e+002	13.29	0.02	0.22	**0.00**	**0.00**	**0.00**	**0.00**
6	5.287864e+002	16.80	0.11	0.83	**0.00**	0.33	**0.00**	**0.06**
7	4.882661e+002	6.16	0.44	2.30	0.27	1.26	**0.00**	**0.32**
10	4.238042e+002	7.50	0.81	2.23	0.51	1.42	**0.00**	**0.45**
11	4.037304e+002	6.86	2.29	4.59	0.68	1.98	**0.00**	**0.65**
13	3.665767e+002	8.69	2.37	4.52	1.01	3.06	**0.00**	**0.90**
15	3.362085e+002	7.14	2.19	5.35	0.91	2.89	**0.00**	**1.40**
20	2.874700e+002	7.62	1.40	3.63	0.86	**2.29**	**0.00**	2.36
User knowledge								
2	9.149010e+001	14.85	0.09	2.44	0.09	0.10	**0.00**	**0.03**
3	7.815654e+001	9.78	0.01	1.14	0.01	0.03	**0.00**	**0.00**
4	7.020935e+001	8.98	0.23	0.78	0.01	0.56	**0.00**	**0.10**
6	5.786701e+001	14.20	0.15	2.94	0.05	0.19	**0.00**	**0.01**
7	5.390618e+001	7.73	0.20	1.27	0.07	0.46	**0.00**	**0.30**
10	4.458389e+001	10.35	1.06	2.92	0.84	1.91	**0.00**	**0.54**
11	4.247816e+001	9.49	0.78	2.73	0.78	1.89	**0.00**	**0.73**
13	3.915801e+001	8.62	0.59	3.50	0.02	1.73	**0.00**	**0.49**
15	3.579933e+001	9.45	1.74	5.32	2.81	3.85	**0.00**	**0.68**
20	3.061470e+001	7.45	4.25	5.22	2.85	5.35	**0.00**	**1.88**
Body								
2	1.134111e+005	13.18	0.19	0.25	0.19	0.22	**0.00**	**0.15**
3	8.092259e+004	12.03	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
4	6.739189e+004	12.04	0.20	0.25	0.20	0.27	**0.00**	**0.18**
6	5.476624e+004	14.87	0.07	0.21	0.03	0.15	**0.00**	**0.01**
7	4.943618e+004	19.08	0.24	0.63	0.04	0.21	**0.00**	**0.03**
10	4.005231e+004	15.42	0.74	1.21	0.45	0.85	**0.00**	**0.37**
11	3.748249e+004	10.20	0.08	2.76	0.17	**1.16**	**0.00**	1.20
13	3.416424e+004	14.70	0.14	1.03	0.12	0.98	**0.00**	**0.88**
15	3.177629e+004	15.53	1.60	1.95	1.09	1.68	**0.00**	**0.25**
20	2.723185e+004	11.42	1.68	4.26	1.94	2.81	**0.00**	**1.54**
Water treatment plant								
2	2.685423e+010	11.02	**0.00**	**0.00**	**0.00**	0.01	**0.00**	**0.00**
3	2.065459e+010	12.81	0.68	0.68	0.62	0.66	**0.00**	**0.56**
4	1.759911e+010	11.74	1.03	1.10	0.87	1.01	**0.00**	**0.70**
6	1.399003e+010	10.68	0.44	0.50	**0.00**	0.34	**0.00**	**0.00**
7	1.278665e+010	15.94	0.34	0.78	0.23	0.38	**0.00**	**0.21**
10	9.842905e+009	15.77	0.42	1.24	0.41	1.33	**0.00**	**0.26**
11	8.777980e+009	17.67	3.57	4.23	3.54	4.99	**0.00**	**3.18**
13	7.742797e+009	19.02	0.03	1.09	0.16	0.86	**0.00**	**0.00**
15	6.544925e+009	18.01	0.03	0.81	**0.00**	0.06	**0.00**	**0.00**
20	3.516027e+009	22.00	0.13	0.52	0.03	3.34	**0.00**	**0.00**
Breast cancer								
2	1.366899e+008	23.04	0.61	0.78	0.61	0.70	**0.00**	**0.49**
3	8.743161e+007	11.30	0.64	0.64	**0.00**	0.45	**0.00**	**0.00**
4	5.978607e+007	9.43	**0.00**	0.17	**0.00**	0.24	**0.00**	**0.00**
6	3.973995e+007	8.63	**0.00**	0.40	**0.00**	0.56	**0.00**	**0.00**
7	3.468144e+007	8.27	0.03	0.33	0.01	0.32	**0.00**	**0.01**
10	2.593484e+007	12.61	**0.00**	0.84	**0.00**	0.70	**0.00**	**0.00**
11	2.378349e+007	12.57	1.53	1.68	0.12	1.14	**0.00**	**0.00**
13	2.060293e+007	12.51	1.44	1.45	0.01	1.15	**0.00**	**0.01**
15	1.858171e+007	15.76	1.55	1.62	1.40	1.54	**0.00**	**0.01**
20	1.455947e+007	19.20	0.13	1.10	0.10	1.29	**0.00**	**0.01**
Synthetic control								
2	2.073382e+006	34.00	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
3	1.239480e+006	15.82	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
4	1.269957e+006	30.90	0.38	0.48	**0.00**	0.13	**0.00**	**0.07**
6	1.001825e+006	19.42	**0.00**	0.13	**0.00**	0.03	**0.00**	**0.00**
7	1.026072e+006	25.52	0.22	0.90	0.14	0.51	**0.00**	**0.08**

Table 5
Comparative results between MA and the two reference algorithms bk-means and VNS-LIMA in terms of percentage deviation from the best solutions obtained by all the three compared algorithms. The best results are marked in bold (part III).

