Natural Computing (2021) 20:611–636
https://doi.org/10.1007/s11047-020-09809-z

An evaluation of k-means as a local search operator in hybrid memetic group search optimization for data clustering

Luciano D. S. Pacifico[1] · Teresa B. Ludermir[2]

Accepted: 17 September 2020 / Published online: 6 October 2020

Abstract
Cluster analysis is one important field in pattern recognition and machine learning, consisting in an attempt to distribute a set of data patterns into groups, considering only the inner properties of those data. One of the most popular techniques for data clustering is the K-Means algorithm, due to its simplicity and easy implementation. But K-Means is strongly dependent on the initial point of the search, what may lead to suboptima (local optima) solutions. In the past few decades, Evolutionary Algorithms (EAs), like Group Search Optimization (GSO), have been adapted to the context of cluster analysis, given their global search capabilities and flexibility to deal with hard optimization problems. However, given their stochastic nature, EAs may be slower to converge in comparison to traditional clustering models (like K-Means). In this work, three hybrid memetic approaches between K-Means and GSO are presented, named FMKGSO, MKGSO and TMKGSO, in such a way that the global search capabilities of GSO are combined with the fast local search performances of K-Means. The degree of influence of K-Means on the behavior of GSO method is evaluated by a set of experiments considering both real-world problems and synthetic data sets, using five clustering metrics to access how good and robust the proposed hybrid memetic models are.

Keywords Data clustering · Evolutionary algorithms · Group search optimization · K-means

1 Introduction

In the past few decades, the amount of daily produced data in electronic devices, such as smartphones, tablets, computers, cars, GPS, smart TVs, Internet of Things applications, and so on, has increased exponentially, in such a way that automatic and scalable computational systems are even more required. To extract useful information from the large data sets such systems are based on, it is impossible to rely in human analysis only, once the need for precise and reliable information in a short period of time has become mandatory (Naldi and Campello 2014).

Data clustering is one of the most important and primitive activities in pattern recognition, consisting in an important mechanism for exploratory data analysis. Clustering is characterized by an unsupervised attempt to categorize a set of data patterns in clusters, in such a way that observations belonging in a same cluster are more close related (according to their feature set) than observations from different clusters. In clustering, no prior knowledge about the data set at hand is required, so the clustering models take decisions about how to group the observations taking into consideration only the inner properties of such observations. Clustering algorithms have been successfully employed in many real-world applications, given their flexibility and adaptability, in fields such as: Agriculture (Rahamathunnisa et al. 2020), Data Mining (Sapkota et al. 2019), Ecology (Sreepathi et al. 2017), Image Understanding (Saraswathi and Allirani 2013; Wan et al. 2017; Diderot et al. 2019; Wei et al. 2019), Medicine (Li et al. 2016; Bruse et al. 2017; Premalatha and Subasree 2017; BEDDAD et al. 2019), Text Mining (Liu and Xiong 2011), Web Services (Parimalam and Sundaram 2017), Wireless

✉ Luciano D. S. Pacifico
luciano.pacifico@ufrpe.br

Teresa B. Ludermir
tbl@cin.ufpe.br

1 Departamento de Computação (DC), Universidade Federal Rural de Pernambuco (UFRPE), Recife, PE, Brazil

2 Centro de Informática (CIn) Universidade Federal de Pernambuco (UFPE), Recife, PE, Brazil

Sensors Networks (WSN) (Misra and Kumar 2016; Dhiviya et al. 2017; Masoud et al. 2019), and so on. An interesting survey on data clustering algorithms can be found in (Xu and Tian 2015).

From an optimization perspective, clustering is considered as a particular kind of NP-hard grouping problem. The most popular clustering approaches are the partitional clustering algorithms, which provide a partition of the data set into a prefixed number of clusters (an input parameter for the models). Each cluster is represented by its centroid vector, and the clustering process is driven in an effort to optimize a criterion function iteratively, by updating these cluster centroids, seeking out to improve the quality of the final solution (final partition) provided by the algorithm.

One of the most popular partitional clustering methods is the K-Means algorithm (MacQueen et al. 1967). Although traditional partitional clustering models are able to promote fast convergence speeds, these methods are known for their sensibility to the initial centroid position, what may lead to weak solutions (i.e., the model may be trapped in a local minimum point) if the algorithm starts in a poor region of the problem space. Standard partitional clustering approaches lack the mechanisms to escape from local optima points, promoting local searches on the problem space only.

