An Adaptive and Opposite *K*-means Operation based Memetic Algorithm for Data Clustering

Xi Wang[a], Zidong Wang[b], Mengmeng Sheng[c,d], Qi Li[a], Weiguo Sheng[a,1]

[a]*Department of Computer Science, Hangzhou Normal University, Hangzhou 311121, PR China*
[b]*Department of Computer Science, Brunel University, Uxbridge, Middlesex, UB8 3PH, UK*
[c]*School of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, PR China*
[d]*Zhejiang Police College, 310053, PR China*

Abstract

Evolutionary algorithm (EA) incorporating with *k*-means local search operator represents an important approach for cluster analysis. In the existing EA approach, however, the *k*-means operators are usually directly employed on the individuals and generally applied with fixed intensity as well as frequency during evolution, which could significantly limit their performance. In this paper, we first introduce a hybrid EA based clustering framework such that the frequency and intensity of *k*-means operator could be arbitrarily configured during evolution. Then, an adaptive strategy is devised to dynamically set its frequency and intensity according to the feedback of evolution. Further, we develop an opposite search strategy to implement the proposed adaptive *k*-means operation, thus appropriately exploring the search space. By incorporating the above two strategies, a memetic algorithm with adaptive and opposite *k*-means operation is finally proposed for data clustering. The performance of the proposed method has been evaluated on a series of data sets and compared with relevant algorithms. Experimental results indicate that our proposed algorithm is generally able to deliver superior performance and outperform related methods.

Keywords: Data clustering, memetic algorithm, adaptive local search, opposite local search, *k*-means

[1] Corresponding author: Weiguo Sheng
Email address: w.sheng@ieee.org (Weiguo Sheng)

1. Introduction

Data clustering, as an important branch of data mining, has been deemed as a critical task in unsupervised learning. For a given data set, clustering attempts to divide it into groups/clusters such that data objects in the same group/cluster are similar with each other while dissimilar with the data objects from other groups/clusters. A number of methods have been developed for clustering, which can be generally categorized as partitional approach and hierarchical approach [1]. For partitional methods, they can be classified as hard approach, in which each object is assigned to one cluster only, and fuzzy approach [2], in which each object belongs every cluster with a certain membership. In this work, we focus on the hard partitional approach.

To implement the hard partitional clustering, the task is generally achieved via optimizing certain clustering functions [1, 3, 4]. Among the various clustering functions proposed in literature [5], the SSE (i.e., sum of squared errors) has been widely employed. However, optimizing SSE to deliver appropriate clusters is known to be a *NP*-hard problem [6]. Traditional methods [7-10] typically employ alternating optimization schemes (such as *k*-means algorithm) to optimize SSE, which could be easily trapped into local optima.

Evolutionary algorithms (EAs), which aim to identify optimal or near-optimal solutions of optimization problems, have been widely employed to optimize SSE for obtaining appropriate clusters [11, 12]. Typically, it could be time-consuming to employ traditional EAs for clustering a dataset with non-trivial size. To improve the time efficiency, hybrid EAs, also called memetic algorithms (MAs), whose idea is to incorporate *k*-means based local search procedure into traditional EAs for clustering, have been proposed in literature [13-16]. These works indicate that the search efficiency of traditional EA-based data clustering could be greatly improved by incorporating them with *k*-means operation. This is because traditional EA-based clustering methods are not good at fine-tuning the solutions while the *k*-means based procedure can significantly enhance this capability. Despite its success in improving the performance of EA-based clustering [25], [33], [36], the *k*-means operation in existing methods is usually employed with fixed intensity (i.e., a fixed number of iterations of the algorithm) and fixed frequency (i.e., applied at each generation) during evolution [30], [33], [34]. As the evolutionary search of EA is a dynamical procedure, this could significantly reduce the performance of existing EA-based clustering methods. Moreover, the *k*-means operation in these methods is generally applied directly on the individual without considering the global information of population, which therefore renders the performance further.

This work intends to address the issue of appropriately employing the *k*-means local search procedure in EA-based clustering. To this end, a generalized framework, which supports the arbitrary setting of the intensity as well as frequency of *k*-means operator during evolution, is first introduced. Specifically, it is realized by employing two parameters to control the *k*-means operation's intensity as well as frequency in the evolutionary process. Then, an adaptive strategy is devised to set these two parameters, which could dynamically control the *k*-means operation's intensity and frequency during evolution. Further, an opposite search mechanism has been proposed to implement the adaptive *k*-means operation. The main idea of this mechanism is to sample points along the opposite direction of individuals attracted and replace them with the sampling points with better fitness, thus appropriately exploring the search space. By integrating the above two strategies, an adaptive and opposite *k*-means operation based memetic algorithm is finally presented for data clustering. In experiments, we evaluate the proposed algorithm by using both simulated and real data sets. The results clearly show the significance of the adaptive *k*-means operation and opposite search strategies for evolutionary data clustering. Experimental results also reveal that our proposed method can outperform related methods implemented for comparison.

It should be mentioned that this work represents a significant extension of our previous conference publication [17] that briefly describes an adaptive *k*-means operation based MA for data clustering and its initial results. This paper extends the previous work by, firstly, providing a detailed description of the devised adaptive *k*-means operation strategy along with extensive experimental studies to analyze its effectiveness. Moreover, an opposite search mechanism used to implement the adaptive *k*-means operation is also designed and incorporated into the proposed method in [17]. Therefore, the major contribution of the current work is to develop an adaptive and opposite *k*-means operation based MA for effective and efficient data clustering.

2. Related Work

2.1. Hybrid EA-based partitional clustering

EAs have been widely employed for partitional data clustering [18-24], which is known to be *NP*-hard. These methods could require extensive computational time to achieve convergence for data sets with non-trivial sizes. To improve the time efficiency, hybrid EAs, which incorporate the *k*-means operator into EAs for fine-tuning the solutions, have also been developed and widely used for data clustering [13-15, 25-37]. For example, in [25], Xiao *et al.* applied one iteration of the *k*-means algorithm on each individual to improve initial solutions in the population before the evolutionary search. Beg *et al.* [26-28] tried to generate a high-quality initial population by running *k*-means on each individual until convergence. In [29], the best solution identified by the GA is fine-tuned by *k*-means to obtain a final solution. Apart from improving the initial or final solutions using the *k*-means operation, it has typically been used to fine-tune solutions during the evolutionary clustering. In [13] and [15], one iteration of *k*-means is used as a replacement of the crossover operation in EA for improving the offspring solutions during evolutionary clustering. Badyopadhyay *et al.* [30], Kivijarvi *et al.* [14], Sheng *et al.* [31, 32], He *et al.* [34] and Laszlo *et al.* [33] applied one or two iterations of the *k*-means algorithm to fine-tune the offspring generated by genetic operator. While, in [35-37], the *k*-means is designed to run on each offspring until converge during evolution. Among various alternatives, hybrid EAs, which apply the *k*-means operator to fine-tune solutions during evolution, has appeared to be the most successful one. However, in existing work, the *k*-means operation is usually employed with fixed intensity (i.e., a fixed number of iterations) and fixed frequency (i.e., applied at each generation). Since the evolution is a dynamic procedure, the performance of these methods could be significantly limited by employing the *k*-means operator in a fixed manner. Moreover, in these methods, the *k*-means based operator is usually employed directly to improve the individual and ignores the global information of population. This could reduce the performance further.

2.2. MA and adaptive local search

The hybrid evolutionary clustering algorithms mentioned above are closely related to memetic computation (MC), which is defined as a paradigm that combines population-based heuristics with individual learning or local search procedures for the sake of problem solving [38]. In the literature concerned with MC, an individual learning or local search procedure, which is usually referred to as a meme, has been used to enhance the capability of population-based heuristics. The integration of memes has been established as an extension of canonical EAs, by the name of MA. In this work, since the *k*-means operation is a domain-specific local search and plays a significant role in the proposed algorithm, the term MA is adopted.