			bk-means		VNS-LIMA		MA	
k	*cost*	*time(s)*	$Best_{Dev}$	Avg_{Dev}	$Best_{Dev}$	Avg_{Dev}	$Best_{Dev}$	Avg_{Dev}
Synthetic control								
10	8.986148e+005	21.66	0.37	0.85	**0.00**	**0.18**	**0.00**	0.23
11	8.677623e+005	34.98	1.22	4.30	0.31	0.58	**0.00**	**0.55**
13	7.945551e+005	22.98	0.32	1.86	0.19	0.95	**0.00**	**0.08**
15	7.221920e+005	17.89	0.21	1.54	**0.00**	0.27	**0.00**	**0.00**
20	6.698253e+005	18.31	1.67	1.86	0.80	2.54	**0.00**	**0.61**
Vehicle								
2	1.234444e+007	188.50	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
3	5.565073e+006	72.80	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
4	4.307852e+006	74.11	0.33	0.43	0.33	0.45	**0.00**	**0.18**
6	2.894321e+006	49.00	**0.00**	0.24	**0.00**	**0.00**	**0.00**	**0.00**
7	2.547200e+006	60.98	0.53	0.70	0.53	0.64	**0.00**	**0.42**
10	1.982152e+006	67.32	0.78	1.00	0.71	0.93	**0.00**	**0.50**
11	1.884479e+006	74.11	0.03	0.20	**0.00**	0.18	**0.00**	**0.00**
13	1.715279e+006	66.06	0.03	0.25	0.01	0.09	**0.00**	**0.00**
15	1.542432e+006	96.79	0.22	0.81	0.15	0.47	**0.00**	**0.07**
20	1.255201e+006	63.51	0.83	1.41	0.14	0.59	**0.00**	**0.19**
Vowel recognition								
2	3.990564e+003	344.33	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
3	3.420870e+003	281.01	0.01	0.04	**0.00**	**0.00**	**0.00**	**0.00**
4	3.068941e+003	219.45	0.18	0.20	0.04	0.06	**0.00**	**0.04**
6	2.614537e+003	156.23	0.45	1.44	0.04	0.36	**0.00**	**0.01**
7	2.458453e+003	141.74	0.33	1.35	0.10	0.61	**0.00**	**0.11**
10	2.059913e+003	137.61	0.10	1.70	0.20	1.13	**0.00**	**0.23**
11	1.965081e+003	124.00	0.01	**0.34**	0.23	0.88	**0.00**	0.38
13	1.808734e+003	119.42	0.06	1.79	0.39	0.86	**0.00**	**0.28**
15	1.680291e+003	113.50	1.73	2.69	0.53	1.73	**0.00**	**1.04**
20	1.453449e+003	110.67	1.06	3.34	1.06	2.15	**0.00**	**1.03**
Yeast								
2	9.773783e+001	3519.64	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
3	8.618621e+001	1920.44	**0.00**	0.57	0.01	0.44	**0.00**	**0.16**
4	7.674756e+001	1305.07	**0.00**	1.00	**0.00**	0.01	**0.00**	**0.00**
6	6.355879e+001	913.82	0.03	0.12	0.01	0.05	**0.00**	**0.00**
7	5.994364e+001	499.83	0.17	0.21	0.01	0.05	**0.00**	**0.00**
10	5.320666e+001	399.93	0.07	0.22	0.11	0.17	**0.00**	**0.02**
11	5.170488e+001	407.82	0.37	0.60	0.06	0.22	**0.00**	**0.06**
13	4.917235e+001	440.11	0.06	0.86	0.13	0.35	**0.00**	**0.02**
15	4.696716e+001	426.49	0.78	1.17	0.09	0.36	**0.00**	**0.08**
20	4.287956e+001	347.86	0.74	1.96	0.32	0.59	**0.00**	**0.07**
Multiple features								
2	2.575342e+006	2161.69	0.01	1.27	**0.00**	0.55	**0.00**	**0.00**
3	2.386193e+006	2060.79	0.16	0.84	0.07	0.43	**0.00**	**0.00**
4	2.202546e+006	2058.69	**0.00**	1.49	**0.00**	0.11	**0.00**	**0.00**
6	1.996044e+006	1911.31	0.05	0.92	0.01	0.54	**0.00**	**0.01**
7	1.949002e+006	2004.90	0.03	0.40	0.02	0.12	**0.00**	**0.00**
10	1.750399e+006	1143.41	**0.00**	0.23	0.06	0.24	**0.00**	**0.00**
11	1.709637e+006	1454.06	0.02	0.61	0.07	0.29	**0.00**	**0.00**
13	1.639984e+006	1321.74	0.05	0.47	0.14	0.24	**0.00**	**0.04**
15	1.580220e+006	989.70	0.34	0.77	0.22	0.43	**0.00**	**0.08**
20	1.473096e+006	880.52	0.21	0.72	0.44	0.98	**0.00**	**0.01**
Image segmentation								
2	4.042200e+007	4528.08	0.67	3.60	0.23	2.47	**0.00**	**0.00**
3	3.109459e+007	4035.73	0.50	0.94	0.19	0.67	**0.00**	**0.00**
4	2.769080e+007	3736.06	0.53	1.86	0.25	1.06	**0.00**	**0.00**
6	2.248472e+007	2185.56	**0.00**	0.53	0.01	0.12	**0.00**	**0.00**
7	2.107812e+007	1805.85	0.39	0.98	0.01	0.65	**0.00**	**0.16**
10	1.892045e+007	1450.48	0.36	2.12	0.11	1.64	**0.00**	**0.26**
11	1.802198e+007	1330.74	**0.00**	0.74	0.05	0.43	**0.00**	**0.25**
13	1.658152e+007	1272.68	0.49	1.10	0.57	1.36	**0.00**	**0.44**
15	1.581005e+007	1450.09	**0.00**	1.50	0.11	0.27	**0.00**	**0.06**
20	1.423043e+007	1477.56	0.21	0.57	0.29	1.55	**0.00**	**0.03**

Table 6
Comparative statistical results between MA and the two state-of-the-art algorithms bk-means and VNS-LIMA. The best performances are marked in bold.

	bk-means	VNS-LIMA	MA
$\#Best/Avg$	30/14	47/21	**159/155**
Average $Best_{Dev}/Avg_{Dev}(\%)$	0.99/2.09	0.45/1.38	**0.14/0.49**
p-value$_{best}$/p-value$_{avg}$	7.47e-27/7.93e-33	7.74e-27/3.29e-30	

Table 7
Results of the Wilcoxon signed rank test for MA and the reference algorithms in terms of the best and average objective value, with a significance level of 0.05.