Evolutionary Algorithms (EAs) have been increasingly applied to solve a great variety of difficult problems, such as hard functions optimization (He et al. 2009; Oliveira et al. 2013; Pacifico and Ludermir 2014; Jain et al. 2019), weights and architecture optimization in Neural Networks (Silva et al. 2011; Idrissi et al. 2016; Darwish et al. 2020), feature weighting and selection (Ramos and Vellasco 2018; Taşci et al. 2018; Xu et al. 2020), and so on, given their global search capability and their mechanisms to avoid local optima points. In EAs, a set (*population*) of candidate solutions (*individuals*) for the problem at hand is kept and evolved according to a generational process, seeking out the optimization of a criterion function (the *fitness function*). In EAs such as Genetic Algorithm (GA) (Holland 1992), Evolutionary Programming (EP) (Fogel et al. 1966; Fogel 2009), Evolution Strategies (ES) (Rechenberg 1973; Schwefel 1993) and Differential Evolution (DE) (Storn and Price 1995, 1997), the searching scheme is driven by operators that simulate biological processes like mutation, recombination and selection. In this context, Swarm Intelligence (SI) methods are extensions of EAs which execute their search in an attempt to simulate self-organizing collective behavior of social animals, like swarming, flocking and herding (Bonabeau et al. 1999). Examples of SI algorithms are the Ant Colony Optimization (ACO) (Dorigo et al. 1996), Particle Swarm Optimization (PSO) (Kennedy and Eberhart 1995; Kennedy et al. 2001) and Group Search Optimization (GSO) (He et al. 2006, 2009). Table 1 presents some of the most recent and popular algorithms from Evolutionary Computing literature. Interesting texts in Evolutionary Computing can be found in (Kennedy 2006; Simon 2013; Eiben and Smith 2015).

EAs have been increasingly applied in clustering applications (Hruschka et al. 2009; Inkaya et al. 2016; Canuto et al. 2018; Figueiredo et al. 2019), but, given their stochastic nature, such techniques may be too slow to converge in comparison to standard partitional clustering algorithms. It is also known that, in general, evolutionary algorithms local search mechanisms are not very much effective (Ren et al. 2014).

To overcome this drawback, Hybrid Intelligence Systems (HISs) and Memetic Algorithms (MA) combining evolutionary approaches and K-Means have been proposed in literature, such that global searches performed by EAs are complemented by local searches performed by K-Means. Some examples of hybrid evolutionary systems that use K-Means as a local searcher are found in (Ahmadyfard and Modares 2008; Abdel-Kader 2010; Bhavani et al. 2011; Pacifico and Ludermir 2018, 2019). But most works in data clustering literature only employ K-Means after the generational process of the selected EA is concluded, which means that the best solution found by the EA may already be trapped in a local minimum point, making the application of K-Means sub-optimal (it will only exploit the problem region that contains a local minimum). To avoid such problem, in this work three hybrid memetic algorithms are proposed, named FMKGSO, MKGSO and TMKGSO, which combine the global search capabilities of GSO with local search speed provided by K-Means, which use K-Means (in different ways) to improve the population of GSO during the generational process. GSO was proposed as a method for continuous optimization problems based on the behavior of social animals attempting to find food resources, and, in the past few years, GSO has been successfully applied in many real world applications (Chen et al. 2014b; Krishnaprabha and Aloor 2014; Li et al. 2015; Pacifico et al. 2018), presenting competitive optimization performances in comparison to other evolutionary algorithms, such as GA, PSO, EP and ES.

The proposed approaches are implemented in an attempt to comprehend the influence of K-Means on the behavior of GSO when dealing with data clustering task, and, for that, a testing bed consisting of 20 (twenty) real-world problems (obtained from UCI Machine Learning Repository - Asuncion and Newman 2007) and 10 synthetic data sets is proposed. Five evaluation metrics from clustering literature are employed to validate the quality of the solutions achieved by the proposed models, and five partitional evolutionary algorithms are selected for comparison purposes.