In the literature on MA, it has been widely recognized that appropriate usage the local search (LS) is crucial for the success of MAs [42]. To tackle this issue, many efforts have been carried out aiming at controlling the LS's intensity and/or frequency during evolution. The resulting methods have been applied to address problems in context other than the clustering that we considered here. For example, in [39], Hart *et al.* proposed two schemes, i.e., fitness-based scheme and distribution-based scheme, to adaptively apply the LS to individuals, where the fitness-based scheme tried to bias the LS towards solutions in the population with higher fitness, while the distribution-based scheme tended to avoid unnecessary LS operation based on redundancy information in the population. In [40], Lozano *et al.* suggested a simple method to apply a LS such that the offspring are subject to the LS if their fitness are better than the worst individual in the population, otherwise a fixed probability of 0.0625 would be applied. In [41], Molina *et al.* divided individuals of the population into three groups (i.e., most promising, median and less promising) and assigned different probabilities and intensities of LS to the individuals in different groups. Specifically, for offspring whose fitness are better than the best fitness in current population (i.e., most promising group), a maximum probability and intensity of LS would be assigned. For the individuals in median group (i.e., the offspring whose fitness are between the worst and best one in the population), they are subject to a low probability and median intensity of LS. While, for the individuals in less promising group, no LS would be employed. In [42], Nguyen *et al.* derived a theoretical bound for the intensity of LS and developed a probabilistic memetic framework to govern the intensity of LS. In this work, a theoretical upper bound and expected intensity of LS to be applied are estimated based on the distribution of individuals. In [43], Nobahari *et al.* proposed to employ the concentration information of high-quality solutions in the population to adapt the intensity of LS as well as the ratio of local-global search. In this method, a population with a more concentration of high-quality solutions would be assigned with a higher intensity of LS and lower ratio of local-global search, and vice versa. In [44], Noman *et al.* devised a Lamarckian LS that adaptively determines the intensity of LS by taking into account of feedback (i.e., fitness improvement) from the evolution. In [45], Liu *et al.* presented an adaptive strategy, which adjusts the intensity of LS based on the performance of global search and LS during evolutionary search. The performance of global search and LS is measured in terms of average fitness improvement (AFI) dynamically calculated at each generation. The method increases the intensity of LS at next generation when the AFI of global search is outperformed by that of LS, otherwise decreases it. In [46], Bambha *et al.* devised a method to control the intensity of parameterized local search algorithms (PLSA) used in EAs via a simulated heating framework. This method tries to gradually allocate more time to each PLSA along with the advance of evolution. In [47], Ma *et al.* proposed an adaptive strategy to adaptively determine the capability level of Gaussian local search [48] for different individuals in the population according to a probability model. The model is calculated based on the information of individuals' fitness increment in the process of local search. While, in [49], Shahidi *et al.* encoded the parameters (i.e., step size and memory term) of conjugate gradient method into the chromosome, thereby dynamically controlling the intensity and direction of LS.

2.3 Motivations

In this work, we proposed a memetic algorithm with an adaptive and opposite *k*-means operation for data clustering. The motivations of the adaptive *k*-means operation and opposite search mechanism in the proposed method are as follows.

Motivation behind the adaptive k-means operation: In the MA based clustering methods, the *k*-means operators are usually employed with a fixed intensity and frequency. It has now been established that the intensity/frequency of local search in a MA should be appropriately controlled in order to have a good performance. This is due to an excessive intensity and/or frequency of local search could easily lead the population to premature convergence without properly exploring the search space. While, an insufficient intensity and/or frequency of local search could hinder the efficiency of evolutionary search. Moreover, the appropriate intensity/frequency depends on the problem instances at hand as well as the stages of EA evolution. It is therefore desirable to enhance the performance of MA based clustering by adaptively adjusting the intensity as well as frequency of k-means operator during evolution.

Motivation behind the opposite search: In MA literature, the local search is generally designed to apply directly on the individual without considering the exploration aspect of population. In this case, although the local search is able to greatly improve the algorithm's efficiency, it could also significantly accelerate the speed of premature convergence, for which the traditional EAs are generally suffered from. To alleviate such an issue, a promising option is perhaps to consider the exploration aspect of the population to perform the local search, thus enhancing its global search capability. This, therefore, motivates us to propose an opposite search mechanism to implement the adaptive *k*-means operation.

3. Proposed Algorithm

In this section, we propose a memetic algorithm with adaptive and opposite *k*-means operation (MAAOK) for data clustering. In the proposed method, apart from standard genetic operators, the solutions are evolved via an adaptive *k*-means operation (AKO) implemented based on an opposite search (OS) mechanism. The evolution will be terminated when the best individual in population does not change for T_g consecutive generations. Algorithm 1 shows a general procedure of the algorithm.

In the following sections, we shall give a brief description of the representation, genetic operation and fitness criterion used in the algorithm as well as the algorithm’s complexity. The details of proposed adaptive *k*-means operation and opposite search mechanism will be presented in

```
Step 1. Generate an initial population with $n$ individual solutions using real-value
        representation.
Step 2. Calculate the fitness of each solution in the initial population according to
        the SSE criterion.
Step 3. Select pairing parents using the roulette wheel strategy. This process is
        repeated until $n/2$ parent pairs are selected.
Step 4. Crossover the parent pairs to generate intermediate offspring and then
        apply mutation operation on the offspring.
Step 5. Perform an opposite search (see Section 5) based adaptive $k$-means
        operation (see Section 4) to improve the offspring.
Step 6. Calculate the fitness value of offspring if they are not calculated at step 5.
Step 7. Create a new population of size $n$ from the offspring with elite strategy if it
        is not created at step 5.
Step 8. Terminate the evolution if the stopping condition is met, otherwise go to
        Step 2.
Step 9. Output the best solution in the terminal population.
```

Algorithm 1. An adaptive and opposite *k*-means operation based memetic algorithm for data clustering.

Sections 4 and 5, respectively.

To initialize the solutions, a real-value based representation [6] is adopted to denote the values of cluster centers. For instance, to cluster a data set with three features, a solution can be represented as (0.65, 0,43, 0.32, 0.81, 0.72, 0.16), which encodes 2 clusters centered at (0.65, 0,43, 0.32) and (0.81, 0.72, 0.16), respectively. The initial values of solutions are randomly assigned according to the range of attributes. During the evolution, the roulette wheel strategy [50] is employed to select parent pairs for recombination. The selected parents then undergo the arithmetic crossover scheme [51] with a probability of P_c to generate intermediate offspring. Subsequently, the generated offspring is subject to Gaussian mutation [50] with a probability P_m.

The solution's fitness is measured according to the SSE function, which is calculated as:

$$SSE = \sum_{j=1}^{k} \sum_{x \in C_j} \left\| x - m_j \right\|^2 \tag{1}$$

where

$$m_j = \frac{1}{\left| C_j \right|} \sum_{x \in C_j} x \tag{2}$$

Here, x_i is the i^{th} object in data set, n is the size of data set, k denotes the number of clusters in the solutions, m_j represents the center of cluster C_j and $|C_j|$ denotes the number of data objects within cluster C_j.

Based on the calculated SSE value of each individual, we then calculate its fitness as $f = 1/SSE$, and thus higher fitness means better solutions. It should be noted that empty clusters could be existed in certain solutions. To punish these solutions, we rewrite the function as $f=1/SSE'$, where $SSE' = SSE + (c_e/c_t)*SSE$. Here, c_e and c_t denote the number of empty clusters and total clusters, respectively, in the solution.

During the run of the proposed method, the main computational load at each generation is the fitness evaluation as well as the implementation of adaptive and opposite *k*-means operation. Calculating a given solution's fitness takes $O(nkd^2)$ time, where n and d is the size and dimension of data set, respectively, and k denotes the number of clusters of the data set. The adaptive and opposite *k*-means operation for a given individual requires $O(\sigma nkd^2+kd)$ time, where σ denotes the number of iterations of *k*-means operation. The proposed method, therefore, has an overall time complexity of $O(\sigma nkd^2 g n_p)$, where n_p is the population size and g is the number of generations.

4. Adaptive *K*-means Operation

It has been well established that incorporation of the *k*-means operator can greatly enhance the EA-based clustering. Nevertheless, the *k*-means operator in existing methods is usually employed with fixed intensity and frequency, which may significantly limit the performance. Here, we try to address the issue by firstly presenting a generalized framework such that the frequency and intensity of *k*-means operator could be arbitrarily configured. Then, we devise an adaptive strategy to dynamically control its frequency and intensity during evolutionary clustering.

```
Step 1.  Before EA's evolution:
           i.   Initialize the frequency counter (curF) as curF = 1.
           ii.  Specify the frequency ($\tau$) and intensity ($\sigma$) of k-means operation.
Step 2.  During the evolution, if curF = $\tau$, then:
           i.   Apply $\sigma$ iterations of k-means on each offspring generated by the
                recombination operation of EA.
           ii.  Specify a new value for $\tau$ and $\sigma$, if necessary.
           iii. Reset curF = 0.
Step 3.  After each generation of evolution, increase the curF by 1 (i.e., curF = curF
         + 1).
```

Algorithm 2. The procedure of a generalized *k*-means usage framework.