Comparison	R^{+}_{best}	R^{-}_{best}	p-value	R^{+}_{avg}	R^{-}_{avg}	p-value
MA versus bk-means	129	1	7.57e-22	146	2	5.90e-25
MA versus VNS-LIMA	113	1	3.70e-19	138	4	1.48e-23

average objective values is calculated by $(f - f^*)/f^* \times 100$, where f^* is the best solution value obtained by all the compared algorithms, and f is the best or the average objective value obtained with each algorithm. A smaller percentage deviation indicates that a better result is achieved in terms of the best or the average objective value.

Table 6 summarizes the results reported with the three compared approaches on all the 160 instances. In Table 6, row "$\#Best/Avg$" shows the number of cases for which an algorithm produces the best results among all the compared algorithms in terms of the best/average objective value. Row "Average $Best_{Dev}/Avg_{Dev}(\%)$" indicates the average percentage deviation of the best/average objective values from the best solution values over all the 160 instances. To verify whether there is a significant statistical difference in terms of the best/average objective values between MA and the two reference algorithms, the statistical results (p-values) from the non-parametric Friedman test are provided in row "p-value$_{best}$/p-value$_{avg}$". Table 6 illustrates that in terms of the best/average results ("$\#Best/Avg$"), MA yields the best results on 159/155 out of the 160 instances, while bk-means and VNS-LIMA produce the best results on 30/14 and 47/21 cases, respectively. With regard to the average percentage deviation of the best/average objective values from the best solution values over all the 160 instances ("Average $Best_{Dev}/Avg_{Dev}(\%)$"), MA reports the smallest deviation (0.14%/0.49%) from the best solution values obtained by all the compared algorithms. The non-parametric Friedman test (p-value ≤ 0.05) confirms the significant statistical difference between the results of MA and those of the compared algorithms. Table 7 shows the results of the Wilcoxon signed rank test [9] to detect any statistical significant performance difference. Column "R^{+}_{best}" ("R^{+}_{avg}") indicates the sum of ranks for the benchmark graphs, where MA performs better than the compared algorithm in terms of the best (average) objective value, while "R^-_{best}" ("R^-_{avg}") represents the sum of ranks for the opposite cases. MA shows a significant improvement over the two compared reference algorithms with a significance level of 0.05. This experiment discloses the effectiveness of the MA algorithm in solving these BMSSC instances.

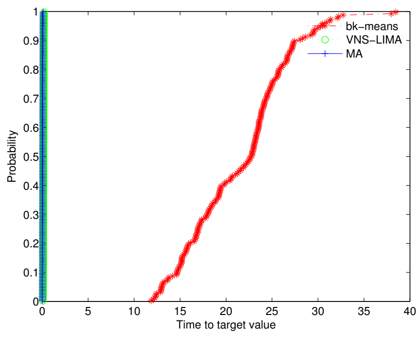
This work also tries to provide some implications from the computational and comparison results as managerial insights for the decision makers in certain domains, such as cluster analysis. First, the proposed MA method could be adapted to other cardinality-constrained MSSC clustering problems where the cardinality of each cluster is predefined but does not need to be the same [50]. Second, the solution cost significantly decreases with the increase in the number of clusters k, and the running time required to reach the obtained solution generally becomes less. Accordingly, in a studied clustering problem where the k value is unknown in advance, the decision makers can use as many clusters as possible (i.e., determine a k value as large as possible) within a given upper bound to reduce operating costs.

3.4 Time-to-target (TTT) and convergence analyses

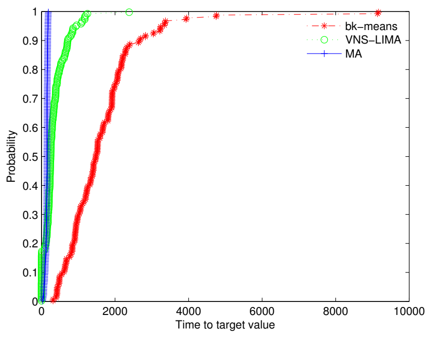
To obtain a deeper insight into the performance of the three algorithms, this section produced the TTT plots that are useful tools to compare stochastic algorithms by comparing their running time distributions [1]. A TTT plot for an algorithm is generated as follows [1]: Ex independent executions of the algorithm are performed for each given instance. In each execution, the running time required to achieve a given target value is recorded. The running times are sorted in an ascending order, and a probability $pb_l = (l - 0.5)/Ex$ is associated with the lth sorted running time t_l. Then, the point (t_l, pb_l) is plotted.

To produce the TTT plots, experiments were performed on eight instances randomly selected from the following datasets: Ionosphere, User knowledge, Water treatment plant, Breast cancer, Vehicle, Vowel recognition, Multiple features, and Image segmentation. Two hundred independent runs were executed for each instance. To enable all the compared approaches to arrive at the target in each run, the target value was set to be a value that was 0.5% larger than the best objective value found by bk-means.