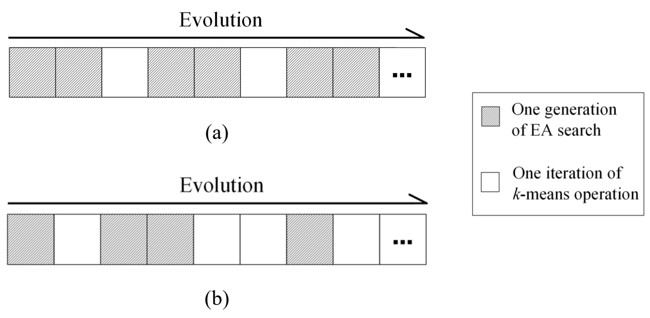

Evolution
...
(a)
Evolution
...
(b)
One generation of EA search
One iteration of *k*-means operation

Fig. 1. Two instances of GKUF with (a) fixed and (b) dynamic values of τ and σ.

```
Step 1.  Before EA's evolution:
         i.   Initialize the frequency counter (curF), frequency ($\tau$) and intensity ($\sigma$) of
              k-means operation as $curF = 1$, $\tau = 1$ and $\sigma = 1$, respectively.
         ii.  Record the fitness information of initial population.
Step 2.  During the evolution, if $curF = \tau$, then:
         i.   Apply $\sigma$ iterations of k-means on each offspring generated by the
              recombination operation of EA.
         ii.  Calculate the fitness value for each of the offspring.
         iii. Create a new population of size $n$ from the offspring with elite strategy.
         iv.  Record the diversity information of current population.
         v.   Update the values of $\tau$ and $\sigma$ according to equation (3) and (4),
              respectively.
         vi.  Reset $curF = 0$.
Step 3.  After each generation of evolution, increase the curF by 1 (i.e., $curF = curF$
         $+ 1$).
```

Algorithm 3. The procedure of adaptive *k*-means operation.

4.1. A Generalized K-means Usage Framework

To allow for variable frequency and intensity of *k*-means employment during EA evolution, a generalized *k*-means usage framework (GKUF), shown in Algorithm 2, has been devised. Essentially, in the framework, the frequency and intensity of *k*-means operator is controlled by introducing two free parameters τ and σ. Specifically, σ iterations of *k*-means will be performed on the offspring after each τ generations of EA evolution. By setting different values for τ and σ, we are therefore able to arbitrarily set the frequency and intensity of *k*-means operator during evolution. Two instances of GKUF are shown in Fig. 1. Fig. 1(a) gives an example of GKUF with fixed σ and τ respectively equaling to 1 and 2, which means that one iteration of *k*-means will be used to improve the individuals after each two consecutive generations of EA evolution. Clearly, the fixed *k*-means operation, as adopted in most of the existing EA-based clustering algorithms, could be considered as special cases of the proposed framework. Fig. 1(b) shows another example of GKUF, in which the values of τ and σ are dynamically changed along the evolution.

4.2. Adaptive Strategy

Although the above framework can enable us to dynamically vary the frequency and intensity of *k*-means operator, how to specify appropriate values of τ and σ is not a trivial task. It is due to that, for τ and σ, the selection of their values is usually problem-specific and also their values may vary during the execution of EAs. In the following subsection, we propose a mechanism called adaptive *k*-means operation (AKO), which seeks to automatically adjust the frequency and intensity of *k*-means operator based on the feedback information of evolution.

Among the various feedback information of evolution, the diversity of population is perhaps the most critical one. It is therefore natural to achieve the adaptation via monitoring this information during evolutionary clustering. A number of indexes have been designed to measure the diversity of population [52-54]. Here, we adopt an index, which is based on the idea proposed in [55], to measure the population diversity. Specifically, the diversity is measured in terms of the difference between the maximum ($f_{\max}$) and average ($\bar{f}$) fitness of the population. The rational is that the difference tends to be larger for a population with scattered solutions than that with converged ones. By employing an index based on such an idea, an AKO strategy is then proposed, the procedure of which is shown in Algorithm 3. In this strategy, initial values of the intensity and frequency of *k*-means operator are set as 1 (i.e., τ=σ=1) and the diversity information (i.e., $f_{\max}-\bar{f}$) of initial population is recorded. At the initial generation, one iteration of *k*-means will therefore be used to improve the offspring produced by recombination operation. Following the initial stage, we calculate the population diversity at each generation (i.e., $f'_{\max}-\bar{f}'$) after the *k*-means operation is evoked and compare it with that at previous generation. When a reduced diversity is recorded (i.e., $f'_{\max}-\bar{f}' < f_{\max}-\bar{f}$), the frequency of *k*-means operator will be adjusted as:

$$\tau = \left\lceil c_1 \cdot \frac{\bar{f}' - f'_{\min}}{f'_{\max} - f'_{\min}} \right\rceil \tag{3}$$

and the intensity will be kept as 1. The $f'_{\min}$, $f'_{\max}$ and $\bar{f}'$ here denote the minimum, maximum and average fitness of the population at current generation while c_1 denotes a constant value. The difference of $f'_{\max}-f'_{\min}$ in the equation is utilized for normalization purpose. Based on this formula, the *k*-means operator will therefore be applied with a lower frequency for a more converged population. While, if the comparison indicates an increased diversity (i.e., $f'_{\max}-\bar{f}' \geq f_{\max}-\bar{f}$), the intensity of *k*-means operator will be calculated as:

$$\sigma = \left[c_2 \cdot \frac{f'_{\max} - \bar{f}'}{f'_{\max} - f'_{\min}} \right] \tag{4}$$

and the frequency will be kept as 1. Here, c_2 is a constant value. Accordingly, based on this formula, the *k*-means operator will be more intensely used to improve the solutions during evolution when the population has a higher diversity. By jointly employing the above formulas, the proposed AKO strategy therefore aims to appropriately adjust the frequency and intensity of *k*-means local search based on the information of population diversity during evolution.

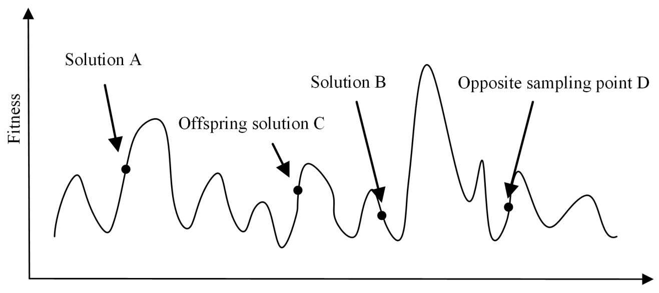


Fig. 2. Illustration of opposite search mechanism on one-dimensional maximum optimisation problem.

```
Step 1.  For each individual $p$ to be improved, determine an attraction point $p_1$ with a
         higher fitness value than $p$ using the roulette wheel strategy.
Step 2.  Apply k-means operation (whose intensity is determined by the AKO
         mechanism) on the individual $p$.
Step 3.  Sample an opposite point $p'$ for the individual $p$ using the equation (5).
Step 4.  Apply the same k-means operation on the individual $p'$.
Step 5.  Calculate the fitness of individuals $p$ and $p'$. If the fitness of $p'$ is better than
         that of $p$, then replace the individual $p$ with $p'$ as the offspring.
```

Algorithm 4. The procedure of opposite search based adaptive *k*-means operation.

5. Opposite Search Mechanism

It has been widely agreed that local searches are able to improve exploitation capability of EAs, thus making them efficient. However, at the same time, local searches could also accelerate the speed of genetic drift in a population, which leads to premature convergence. This is due to that local searches in existing MAs are typically implemented directly on the individual without considering the exploration aspect of population. As a result, under the search mechanism of EAs along with local searches, the individuals will have a high tendency to move towards the best solution in population, before the search space being appropriately explored. To address such an issue, an opposite search (OS) mechanism has been further introduced in this section to implement the AKO.

Compared to the direct application of local search operator, the OS mechanism based local search is able to enhance the exploration capacity of a population. An illustration of this is shown in Fig. 2 on a one-dimensional maximum optimization problem. Supposing there are two individual solutions *A* and *B* in the population, generally the solution *B* will be attracted towards the solution *A* which has a better fitness value under the mechanism of EA. Assume that an offspring *C*, produced by genetic operators, is the next point of inquiry of solution *B*. Once the solution *B* is replaced by the solution *C* under the replacement strategy of EA, the segment *CB* will unlikely be searched again. As a result, the segment *CB* may not be appropriately explored. At the same time, as the solution *B* tends to be attracted towards the solution *A*, premature convergence may occur. On the other hand, if the solution *B* is replaced by the solution *D*, which is located at the opposite direction of the search along the solution *B*, then the segment *CB* will still get a chance to be searched and meanwhile the solutions tend to be diversified. Consequently, the exploration capability of solutions *A* and *B* could be significantly enhanced.

The proposed OS mechanism uses the population information along opposite direction of the individual attracted to performing the AKO. Specifically, for an individual p to be improved, the mechanism first determines an attraction point for the individual by choosing an individual p_1 with a higher fitness value than p from the population. Then, the *k*-means operation (whose intensity is determined by the AKO mechanism) is performed to improve the individual p. Subsequently, an opposite point p' is sampled using the following equation:

$$p' = (1-\lambda)\cdot p_1 + \lambda \cdot p \quad (5)$$

This equation is obtained as follows. Let $\left|\overrightarrow{p_1 p}\right|$ and $\left|\overrightarrow{p_1 p'}\right|$ denote the Euclidean distances from p_1 to p and p', respectively. Then, the relationship between $\left|\overrightarrow{p_1 p}\right|$ and $\left|\overrightarrow{p_1 p'}\right|$ can be written as:

$$\left|\overrightarrow{p_1 p}\right| \Big/ \left|\overrightarrow{p_1 p'}\right| = 1/\lambda, \quad (6)$$

where λ is a constant value greater than 1. The position of sampling point p' can be subsequently derived as:

$$\Leftrightarrow \left(\vec{p} - \vec{p_1}\right) / \left(\vec{p'} - \vec{p_1}\right) = 1/\lambda \quad (7)$$

$$\Leftrightarrow \overrightarrow{p'} = (1-\lambda)\cdot\overrightarrow{p_1} + \lambda\cdot\overrightarrow{p}\,. \tag{8}$$

According to the above equation, the position of sampling point p' is determined by the value of λ. A large value of λ will lead p' sampled far away from p, while a small value will make p' close to p. To ensure a good performance of OS mechanism, the value of λ should be set appropriately. A too small value of λ will lead to excessiveness of exploitation and rapidly result in convergence to local optima. Conversely, a too large value of λ could critically slow down the convergence rate. Thus, to balance the search, a value of 1.8 has been empirically identified and used in our experiments.

After obtaining the opposite point p', the same k-means operation as applied on the individual p is employed to improve it. Finally, replace the individual p with p' if it has a better fitness, otherwise keep the p as the offspring. The procedure of the mechanism is shown in Algorithm 4.

6. Experiments

In this section, we first describe the experimental data and parameter configurations of our proposed method. Then, the performance of devised mechanisms is evaluated. After that, we compare our algorithm with related work. All experiments were performed on a workstation with an Intel (R) Core^{TM} i7-3630QM CPU at 2.40GHz and 8 GB RAM running Windows^{TM} 7. The results reported here were averaged over 100 trials of the methods, unless stated otherwise.

Table I: A List of Data Sets Used in the Experiments

Data sets	*No.* of data points	*No.* of attributes	*No.* of clusters
*Art*_9	3300	2	9
*Art*_15	4400	2	15
*Art*_20	4000	2	20
Car	1728	6	4
Musk	476	166	2
Landsat	4435	36	6
Turkiye	5820	32	5
MFCCs	7195	22	10
Subcellcycle	387	17	5
*Yeast*2945	2945	15	30

6.1. Data Sets and Parameter Settings

A series of data sets, as listed in Table I, including artificial as well as real data have been used for evaluation purpose. The artificial data (i.e., *Art*_9, *Art*_15 and *Art*_20 as shown in Fig. 3) are generated with various levels of difficulty for partitioning. The *Art*_9 is relatively simple. There are four dense and sparse clusters, respectively, in the data set along with one highly sparse cluster located at the middle of Fig. 3(a). In *Art*_15, the clusters are generated with rather different volumes and sizes while many of them are overlapped. In *Art*_20, there are sixteen clusters in the middle along with four isolated clusters, which makes the problem even difficult to be solved.

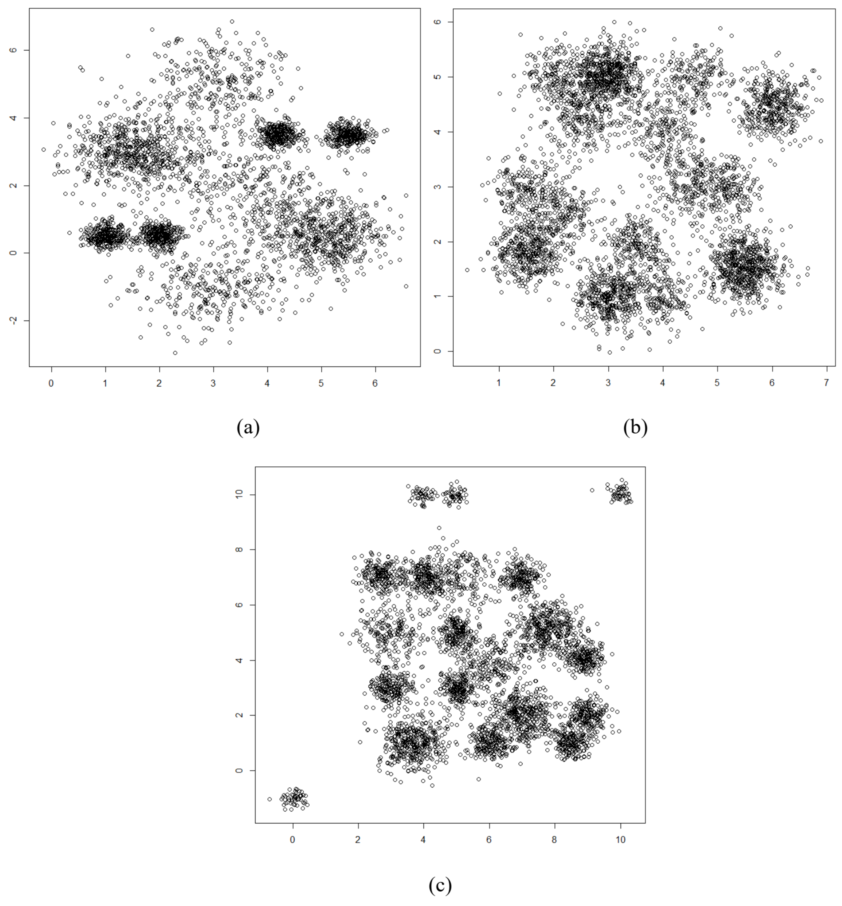
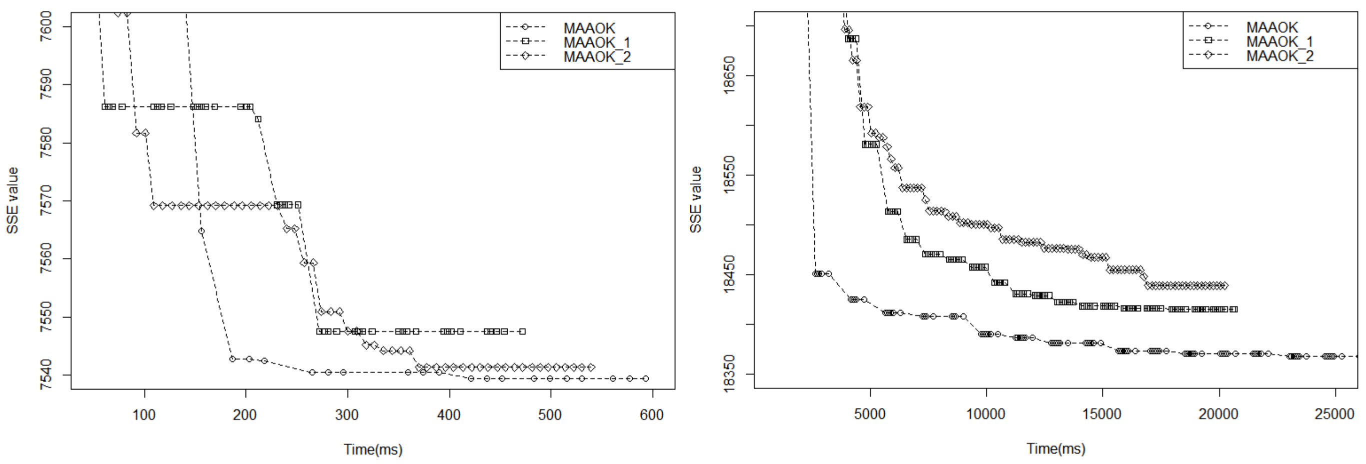


Fig. 3. Artificial data sets (a) *Art*_9, (b) *Art*_15 and (c) *Art*_20.

For real data, we consider the *Car*, *Musk*, *Landsat*, *Turkiye* and *MFCCs* data sets, which are available at UCI Repository [56]. The *Car* data consist of 1728 data objects with 6 attributes (i.e., buying, doors, maint, lug_boot, persons, and safety). This data set will be grouped into 4 clusters corresponding to 4 types of car evaluation results. The *Musk* data contain 476 instances of 166 attributes, which are used to describe the molecules depending upon the exact shape or conformation of molecule. These molecules are judged to be either musks or non-musks. Therefore, there are 2 clusters in the data. The *Landsat* data contain multi-spectral values of pixels in 3*3 neighborhoods in a satellite image. This data has 4435 instances of 36 attributes and will be clustered into 6 groups corresponding to 6 classes of land types. The *Turkiye* data contain a total 5820 evaluation scores with 32 attributes collected from the students at Gazi University. These data are expected to be grouped into 5 clusters. The *MFCCs* consist of 22 attributes of syllables of anuran (frogs) calls. There is a total of 7195 records with 3 labels (i.e., family, genus and species). Based on the label of species, there are 10 clusters in this data.