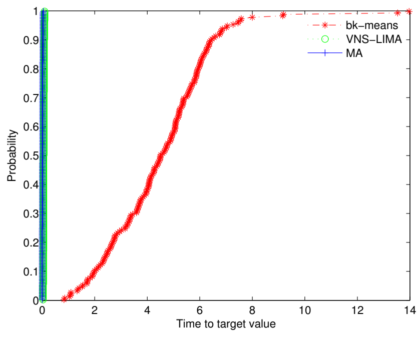
Fig. 2 presents the TTT plots for the compared algorithms on the eight selected instances. Fig. 2 demonstrates that MA outperforms all the reference algorithms on the six instances from the datasets Image segmentation, Multiple features, User knowledge, Vehicle, Vowel recognition, and Water treatment plant by obtaining the given target value with the shortest running time. For the remaining two instances from the datasets Breast cancer and



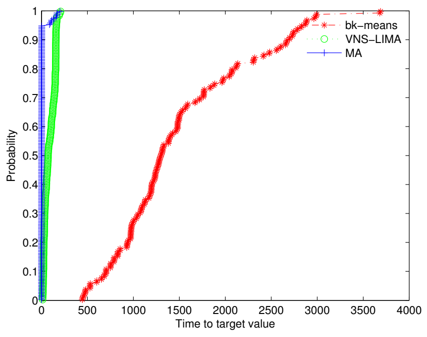
(a) Breast cancer ($k = 2$)



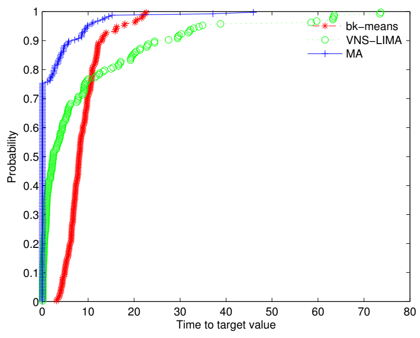
(b) Image segmentation ($k = 7$)



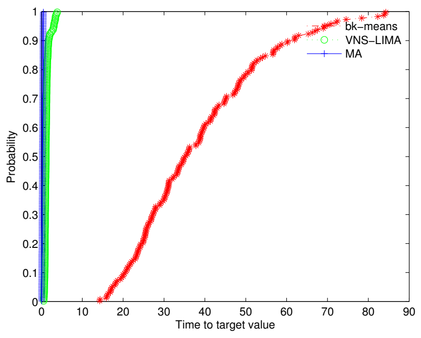
(c) Ionosphere ($k = 2$)



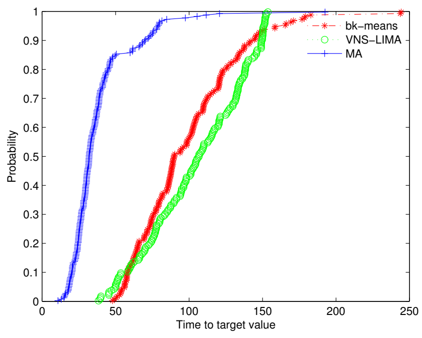
(d) Multiple features ($k = 7$)



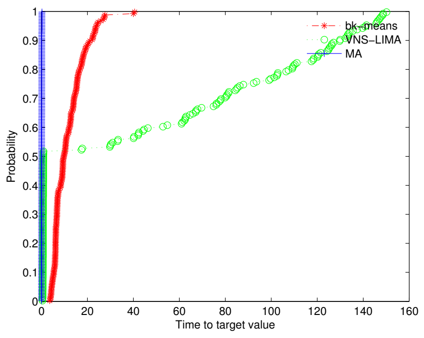
(e) User knowledge ($k = 4$)



(f) Vehicle ($k = 6$)

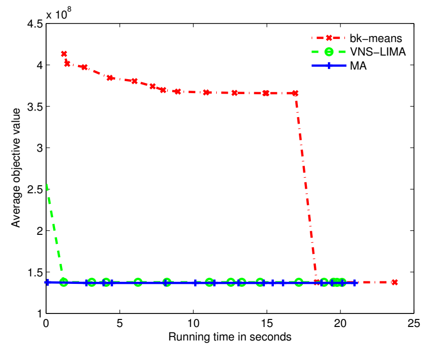


(g) Vowel recognition ($k = 11$)

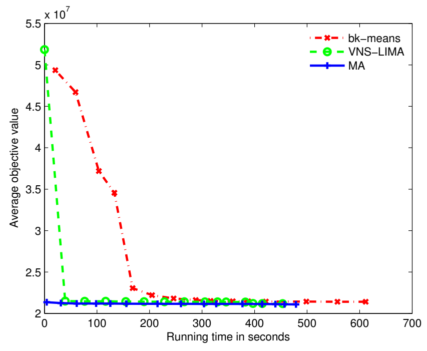


(h) Water treatment plant ($k = 13$)

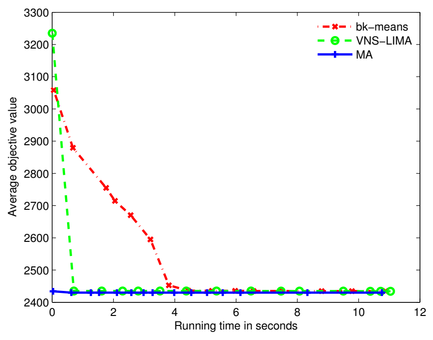
Fig. 2. Probability distribution for the time (in seconds) to obtain a given target value.



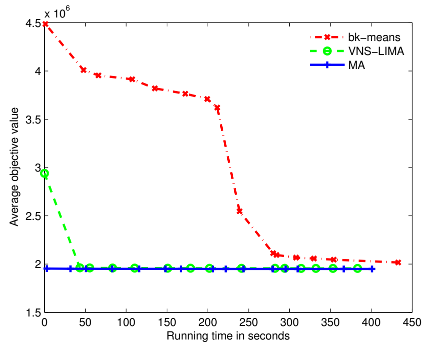
(a) Breast cancer ($k = 2$)



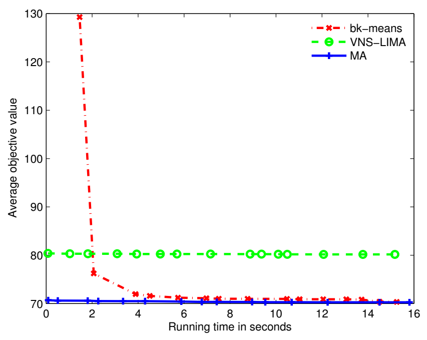
(b) Image segmentation ($k = 7$)