Additionally, gene expression data including the *Subcellcycle* and *Yeast*2945 have also been considered in the experiments. Both data sets are part of the yeast cell-cycle data and have been widely used for bioinformatics study [57]. The *Subcellcycle* contains expression information of 384 genes, which corresponds to five phases (i.e., S, M, G2, early and late G1) during the cell-cycle. Correspondingly, 5 clusters are existed in this data. The *Yeast*2945 data, given by Tavazoie *et al.* [58], have expression information of 2945 genes. As in [58], we also partition the data into 30 clusters. All the data mentioned above are normalized via the *Z*-score method such that each variable has a mean value of 0 with standard deviation of 1. For real data, principal component analysis has also been employed to reduce the dimension by selecting the top principal components, which account for over 95% of the variance.

In experiments, several parameters in our proposed algorithm need to be specified, which involve the rates of mutation and crossover, population size, termination criterion, scaling parameters c_1 and c_2 in AKO and control constant λ in OS mechanism. Specifically, the mutation rate and crossover rate are set to be 0.03 and 0.7, respectively. These values are determined experimentally on the aforementioned data sets. Three trials of the proposed method are performed on various mutation and crossover rates while the rest parameter values are kept fixed. Generally, we find that a mutation probability value ranging from 0.02 to 0.05 along with a crossover probability value ranging from 0.6 to 0.85 could offer the best results. A size of 30 is used for population initialization, and the proposed method will be terminated when there is no improvement detected for the best individual in T_g=20 consecutive generations. A larger population size or value T_g may result in a more expensive algorithm with no significant improvement of performance. The values of scaling constants c_1 and c_2 in AKO are both set to be 8. For the scaling constants, there is a trade-off between computational efficiency and solution quality. A larger value of scaling constants will generally lead to a more efficient algorithm, but it will be more susceptible to less promising solutions. Our experiments indicate that a value between 7 to 10 could give an adequate trade-off between the efficiency and solution quality across the experimental data sets. For the λ in OS mechanism, a value of 1.8 is used.



(a) (b)

Fig. 4. Typical results of SSE values over runtime corresponding to the MAAOK, MAAOK_1 and MAAOK_2 on (a) *Car* and (b) *Yeast*2945 data set.

Table II: Comparing the Results Delivered by Our Proposed Method and its Two Variants

Data Sets	Evaluation Indexes	Methods		
		MAAOK	MAAOK_1	MAAOK_2
Art_9	SSE	518.271	518.668	**518.092**
	Runtime (s)	0.90165	**0.58299**	0.86062
Art_15	SSE	**330.178**	330.35	330.388
	Runtime (s)	2.19899	**1.39015**	2.02118
Art_20	SSE	**208.878**	209.278	210.028
	Runtime (s)	3.16697	**2.19005**	3.04961
Car	SSE	**7542.93**	7549.98	7546.31
	Runtime (s)	0.4908	**0.38356**	0.44614
Musk	SSE	**53379.4**	**53379.4**	**53379.4**
	Runtime (s)	0.14471	**0.1083**	0.17627
Landsat	SSE	26985.4	26986	**26985.3**
	Runtime (s)	1.62432	**1.22219**	1.93652
Turkiye	SSE	**49999.2**	49999.3	**49999.2**
	Runtime (s)	3.09903	**2.30424**	4.02371
MFCCs	SSE	**44651.1**	44708.8	44705
	Runtime (s)	8.60205	**7.03384**	11.1001
Subcellcycle	SSE	**1646.2**	1653.43	1651.43
	Runtime (s)	0.09735	**0.07045**	0.08927
*Yeast*2945	SSE	**18382.8**	18410.3	18439.8
	Runtime (s)	25.2307	19.3308	**18.1835**

6.2. Results

Firstly, we evaluate the impact of AKO and OS mechanisms in our proposed method. To this end, we examine and compare our method, MAAOK, with its variants: MAAOK without the OS mechanism to implement the AKO (denoted as MAAOK_1), which were proposed in our previous conference work [17] and MAAOK with neither AKO nor OS mechanism (i.e., only one iteration of *k*-means is incorporated to improve each offspring individual at every generation), which is denoted as MAAOK_2. These variants are run with the same setting of parameters as in MAAOK. Their performance in terms of average SSE and runtime are shown in Table II.

Comparing the MAAOK_1 and MAAOK_2, it can be found that, by incorporating the AKO mechanism, the MAAOK_1 is more efficient than MAAOK_2 on all data sets used in the experiments. For instance, on *Art*_20, MAAOK_2 needs 3.04961s to give solutions with an average SSE value of 210.028. By contrast, MAAOK_1 takes only 2.19005s on average to provide solutions with an even better average SSE value of 209.278. On *Musk* data set, the MAAOK_2 and MAAOK_1 require 0.17627s and 0.1083s, respectively, to deliver solutions with same average SSE value of 53379.4. Looking at the MAAOK_1 and MAAOK, it can be observed that the OS mechanism is generally able to improve the quality of solutions. For example, on the *MFCCs* data, by incorporating the OS mechanism to implement the AKO, the average SSE value delivered by MAAOK is 44651.1, which is better than the one delivered by MAAOK_1. This is due to that the OS mechanism can be used to maintain an appropriate balance between the exploitation and exploration of evolutionary search. To clearly show the results, typical convergence curves of the three algorithms on two representative data sets have also been shown in Fig. 4. From the above results, we can conclude that the AKO mechanism is able to improve the efficiency of evolution while the OS mechanism helps to locate promising solutions. By incorporating the two mechanisms, the resulting algorithm is therefore able to efficiently and effectively search the space to deliver high quality solutions.

Next, experiments are carried out to examine the performance of our method by comparing it with related methods, listed below. These methods are either classical or recently proposed EA based methods, which is used to optimize SSE for data clustering.

- GKA [13], devised for clustering, is a variant of GA. In this method, the crossover operation is removed while one iteration of *k*-means is incorporated for improving the offspring solutions during evolutionary clustering.
- EPSONS [59] is based on a particle swarm optimisation (PSO) algorithm. In this method, a neighborhood search strategy along with a diversity strategy is applied to improve the evolutionary clustering.
- MEQPSO [60] is based on a variant of quantum-behaved PSO. This method also adopts one iteration of *k*-means as a local search procedure

in order to fine-tune clustering solutions during evolution, which has been applied for clustering gene expression data.

- HABC [61] is based on an artificial bee colony (ABC) algorithm, in which the information exchange among bees is enhanced by a newly designed crossover operation.
- CGABC [62] is a hybrid ABC algorithm, in which two local search paradigms (i.e., chaotic local search and gradient search) are incorporated to improve the convergence rate of the algorithm. These two local procedures are applied to fine-tune the best individual at each generation.

To ensure a fair comparison, we implement all the methods based on the same termination condition (i.e., the best individual in the population does not change in 20 consecutive generations). The values of rest parameters in GKA, EPSONS, MEQPSO, HABC and CGABC remain the same as specified in the original work.

To evaluate the quality of clustering solutions, two additional metrics, the Entropy [63] index and Calinski-Harabasz (CH) [64] index, have also been used to report the results. The Entropy is an external index, which can be used to measure the purity of clusters with respect to given class labels of the data. For a given class distribution of the data, the entropy of jth cluster in the solution is computed as

$$E_j = -\sum\nolimits_{i=1}^{k} p_{ij} \log(p_{ij}) \tag{9}$$

where p_{ij} is the probability that a member of jth cluster belongs to ith cluster and k is the cluster number. Then, the total entropy of all clusters is calculated as:

$$E = \sum_{j=1}^{k} \frac{n_j}{n} E_j \tag{10}$$

where n_j and n are the size of jth cluster and the entire data set, respectively. A lower value of E indicates a higher correctness of the solutions. The CH is an internal index, which evaluates the validity of clusters based on the average within- and between-cluster sum of squares. Given a data set of n objects with k clusters, this index is computed as:

$$CH = \frac{n-k}{k-1} \cdot \frac{\sum\nolimits_{i=1}^{k} n_i \left\| m_i - m \right\|^2}{\sum\nolimits_{i=1}^{k} \sum\nolimits_{j=1}^{n_i} \left\| x_j - m_i \right\|^2} \tag{11}$$

where m and m_i are the centers of data set and cluster C_i, respectively, while n_i denotes the number of objects belonging to cluster C_i. A larger value of CH corresponds to a better clustering solution.