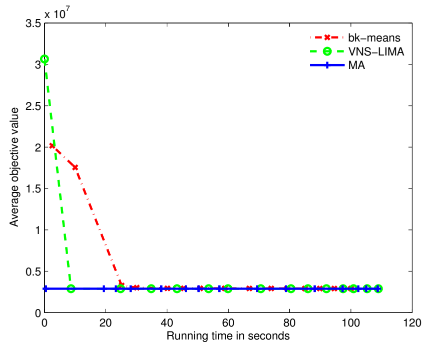
(c) Ionosphere ($k = 2$)



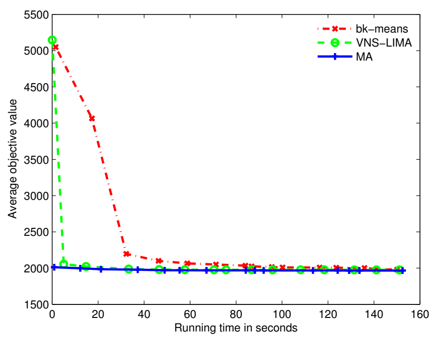
(d) Multiple features ($k = 7$)



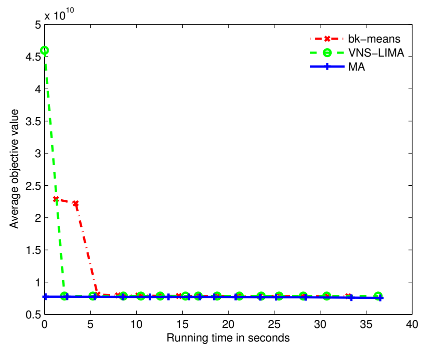
(e) User knowledge ($k = 4$)



(f) Vehicle ($k = 6$)



(g) Vowel recognition ($k = 11$)



(h) Water treatment plant ($k = 13$)

Fig. 3. Convergence profiles of MA and the two reference algorithms bk-means and VNS-LIMA.

Ionosphere, MA similarly performs compared with VNS-LIMIA and dominates bk-means. Fig. 3 shows the convergence profiles of MA and the two reference algorithms on the eight selected graphs to illustrate the evolution of the average objective value across 10 independent runs during the search process. MA requires less time to obtain even better solutions than bk-means and VNS-LIMA for all the tested graphs. Similar results are observed for the remaining instances. This experiment further confirms the highly competitive performance of the proposed MA approach with respect to the two reference algorithms.

4 Analysis

In this section, some key ingredients of the proposed algorithm are analyzed to understand their effects on the algorithm's performance, including the RTS and the backbone-based crossover operator.

4.1 Efficiency of the RTS

To evaluate the efficiency of the RTS, it is compared with the state-of-the-art algorithms bk-means and VNS-LIMA. For this comparison, the same three datasets (Wine, Breast cancer, and Vehicle) as in Section 3.2 are used. All the compared algorithms were performed under the same experimental condition as shown in Section 3.1 and executed 10 independent runs for each instance. Table 8 reports the comparative results of the compared algorithms. Row "*#Best*" summarizes the number of instances for which the corresponding algorithm produces the best solutions among the compared algorithms. Row "*Avg*" reports the average percentage deviation from the best solution values obtained by all the compared algorithms within this experiment. The p-values from the non-parametric Friedman tests are reported in the last row of the table.

Table 8 illustrates that for all the 30 instances, RTS produces the best results on 30/30 instances in terms of the best/average objective values, while bk-means and VNS-LIMA yield the best results on 12/3 and 16/4 cases, respectively. In terms of the average percentage deviation ("*Avg*") from the best solution values obtained by all the compared algorithms, RTS shows the smallest deviation (0.00%/0.11%). The p-values (≤ 0.05) from the non-parametric Friedman test in terms of the best and average objective values reveal a significant statistical difference between the RTS and the two compared algorithms. The results of Wilcoxon signed rank test shown in Table 9 also confirm the superiority of RTS. This experiment demonstrates that RTS

Table 8
Comparative results between RTS and two state-of-the-art algorithms bk-means and VNS-LIMA in terms of percentage deviation from the best solutions obtained by all the compared algorithms. The best results are indicated in bold.

		bk-means		VNS-LIMA		RTS	
k	*cost*	$Best_{Dev}$	Avg_{Dev}	$Best_{Dev}$	Avg_{Dev}	$Best_{Dev}$	Avg_{Dev}
Wine							
2	6.507529e+006	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
3	2.962226e+006	**0.00**	0.42	**0.00**	0.62	**0.00**	**0.00**
4	1.904950e+006	**0.00**	0.18	**0.00**	0.38	**0.00**	**0.00**
6	1.010016e+006	0.24	2.01	0.24	1.85	**0.00**	**0.27**
7	7.352635e+005	2.25	3.12	0.39	3.90	**0.00**	**0.13**
10	5.074534e+005	1.08	1.63	0.33	1.31	**0.00**	**0.10**
11	4.327903e+005	0.33	0.87	**0.00**	2.76	**0.00**	**0.00**
13	3.612952e+005	3.05	6.06	2.28	6.79	**0.00**	**0.24**
15	2.771871e+005	7.09	9.35	0.64	6.74	**0.00**	**0.07**
20	1.737925e+005	0.43	6.27	0.53	12.21	**0.00**	**1.46**
Breast cancer							
2	1.375244e+008	**0.00**	0.17	**0.00**	0.09	**0.00**	**0.00**
3	8.743161e+007	0.64	0.64	**0.00**	0.45	**0.00**	**0.00**
4	5.978607e+007	**0.00**	0.17	**0.00**	0.24	**0.00**	**0.00**
6	3.973995e+007	**0.00**	0.40	**0.00**	0.56	**0.00**	**0.00**
7	3.468144e+007	0.03	0.33	0.01	0.32	**0.00**	**0.00**
10	2.593484e+007	**0.00**	0.84	**0.00**	0.70	**0.00**	**0.00**
11	2.378349e+007	1.53	1.68	0.12	1.14	**0.00**	**0.00**
13	2.060493e+007	1.43	1.44	**0.00**	1.14	**0.00**	**0.48**
15	1.858171e+007	1.55	1.62	1.40	1.54	**0.00**	**0.02**
20	1.455947e+007	0.13	1.10	0.10	1.29	**0.00**	**0.02**
Vehicle							
2	1.234444e+007	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
3	5.565073e+006	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
4	4.322273e+006	**0.00**	0.10	**0.00**	0.12	**0.00**	**0.00**
6	2.894321e+006	**0.00**	0.24	**0.00**	**0.00**	**0.00**	**0.00**
7	2.560683e+006	**0.00**	0.17	**0.00**	0.11	**0.00**	**0.00**
10	1.995501e+006	0.11	0.33	0.04	0.25	**0.00**	**0.00**
11	1.884479e+006	0.03	0.20	**0.00**	0.18	**0.00**	**0.00**
13	1.715241e+006	0.03	0.25	0.01	0.09	**0.00**	**0.04**
15	1.542432e+006	0.22	0.81	0.15	0.47	**0.00**	**0.26**
20	1.255876e+006	0.77	1.36	0.09	0.54	**0.00**	**0.20**
#Best		12	3	16	4	**30**	**30**
Avg		0.69	1.39	0.21	1.53	**0.00**	**0.11**
p-value		2.61e-4	2.07e-6	1.08e-4	2.03e-7		