Table III: Comparing Results Delivered by the Six Methods on Experimental Data Sets

Data Sets	Evaluation Indexes	Methods					
		GKA	EPSONS	MEQPSO	HABC	CGABC	MAAOK
*Art*_9	SSE	524.412	553.788	530.582	534.859	547.517	**518.271**
	CH	4737.26	4501.1	4690	4674.45	4562.16	**4827.33**
	Entropy	0.601457	0.659249	0.603306	0.604496	0.650066	**0.496718**
	Runtime (s)	0.92464	19.0842	1.27685	27.6927	**0.48518**	0.90165
*Art*_15	SSE	331.42	365.482	384.753	345.371	362.812	**330.178**
	CH	7923.12	7227.27	6873.65	7677.42	7315.74	**8034.64**
	Entropy	0.514951	0.624709	0.697198	0.560497	0.610993	**0.510835**
	Runtime (s)	1.52056	43.8815	1.62543	65.0253	**0.97377**	2.19899
*Art*_20	SSE	240.42	243.048	368.258	232.273	240.387	**208.878**
	CH	6754.68	6753.74	4541.29	7035.86	6848.34	**7813.33**
	Entropy	0.539778	0.541177	0.776882	0.508279	0.501505	**0.415091**
	Runtime (s)	2.45264	76.7496	1.27463	75.3534	**1.23082**	3.16697
Car	SSE	7550.1	7575.11	7561.21	7579.47	7585.86	**7542.93**
	CH	214.482	211.825	213.329	211.429	210.775	**215.231**
	Entropy	1.03924	1.08199	1.05919	1.08073	1.06695	**1.03538**
	Runtime (s)	0.31136	12.3635	0.36169	18.5743	**0.27474**	0.4908
Musk	SSE	**53379.4**	53686.7	**53379.4**	53555.6	55335	**53379.4**
	CH	**192.612**	184.875	**192.612**	189.907	172.73	**192.612**
	Entropy	**0.973769**	0.986263	**0.973769**	0.982761	0.97756	**0.973769**
	Runtime (s)	0.14245	33.4754	**0.11345**	13.7734	0.17705	0.14471
Landsat	SSE	26987.2	29707.4	26988	30116.1	30074.3	**26985.4**
	CH	4105.29	3580.51	4117.36	3624.42	3643.21	**4119.31**
	Entropy	0.943224	1.06594	0.940967	1.09427	1.06457	**0.940038**
	Runtime (s)	1.23431	129.715	2.26149	70.9661	**0.93727**	1.62432
Turkiye	SSE	51419.1	56388.8	49999.8	56482.9	52867.6	**49999.2**
	CH	3523.57	3021.07	3690.9	3023.84	3421.51	**3692.46**
	Entropy	1.02102	1.2133	0.993195	1.22424	1.08775	**0.9929**
	Runtime (s)	3.67825	369.109	2.76554	168.024	**2.34858**	3.09903
MFCCs	SSE	45216.2	54268.7	45453	60259.6	49445.7	**44651.1**
	CH	1862.85	1355.88	1857.56	1140.7	1659.34	**1907.04**
	Entropy	0.707044	0.82942	0.724194	0.986066	0.807823	**0.670661**
	Runtime (s)	5.26716	742.619	8.19415	319.909	**4.60402**	8.60205
Subcellcycle	SSE	1662.59	1704.89	1659.6	1675.33	1684.6	**1646.2**
	CH	259.456	243.574	259.112	256.829	254.951	**262.982**
	Entropy	N/A	N/A	N/A	N/A	N/A	N/A
	Runtime (s)	0.10638	9.31111	0.10562	4.44142	**0.06226**	0.09735
*Yeast*2945	SSE	18412.7	19873.7	18599.2	21576.2	18608.2	**18382.2**
	CH	129.583	110.376	127.236	90.3643	127.183	**129.979**
	Entropy	N/A	N/A	N/A	N/A	N/A	N/A
	Runtime (s)	13.2146	913.153	11.0537	384.341	**5.93786**	25.2307

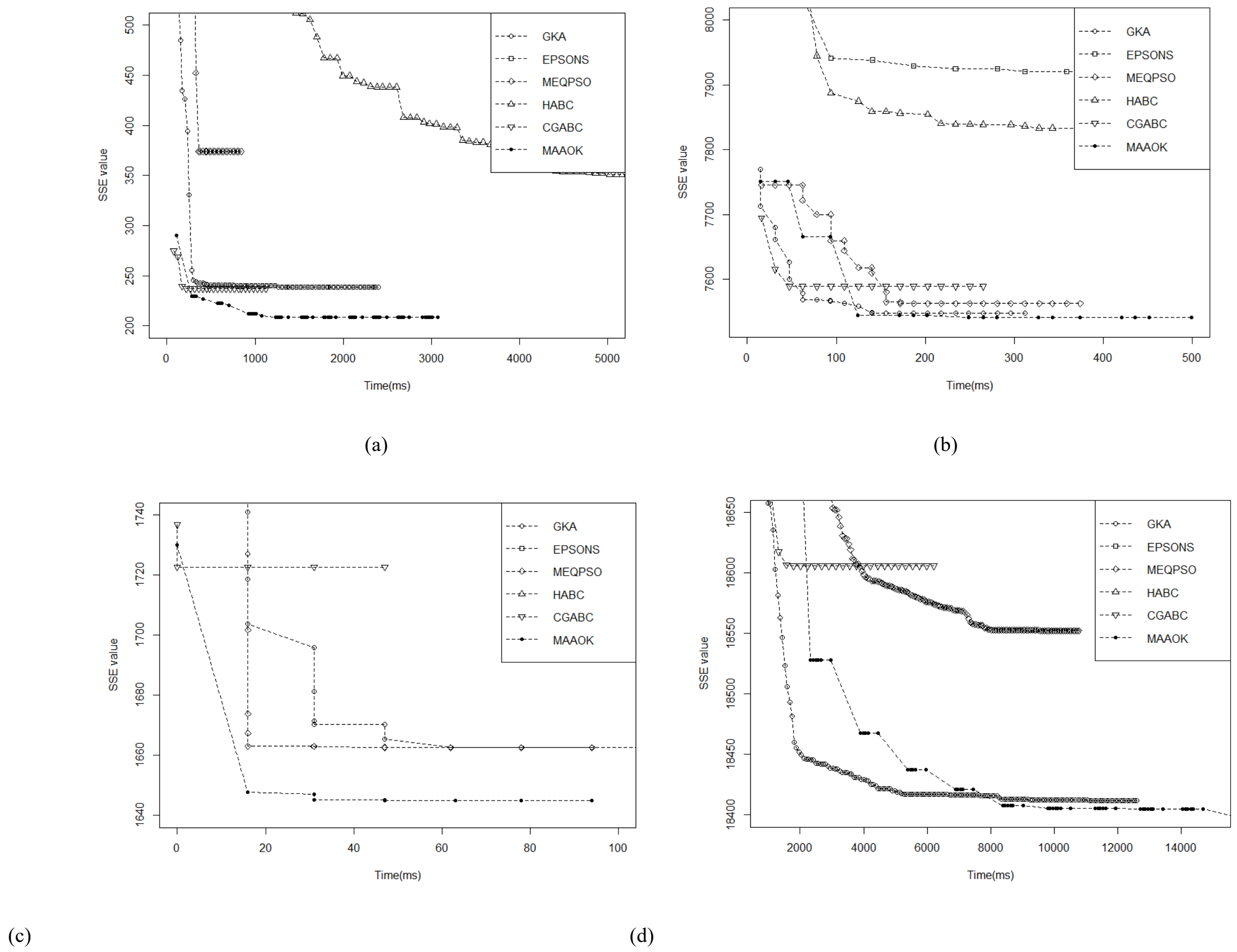


Fig. 5. Typical results of SSE values over runtime corresponding to the six algorithms on (a) *Art*_20, (b) *Car*, (c) *Subcellcycle* and (d) *Yeast*2945 data set.

The comparison results in terms of SSE, Entropy, CH values and running time are reported in Table III. Additionally, typical convergence curves of the six methods on several representative data sets have been shown in Fig. 5. The results reveal that, compared with the five related algorithms, our proposed method can generally deliver better clustering solutions. For example, on *Car*, the GKA, EPSONS, MEQPSO, HABC and CGABC give solutions with average SSE values of 7550.1, 7575.11, 7561.21, 7579.47 and 7585.86, respectively, while the solutions delivered by MAAOK achieve7542.93. Further, the results indicate that the MAAOK could be particularly useful for clustering problems involving large search spaces. For instance, on *Art*_20, the GKA, EPSONS, MEQPSO, HABC and CGABC provide solutions with average SSE values of 240.42, 243.048, 368.258, 232.273 and 240.387, respectively. By contrast, our method reaches 208.878. Similar results can also be observed in terms of Entropy and CH index. For example, in terms of Entropy, the GKA, EPSONS, MEQPSO, HABC and CGABC deliver solutions with average Entropy of 0.707044, 0.82942, 0.724194, 0.986066 and 0.807823, respectively, on the *MFCCs*. By contrast, the solutions delivered by the MAAOK have an average Entropy value of

0.670661. Our method can therefore be used to identify more accurate clustering results. In terms of the internal index of CH on the same data set, the solutions provided by the above five methods as well as our proposed method have the average values of 1862.85, 1355.88, 1857.56, 1140.7, 1659.34 and 1907.04, respectively. This could further confirm that our proposed can be used to identify better clustering solutions than other compared methods. It is not surprising that the performances of the EPSONS, HABC and CGABC are worse since they do not use local search operation or use it to improve the best solution only during evolution, thus limiting their performance. By employing *k*-means to fine-tune the solutions during evolution, the GKA and MEQPSO can have a better performance. However, they still perform worse than MAAOK. The better performance of MAAOK is largely contributed by the AKO and OS mechanisms. Rather than employing the *k*-means operator in a fixed manner, the AKO mechanism can appropriately adjust its frequency and intensity depending on the problem instances to be solved as well as the stages of evolution, which therefore makes the algorithm effective. While, the OS mechanism considers the exploration aspect of population to perform the *k*-means operation, rather than applying it directly on the individual. Such a mechanism is able to alleviate the tendency of population to converge prematurely and enhance its global search capability. This is particularly important for the clustering problem with complex search space.

Regarding to efficiency, the results reveal that our proposed algorithm is generally faster or comparable than EPSONS, HABC, GKA and MEQPSO, but outperformed by CGABC. For example, on *Landsat* data, the GKA, EPSONS, MEQPSO, HABC and CGABC need 1.23431s, 129.715s, 2.26149s, 70.9661s and 0.93727s, respectively, while the MAAOK takes 1.62432s on average to converge. Although the CGABC is efficient, it could be easily trapped into less promising local optima, which leads to much worse solution quality than our proposed algorithm.

7. Conclusions

In this work, we have investigated the issue of appropriately employing the *k*-means local search operator in EA-based clustering. For this purpose, a generalized framework, which supports the arbitrary setting of the frequency and intensity of *k*-means operator during evolutionary clustering, has been first introduced. Then, we have proposed an adaptive strategy to dynamically control the frequency and intensity of *k*-means operator during evolution. Further, an opposite search mechanism has been devised to implement the adaptive *k*-means operation, thus appropriately exploring the search space. Our results have shown that the AKO mechanism can improve the efficiency of evolution while the OS mechanism is able to enhance the exploration capacity of population, thus avoiding less promising optima. By incorporating the two mechanisms, our proposed algorithm is therefore capable of efficiently and effectively delivering high quality solutions, and outperforms the compared methods.

The work proposed in this paper can be extended further in several directions. Firstly, designing and/or employing control mechanisms [65], [66] to dynamically set the frequency and intensity of *k*-means operator could be an interesting direction. Their influence on the algorithm's performance can then be studied. Second, it would be desirable to develop a method to automatically determine the attraction point in the OS mechanism. Additionally, while the effectiveness of AKO and OS mechanism has been demonstrated for EA-based data clustering, they are sufficiently robust and flexible to be incorporated into other metaheuristic algorithms, such as particle swarm optimization [67], [68] to deal with clustering and its related applications, such as image segmentation [69-71]. These can also be investigated in the future.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant No. 61873082, Grant 62003121 and Zhejiang Provincial Natural Science Foundation of China under Grant LQ20F030014.

References

[1] A. K. Jain, "Data clustering: 50 years beyond *k*-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651-666, 2010.

[2] J. C. Bezdek and N. R. Pal, "Some new indexes of cluster validity," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 28, no. 3, pp. 301-315, 1998.

[3] R. O. Duda, P. E. Hart, and D. G. Stork, "Pattern classification," *En Broeck the Statistical Mechanics of Learning Rsity*, 2nd ed., 2000.

[4] M. Popescu, J. C. Bezdek, T. C. Havens and J. M. Keller, "A cluster validity framework based on induced partition dissimilarity," *IEEE Trans. on Cybern.*, vol. 43, no. 1, pp. 308-320, 2013.

[5] Y. Liu, Z. Li, H. Xiong, X. Gao, J. Wu and S. Wu, "Understanding and enhancement of internal clustering validation measures," *IEEE Trans. on Cybern.*, vol. 43, no. 3, pp. 982-994, 2013.

[6] E. R. Hruschka, R. J. G. B. Campello, A. A. Freitas and A. C. Ponce Leon F. de Carvalho, "A survey of evolutionary algorithms for clustering," *IEEE Trans. Syst., Man, Cybern. C*, vol. 39, no. 2, pp. 133-155, 2009.

[7] J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *Proc.* of, Berkeley Symposium on Mathematical Statistics and Probability, pp. 281-297, 1967.

[8] R. D. Baruah, P. Angelov, “DEC: Dynamically evolving clustering and its application to structure identification of evolving fuzzy models,” *IEEE Trans. Cybern.*, vol. 44, no. 9, pp. 1619-1631, 2014.

[9] D. Cai and X. Chen, “Large scale spectral clustering via landmark-based sparse representation,” *IEEE Trans. Cybern.*, vol. 45, no. 8, pp. 1669-1680, 2015.

[10] X. Chang, Q. Wang, Y. Liu and Y. Wang, “Sparse regularization in fuzzy c-means for high-dimensional data clustering,” *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2616-2627, 2017.

[11] L. O. Hall, I. B. Ozyurt and J. C. Bezdek, “Clustering with a genetically optimized approach,” *IEEE Trans. Evolutionary Computation*, vol. 3, no. 2, pp. 103-112, 1999.

[12] U. Maulik and S. Bandyopadhyay, “Genetic algorithm-based clustering technique,” *Pattern Recognition*, vol.33, pp.1455-1465, 2000.

[13] K. Krishna and M. N. Murty, “Genetic *k*-means algorithm,” *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, 29(3), pp.433-439, 1999.

[14] J. Kivijärvi, P. Fränti, and O. Nevalainen, “Self-adaptive genetic algorithm for clustering,” *J. Heuristics*, vol. 9, no. 2, pp. 13-129, 2003.

[15] Y. Lu, S. Lu, F. Fotouhi, Y. Deng, and S. J. Brown, “Incremental genetic *k*-means algorithm and its application in gene expression data analysis,” *BMC Bioinformat.*, vol. 28, no. 172, 2004.

[16] W. Sheng, A. Tucker, X. Liu, “A niching genetic *k*-means algorithm and its applications to gene expression data,” *Soft Computing*, vol. 14, no. 1, pp. 9-19, 2010.

[17] X. Wang and W. Sheng, “Adaptive usage of *k*-means in evolutionary optimized data clustering”, *International Conference on Machine Learning and Cybernetics*, pp. 15-20, 2017.

[18] Y. Zhong, A. Ma, Y.S. Ong, Z. Zhu and L. Zhang, “Computational intelligence in optical remote sensing image processing,” *Applied Soft Computing*, vol. 64, pp. 75-93, 2018.

[19] A. Ma, Y. Zhong, D. He and L. Zhang, “Multiobjective subpixel land-cover mapping,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 1, pp. 422-435, 2018.

[20] E. R. Hruschka and N. F. F. Ebecken, “A genetic algorithm for cluster analysis,” *Intelligent Data Analysis*, vol. 7, no. 1, pp. 15-25, 2003.

[21] S. Deng, Z. He, and X. Xu, “G-ANMI: A mutual information based genetic clustering algorithm for categorical data,” *Knowl. Based Syst.*, vol. 23, no. 2, pp. 144–149, 2010.

[22] U. Maulik and S. Bandyopadhyay, “Non-parametric genetic clustering: Comparison of validity indices,” *IEEE Trans. Syst., Man, Cybern. B*, vol. 31, no. 1, pp. 120–125, 2001.

[23] A. Sibil, N. Godin, M. R’Mili, E. Maillet, and G. Fantozzi, “Optimization of acoustic emission data clustering by a genetic algorithm method,” *J. Nondestructive Eval.*, vol. 31, no. 2, pp. 169–180, 2012.

[24] L. Y. Tseng and S. B. Yang, “A genetic approach to the automatic clustering problem,” *Pattern Recognit.*, vol. 34, no. 2, pp. 415–424, 2001.

[25] J. Xiao, Y. P. Yan, J. Zhang, and Y. Tang, “A quantum-inspired genetic algorithm for *k*-means clustering,” *Expert Syst. Appl.*, vol. 37, no. 7, pp. 4966–4973, 2010.

[26] A. H. Beg and M. Z. Islam, “Novel crossover and mutation operation in genetic algorithm for clustering,” *IEEE Congress on Evolutionary Computation, Vancouver*, BC, pp. 2114-2121, 2016.

[27] A. H. Beg and M. Z. Islam, "Clustering by genetic algorithm- high quality chromosome selection for initial population," *IEEE Conference on Industrial Electronics and Applications*, Auckland, pp. 129-134, 2015.

[28] A. H. Beg and M. Z. Islam, "Genetic algorithm with novel crossover, selection and health check for clustering," *European Symposium on Artificial Neural Networks*, 2016.

[29] M. A. Rahman and M. Z. Islam, "A hybrid clustering technique combining a novel genetic algorithm with *K*-Means," *Knowledge-Based Systems*, vol. 71, no. 71, pp. 345-365, 2014.

[30] S. Bandyopadhyay and S. Saha, "A point symmetry-based clustering technique for automatic evolution of clusters," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 11, pp. 1441–1457, 2008.