Table 9
Results of the Wilcoxon signed rank test for RTS and the compared algorithms for both the best and average objective values, with a significance level of 0.05.

Comparison	R^{+}_{best}	R^{-}_{best}	*p*-value	R^{+}_{avg}	R^{-}_{avg}	*p*-value
RTS versus bk-means	18	0	1.95e-4	27	0	5.58e-6
RTS versus VNS-LIMA	14	0	9.79e-4	26	0	8.30e-6

is competitive with the state-of-the-art algorithms on these BMSSC instances.

Table 10
Comparative results between MA and the multi-start version of RTS in terms of percentage deviation from the best solutions obtained by all the compared algorithms. The best results are indicated in bold.

		RTS		MA	
k	*cost*	$Best_{Dev}$	Avg_{Dev}	$Best_{Dev}$	Avg_{Dev}
Wine					
2	6.507529e+006	**0.00**	**0.00**	**0.00**	**0.00**
3	2.962226e+006	**0.00**	**0.00**	**0.00**	**0.00**
4	1.904950e+006	**0.00**	**0.00**	**0.00**	**0.00**
6	1.008776e+006	0.12	0.40	**0.00**	**0.00**
7	7.345635e+005	0.10	0.22	**0.00**	**0.00**
10	5.061534e+005	0.26	0.35	**0.00**	**0.06**
11	4.327903e+005	**0.00**	**0.00**	**0.00**	**0.00**
13	3.601952e+005	0.31	0.55	**0.00**	**0.01**
15	2.764871e+005	0.25	0.33	**0.00**	**0.00**
20	1.737925e+005	**0.00**	1.46	**0.00**	**0.03**
Breast cancer					
2	1.366899e+008	0.61	0.61	**0.00**	**0.49**
3	8.743161e+007	**0.00**	**0.00**	**0.00**	**0.00**
4	5.978607e+007	**0.00**	**0.00**	**0.00**	**0.00**
6	3.973995e+007	**0.00**	**0.00**	**0.00**	**0.00**
7	3.468144e+007	**0.00**	**0.00**	**0.00**	0.01
10	2.593484e+007	**0.00**	**0.00**	**0.00**	**0.00**
11	2.378349e+007	**0.00**	**0.00**	**0.00**	**0.00**
13	2.060293e+007	0.01	0.49	**0.00**	**0.01**
15	1.858171e+007	**0.00**	0.02	**0.00**	**0.01**
20	1.455947e+007	**0.00**	0.02	**0.00**	**0.01**
Vehicle					
2	1.234444e+007	**0.00**	**0.00**	**0.00**	**0.00**
3	5.565073e+006	**0.00**	**0.00**	**0.00**	**0.00**
4	4.307852e+006	0.33	0.33	**0.00**	**0.18**
6	2.894321e+006	**0.00**	**0.00**	**0.00**	**0.00**
7	2.547200e+006	0.53	0.53	**0.00**	**0.42**
10	1.982152e+006	0.67	0.67	**0.00**	**0.50**
11	1.884479e+006	**0.00**	**0.00**	**0.00**	**0.00**
13	1.715241e+006	**0.00**	0.04	**0.00**	**0.01**
15	1.542432e+006	**0.00**	0.26	**0.00**	**0.07**
20	1.255201e+006	0.05	0.25	**0.00**	**0.19**
#Best		19	14	**30**	**29**
Avg		0.11	0.22	**0.00**	**0.07**
p-value		3.89e-3	9.67e-4		

Table 11
Wilcoxon signed rank test results between MA and RTS in terms of the best and average objective values, with a level of significance of 0.05.

Comparison	R^{+}_{best}	R^{-}_{best}	p-value	R^{+}_{avg}	R^{-}_{avg}	p-value
MA versus RTS	11	0	3.35e-3	16	1	4.18e-4

4.2 Effect of the backbone-based crossover operator

As shown in Section 2, the proposed MA approach adopts a dedicated backbone-based crossover that transfers common points grouped together from parents to offspring. Additional experiments were performed to evaluate the benefit of the backbone-based crossover. For this purpose, the approach was compared with its underlying local search method, i.e., the RTS algorithm. Experiments were executed on the same selection of the three datasets as before. To solve every instance, each algorithm was ran 10 times under the same experimental protocol as described in Section 3.1. To be fair, RTS was run in a multi-restart way until the time limit was reached.

Tables 10 and 11 show the results reported by the two algorithms. MA and RTS produce the best results on 30/29 and 19/14 out of the 30 instances, respectively, in terms of the best/average objective values. Moreover, MA shows a smaller average percentage deviation ("*Avg*") from the best solution values obtained by the two compared algorithms (0.00%/0.07%) in terms of the best and the average objective values. The p-values ($\leq$ 0.05) from the non-parametric Friedman test confirm the statistical significance of the comparison between RTS and MA in terms of the best/average objective values. Meanwhile the Wilcoxon signed rank test results reveal that MA outperforms RTS with a level of significance 0.05. This experiment further confirms that the backbone-based crossover operator plays an important role to the performance of the MA algorithm.