[31] W. Sheng, G. Howells, M. Fairhurst, and F. Deravi, "A memetic fingerprint matching algorithm," *IEEE Trans. Inf. Forensics Security*, vol. 2, no. 3, pp. 402–412, 2007.

[32] W. Sheng, S. Swift, L. Zhang, and X. Liu, "A weighted sum validity function for clustering with a hybrid niching genetic algorithm," *IEEE Trans. Syst., Man Cybern. B, Cybern.*, vol. 35, no. 6, pp. 1156–1167, 2005.

[33] M. Laszlo and S. Mukherjee, "A genetic algorithm using hyper-quadtrees for low-dimensional *k*-means clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 533–543, 2006.

[34] H. He and Y. Tan, "A two-stage genetic algorithm for automatic clustering," *Neurocomputing*, vol. 81, pp. 49–59, 2012.

[35] P. Scheunders, "A genetic c-means clustering algorithm applied to color image quantization," *Pattern Recognit.*, vol. 30, no. 6, pp. 859–866, 1997.

[36] E. R. Hruschka, R. J. G. B. Campello, and L. N. D. Castro, "Evolving clusters in gene-expression data," *Inf. Sci.*, vol. 176, no. 13, pp. 1898–1927, 2006.

[37] P. Merz and A. Zell, "Clustering gene expression profiles with memetic algorithms," *Lect. Notes Comput. Sci.*, pp. 811–820, 2002.

[38] Y. S. Ong, M. H. Lim and X. S. Chen, "Research frontier: Memetic computation-past, present & future," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 24-36, 2010.

[39] W. E. Hart, "Adaptive global optimization with local search," *Mechanical Engineering*, vol. 251 no. 7, pp. 269-298, 1994.

[40] M. Lozano, F. Herrera, N. Krasnogor, *et al.*, "Real-coded memetic algorithms with crossover hill-climbing", *Evolutionary Computation*, vol. 12 no. 3, pp. 273-302, 2004.

[41] D. Molina, F. Herrera and M. Lozano, "Adaptive local search parameters for real-coded memetic algorithms," *IEEE Congress on Evolutionary Computation*, Edinburgh, Scotland, pp. 888-895 vol. 1, 2005.

[42] Q. H. Nguyen, Y. S. Ong and M. H. Lim, "A probabilistic memetic framework," *IEEE Trans. Evolutionary Computation*, vol. 13, no. 3, pp. 604-623, 2009.

[43] H. Nobahari and D. Darabi, "A new adaptive real-coded memetic algorithm," *International Conference on Artificial Intelligence and Computational Intelligence*, Shanghai, pp. 368-372, 2009.

[44] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Trans. Evolutionary Computation*, vol. 12, no. 1, pp. 107-125, 2008.

[45] C. Liu and B. Li, "Memetic algorithm with adaptive local search depth for large scale global optimization," *IEEE Congress on Evolutionary Computation*, Beijing, pp. 82-88, 2014.

[46] N. K. Bambha, S. S. Bhattacharyya, J. Teich and E. Zitzler, "Systematic integration of parameterized local search into evolutionary algorithms," *IEEE Trans. Evolutionary Computation*, vol. 8, no. 2, pp. 137-155, 2004.

[47] A. Ma, Y. Zhong and L. Zhang, "Adaptive multiobjective memetic fuzzy clustering algorithm for remote sensing imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 8, pp. 4202-4217, 2015.

[48] Y. Zhong, A. Ma and L. Zhang, "An adaptive memetic fuzzy clustering algorithm with spatial information for remote sensing imagery," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 4, pp. 1235-1248, 2014.

[49] N. Shahidi, H. Esmaeilzadeh, M. Abdollahi, E. Ebrahimi and C. Lucas, "Self-adaptive memetic algorithm: An adaptive conjugate gradient approach," *IEEE Conference on Cybernetics and Intelligent Systems*, pp. 6-11 vol.1, 2004.

[50] T. Back, *Evolutionary algorithms in theory and practice*. London, U.K., Oxford Univ. Press, p. 120, 1996.

[51] Z. Michalewicz, *Genetic algorithms + data structure = evolution programs*, 3rd ed. New York, NY, USA: Springer, 1996.

[52] E Zitzler, M. Laumanns and L. Thiele, "SPEAII: Improving strength pareto evolutionary algorithm," Technical Report 103, Computer Engineering and Networks Laboratory, Swiss Federation of Technology, Zurich, 2001.

[53] X Li, J. Branke, and M. Kirley, "Performance measures and particle swarm methods for dynamic multiobjective optimization problems," *Genetic and Evolutionary Computation Conference*, D. Thierens, Ed., vol. 1. London, UK: ACM Press, p. 90, 2007.

[54] X. Shen, M. Zhang and T. Li, "A multi-objective optimization evolutionary algorithm addressing diversity maintenance," *International Joint Conference on Computational Sciences and Optimization*, Sanya, Hainan, pp. 524-527, 2009.

[55] M. Srinivas and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, no. 4, pp. 656-667, 1994.

[56] P. M. Murphy and D. W. Aha, "UCI repository for machine learning databases," Univ. Calif., Berkeley, CA, USA, Tech. Rep. 1026, 1994 [Online]. Available: http://www.ics.uci.edu/mlearn/MLRepository

[57] R. J. Cho, M. J. Campbell, E. A. Winzeler, L. Steinmetz, A. Conwaym, L. Wodicka, *et al.*, "A genome-wide transcriptional analysis of the mitotic cell cycle", *Molecular Cell*, vol. 2, no. 1, pp. 65-73, 1998.

[58] S. Tavazoie, D. Hughes, J. M. J. Campbell, R. J. Cho, and G. M. Church, "Systematic determination of genetic network architecture," *Nature Genetics*, vol. 22, pp. 281-285, 1999.

[59] C. T. Dang, Z. Wu, C. Deng, "An improved approach of particle swarm optimization and application in data clustering," *Intelligent Data Analysis*, vol. 19, no. 5, pp. 1049-1070, 2015.

[60] J. Sun, W. Chen, W. Fang, *et al.*, "Gene expression data analysis with the clustering method based on an improved quantum-behaved Particle Swarm Optimization," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 2, pp. 376-391, 2012.

[61] X. Yan, Y. Zhu, W. Zou, *et al.*, "A new approach for data clustering using hybrid artificial bee colony algorithm," *Neurocomputing*, vol. 97, no. 1, pp. 241-250, 2012.

[62] K. K. Bharti, P. K. Singh, "Chaotic gradient artificial bee colony for text clustering," *Soft Comput*, vol. 20, no. 3, pp. 1113-1126, 2016.

[63] H. Xiong, J. Wu and J. Chen, "*K*-means clustering versus validation measures: A data-distribution perspective," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 2, pp. 318-331, 2009.

[64] T. Calinski and J. Harabasz, "A dendrite method for cluster analysis," *Commun. Stat. Theory Methods*, vol. 3, no. 1, pp. 1-27, 1974.

[65] M. Wang, Z. Wang, Y. Chen and W. Sheng, "Observer-based fuzzy output-feedback control for discrete-time strict-feedback nonlinear systems with stochastic noises," *IEEE Transactions on Cybernetics*, vol. 50, no. 8, pp. 3766-3777, 2020.

[66] M. Wang, Z. Wang, Y. Chen and W. Sheng, "Event-based adaptive neural tracking control for discrete-time stochastic nonlinear systems: a triggering threshold compensation strategy," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 6, pp. 1968-1981, 2019.

[67] W. Liu, Z. Wang, Y. Yuan, N. Zeng, K. Hone and X. Liu, "A novel sigmoid-function-based adaptive weighted particle swarm optimizer," *IEEE Transactions on Cybernetics*, 2019, in press, DOI: 10.1109/TCYB.2019.2925015.

[68] N. Zeng, Z. Wang, W. Liu, H. Zhang, K. Hone and X. Liu, "A dynamic-neighborhood-based switching particle swarm optimization algorithm," *IEEE Transactions on Cybernetics*, 2020, in press, DOI: 10.1109/TCYB.2020.3029748.

[69] W. Liu, Z. Wang, X. Liu, N. Zeng and D. Bell, "A novel particle swarm optimization approach for patient clustering from emergency departments," I*EEE Transactions on Evolutionary Computation*, vol. 23, no. 4, pp. 632-644. 2019.

[70] N. Zeng, Z. Wang, H. Zhang, K. E. Kim, Y. Li and X. Liu, "An improved particle filter with a novel hybrid proposal distribution for quantitative analysis of gold immunochromatographic strips," *IEEE Transactions on Nanotechnology*, vol. 18, no. 1, pp. 819-829, 2019.

[71] N. Zeng, H. Li, Z. Wang, W. Liu, S. Liu, F. E. Alsaadi and X. Liu, "Deep-reinforcement-learning-based images segmentation for quantitative analysis of gold immunochromatographic strip," *Neurocomputing*, 2020, in press, https://doi.org/10.1016/j.neucom.2020.04.001.