5 Conclusions

This work proposes a population-based MA for the BMSSC problem that is obtained by imposing the balance property to the classic MSSC. The proposed algorithm hybridizes a backbone-based crossover operator with an effective local optimization procedure based on RTS. The specific crossover operator generates promising offspring solutions by inheriting good properties from parent solutions, while RTS relies on a combined use of two neighborhoods and two complementary search strategies to explore high-quality candidate solutions.

Extensive computational experiments on 16 datasets of 160 instances show that MA outperforms the state-of-the-art BMSSC algorithms, including bk-means and VNS-LIMA, in terms of the best and average solution values. Specifically, MA yields the best results on 159/155 out of the 160 instances in terms of the best/average objective values, and such a performance has never been reported in the BMSSC literature by any other algorithm. These results also largely dominate the those obtained by two state-of-the-art algorithms (bk-means and VNS-LIMA), which is confirmed by the Wilcoxon signed rank test. Moreover, the TTT analysis further confirms the competitiveness of MA with respect to the reference algorithms in terms of the computational efficiency. Additional experiments are also performed to analyze the influences of the key parameters and ingredients on the performance of the algorithm.

Given the excellent results achieved by the proposed algorithm, the algorithm can be considered to be an advanced tool for the general BMSSC model. Given that BMSSC can conveniently formulate a number of real-world applications, the proposed algorithm can be employed by researchers and practitioners to tackle these practical problems. Moreover, the distribution of our code to the public domain will further facilitate such applications.

Considering that the proposed algorithm is a heuristic approach, the gap between the reported solutions and the optimal solutions cannot be determined. Additional research work on the exact and approximate methods that can guarantee the solution quality is thus needed. Furthermore, the RTS strategy can be improved by investigating other thresholding techniques, such as tabu thresholding [15]. Finally, the MA algorithm examines only feasible solutions satisfying the balance constraint on the cardinality of the clusters. Meanwhile, relaxing some constraints for constrained problems during the search can be beneficial [16,44,48]. Search strategies that thus explore feasible and infeasible solutions must be investigated.

Acknowledgments

We are grateful to the reviewers of this paper for their comments and suggestions. We would like to thank the authors of [12] and [34] for sharing the codes of the VNS-LIMA and bk-means algorithms. This work is partially supported by the National Natural Science Foundation of China (Grant nos. 71771099, 71821001, 71810107003, and 71620107002).

References

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

[2] Asgari, N., Rajabi, M., Jamshidi, M., Khatami, M., & Farahani, R. Z. (2017). A memetic algorithm for a multi-objective obnoxious waste location-routing problem: a case study. Annals of Operations Research, 250(2), 279-308.

[3] Bagirov, A. M., Taheri, S., & Ugon, J. (2016). Nonsmooth DC programming approach to the minimum sum-of-squares clustering problems. Pattern Recognition, 53, 12-24.

[4] Bagirov, A. M., & Yearwood, J. (2006). A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems. European Journal of Operational Research, 170(2), 578-596.

[5] Bai, L., Liang, J., Sui, C., & Dang, C. (2013). Fast global k-means clustering based on local geometrical information. Information Sciences, 245, 168-180.

[6] Benlic, U., & Hao, J. K. (2011). A multilevel memetic approach for improving graph K-partitions. IEEE Transactions on Evolutionary Computation, 15(5), 624-642.

[7] Bertoni, A., Goldwurm, M., Lin, J., & Saccà, F. (2012). Size constrained distance clustering: separation properties and some complexity results. Fundamenta Informaticae, 115(1), 125-139.

[8] Brucker, P. (1978). On the complexity of clustering problems. In: R. Henn, B. Korte, & W. Oettli (Eds.), Optimization and Operations Research, Lecture Notes in Economics and Mathematical Systems, vol 157 (pp. 45-54). Berlin: Springer.

[9] Carrasco, J., García, S., Rueda, M. M., Das, S., & Herrera, F. (2020). Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review. Swarm and Evolutionary Computation, 54, 100665.

[10] Chen Y. & Hao, J. K. (2015). Iterated responsive threshold search for the quadratic multiple knapsack problem. Annals of Operations Research, 226(1): 101–131.

[11] Chen, Y., He, F., Li, H., Zhang, D., & Wu, Y. (2020). A full migration BBO algorithm with enhanced population quality bounds for multimodal biomedical image registration. Applied Soft Computing, Vol 93, Article 106335.

[12] Costa, L. R., Aloise, D., & Mladenović, N. (2017). Less is more: basic variable neighborhood search heuristic for balanced minimum sum-of-squares clustering. Information Sciences, 415-416, 247-253.

[13] Desrosiers, J., Mladenović, N., & Villeneuve, D. (2005). Design of balanced MBA student teams. Journal of the Operational Research Society, 56(1), 60-66.

[14] Edwards, A. W. F., & Cavalli-Sforza, L. L. (1965). A method for cluster analysis. Biometrics, 21(2), 362-375.

[15] Glover, F. (1995). Tabu thresholding: improved search by nonmonotonic trajectories. ORSA Journal on Computing, 7(4), 426-442.

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

[17] Gribel, D., & Vidal, T. (2019). HG-means: A scalable hybrid genetic algorithm for minimum sum-of-squares clustering. Pattern Recognition, 88, 569-583.

[18] Hamby, D. M. (1994). A review of techniques for parameter sensitivity analysis of environmental models. Environmental Monitoring and Assessment, 32(2), 135-154.

[19] Hansen, P., & Mladenović, N. (2001). J-means: a new local search heuristic for minimum sum of squares clustering. Pattern Recognition, 34(2), 405-413.

[20] Hao, J. K. (2012). Memetic algorithms in discrete optimization. In: F. Neri, C. Cotta, & P. Moscato (Eds.), Handbook of Memetic Algorithms, Studies in Computational Intelligence, Vol 379 (pp. 73-94). Berlin: Springer.

[21] Hasani, A., & Khosrojerdi, A. (2016). Robust global supply chain network design under disruption and uncertainty considering resilience strategies: A parallel memetic algorithm for a real-life case study. Transportation Research Part E: Logistics and Transportation Review, 87, 20-52.

[22] Hou, N., He, F., Zhou, Y., &Chen, Y. (2020). An Efficient GPU-based parallel tabu search algorithm for hardware/software co-design. Frontiers of Computer Science, 14(5), 1-18.

[23] Jain, A. K. (2010). Data clustering: 50 years beyond K-means. Pattern Recognition Letters, 31(8), 651-666.

[24] Jin, Y., & Hao, J. K. (2016). Hybrid evolutionary search for the minimum sum coloring problem of graphs. Information Sciences, 352-353, 15-34.

[25] Karmitsa, N., Bagirov, A. M., & Taheri, S. (2017). New diagonal bundle method for clustering problems in large data sets. European Journal of Operational Research, 263(2), 367-379.

[26] Kuhn, H. W. (2005). The Hungarian method for the assignment problem. Naval Research Logistics, 52(1), 7-21.

[27] Laszlo, M., & Mukherjee, S. (2007). A genetic algorithm that exchanges neighboring centers for k-means clustering. Pattern Recognition Letters, 28(16), 2359-2366.

[28] Liang, J., Xue, Y., & Wang, J. (2020). Bi-objective memetic GP with dispersion-keeping Pareto evaluation for real-world regression. Information Sciences, 539, 16-35.

[29] Li, H., He, F., Liang, Y., & Quan, Q. (2019). A dividing-based many-objective evolutionary algorithm for large-scale feature selection. Soft Computing, 24(9), 6851-6870.

[30] Likas, A., Vlassis, N., & Verbeek, J. J. (2003). The global k-means clustering algorithm. Pattern Recognition, 36(2), 451-461.

[31] Liu, Y., Yi, Z., Wu, H., Ye, M., & Chen, K. (2008). A tabu search approach for the minimum sum-of-squares clustering problem. Information Sciences, 178(12), 2680-2704.

[32] Lü, Z., Hao, J. K., & Glover, F. (2011). Neighborhood analysis: a case study on curriculum-based course timetabling. Journal of Heuristics, 17(2), 97-118.

[33] Lu, Y., Hao, J. K., & Wu, Q. (2019). Hybrid evolutionary search for the traveling repairman problem with profits. Information Sciences, 502, 91-108.

[34] Malinen, M. I., & Fränti, P. (2014, August). Balanced K-means for clustering. In: P. Fränti, G. Brown, M. Loog, F. Escolano, & M. Pelillo (Eds.), Structural, Syntactic and Statistical Pattern Recognition. S+SSPR 2014. Lecture Notes in Computer Science, vol 8621 (pp. 32-41). Berlin: Springer.

[35] Mirsaleh, M. R., & Meybodi, M. R. (2016). A Michigan memetic algorithm for solving the community detection problem in complex network. Neurocomputing, 214, 535-545.

[36] Montgomery, D. C. (2017). Design and analysis of experiments. John Wiley & Sons.

[37] Neri, F., & Cotta, C. (2012). Memetic algorithms and memetic computing optimization: A literature review. Swarm and Evolutionary Computation, 2, 1-14.

[38] Pereira, T., Aloise, D., Brimberg, J., & Mladenović, N. (2018). Review of basic local searches for solving the Minimum Sum-of-Squares Clustering Problem. In: P. Pardalos, & A. Migdalas (Eds.), Open Problems in Optimization and Data Analysis, Vol 141 (pp. 249-270). Cham: Springer.

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

[40] Qu, B. Y., Zhu, Y. S., Jiao, Y. C., Wu, M. Y., Suganthan, P. N., & Liang, J. J. (2018). A survey on multi-objective evolutionary algorithms for the solution of the environmental/economic dispatch problems. Swarm and Evolutionary Computation, 38, 1-11.

[41] Rajasekhar, A., Lynn, N., Das, S., & Suganthan, P. N. (2017). Computing with the collective intelligence of honey bees–a survey. Swarm and Evolutionary Computation, 32, 25-48.

[42] Shen, X. N., Minku, L. L., Marturi, N., Guo, Y. N., & Han, Y. (2018). A Q-learning-based memetic algorithm for multi-objective dynamic software project scheduling. Information Sciences, 428, 1-29.

[43] Su, W., Hu, J., Lin, C., & Shen, S. (2015, June). SLA-aware tenant placement and dynamic resource provision in SaaS. In 2015 IEEE International Conference on Web Services (pp. 615-622). New York: IEEE.

[44] Sun, W., Hao, J. K., Lai, X., & Wu, Q. (2018). Adaptive feasible and infeasible tabu search for weighted vertex coloring. Information Sciences, 466, 203-219.

[45] Yadegari, E., Alem-Tabriz, A., & Zandieh, M. (2019). A memetic algorithm with a novel neighborhood search and modified solution representation for closed-loop supply chain network design. Computers & Industrial Engineering, 128, 418-436.

[46] Yong, J., He, F., Li, H., & Zhou, W. (2019). A Novel Bat Algorithm based on Cross Boundary Learning and Uniform Explosion Strategy. Applied Mathematics-A Journal of Chinese Universities, 34(4), 480-502.

[47] Zhao, S. Z., Suganthan, P. N., & Das, S. (2010). Dynamic multi-swarm particle swarm optimizer with sub-regional harmony search. In IEEE Congress on Evolutionary Computation (pp. 1-8). Barcelona: IEEE.

[48] Zhou, Q., Benlic, U., Wu, Q., & Hao, J. K. (2019). Heuristic search to the capacitated clustering problem. European Journal of Operational Research, 273(2), 464-487.

[49] Zhou, Q., Benlic, U., & Wu, Q. (2020). A memetic algorithm based on reformulation local search for minimum sum-of-squares clustering in networks. Information Sciences, 541, 271-296.

[50] Zhu, S., Wang, D., & Li, T. (2010). Data clustering with size constraints. Knowledge-Based Systems, 23(8), 883-889.