Table 1 List of some natural-inspired population-based metaheuristics

Algorithm	Acronym	Main reference(s)
Genetic Algorithm	GA	(Holland 1992)
Evolutionary Programming	EP	(Fogel et al. 1966; Fogel 2006, 2009)
Evolution Strategies	ES	(Rechenberg 1973; Schwefel 1993)
Genetic Programming	GP	(Koza and Koza 1992)
Differential Evolution	DE	(Storn and Price 1995, 1997)
Ant Colony Optimization	ACO	(Dorigo et al. 1996)
Particle Swarm Optimization	PSO	(Kennedy and Eberhart 1995; Kennedy et al. 2001)
Harmony Search	HS	(Z.W et al. 2001; Geem 2010)
Artificial Fish Swarm Algorithm	AFSA	(Li 2002)
Bacterial Foraging Optimization Algorithm	BFOA	(Passino 2002)
Group Search Optimization	GSO	(He et al. 2006, 2009)
Artificial Bee Colony	ABC	(Karaboga and Basturk 2007)
Gravitational Search Algorithm	GSA	(Rashedi et al. 2009)
Firefly Algorithm	FFA	(Yang 2009a, b)
Backtracking Search Optimization	BSA	(Civicioglu 2013)
Grey Wolf Optimizer	GWO	(Mirjalili et al. 2014)
Sine-Cosine Algorithm	SCA	(Mirjalili 2016)
Whale Optimization Algorithm	WOA	(Mirjalili and Lewis 2016)
Grasshopper Optimization Algorithm	GOA	(Saremi et al. 2017)
Squirrel Search Algorithm	SSA	(Jain et al. 2019)

This work is organized as follows. Firstly, some introductory concepts are introduced (Sect. 2), like K-Means (Sect. 2.1), partitional evolutionary algorithms (Sect. 2.2) and GSO (Sect. 2.3). After that, the proposed hybrid memetic GSO approaches are explained (Sect. 3), followed by the experimental evaluation (Sect. 4). Finally, some conclusions and interesting trends for future works are presented (Sect. 5).

2 Preliminaries

The baseline models adopted in this work are presented in the following sections (Sects. 2.1, 2.2 and 2.3).

2.1 K-Means

K-Means (MacQueen et al. 1967) has been proposed as a partitional clustering algorithm for continuous data, which groups real-valued data vectors into a predefined number of clusters. Consider a partition P_C of a data set with N data patterns (each data pattern is represented by a vector $\mathbf{x}_j \in \Re^m$, where $j = 1, 2, \ldots, N$) in C clusters, where C is a required input parameter for the algorithm. Each cluster is represented by its centroid vector $\mathbf{g}_c \in \Re^m$ (where $c = 1, 2, \ldots, C$).

In K-Means, clusters are formed based on a dissimilarity measure, the Euclidean distance [Eq. (1)]. For each iteration (until a maximum number of iterations $t_{maxkmeans}$ is reached or another stopping criterion is satisfied), a new cluster centroid vector is calculated, for each cluster, as the mean of its current data vectors (i.e., the data patterns currently assigned to the cluster). After that, the new partition is formed, and each pattern is associated to the cluster with the nearest centroid.

$$d(\mathbf{x}_j, \mathbf{g}_c) = \sqrt{\sum_{k=1}^{m} (x_{jk} - g_{ck})^2} \tag{1}$$

where

$$\mathbf{g}_c = \frac{1}{N_c} \sum_{\forall \mathbf{x}_l \in c} \mathbf{x}_l \tag{2}$$

where N_c is the number of patterns associated to cluster c. The criterion function for K-Means is the Within-Cluster Sum of Squares, given in Eq. (3).

$$J(P_C) = \sum_{c=1}^{C} \sum_{\forall \mathbf{x}_j \in c} d(\mathbf{x}_j, \mathbf{g}_c) \tag{3}$$

K-Means algorithm is presented in Algorithm 1.

Algorithm 1 K-Means

```
$t \leftarrow 0$
**Initialization**: Pick $C$ patterns randomly as the initial cluster centroids $\mathbf{g}_c$ $(c = 1, \ldots, C)$. After that, assign each pattern $\mathbf{x}_j$ to its closest cluster;
**Calculate** the initial criterion value $J(P_C^0)$ (eq.(3)).
**while** $(t < t_{maxkmeans})$ **do**
    **New Centroids Determination**: for each cluster $c$, update its centroid $\mathbf{g}_c$ using eq.(2).
    **New Partition Determination**: for each data pattern $\mathbf{x}_j$, assign it to the cluster with the nearest centroid $\mathbf{g}_c$.
    **Calculate** the new criterion value $J(P_C^t)$ (eq.(3)).
    $t \leftarrow t + 1$
**end while**
**Return** the final partition $P_C^{t_{maxkmeans}}$
```

2.2 Evolutionary algorithms for data clustering

In this section, we explain the most commonly adopted representation schema for evolutionary algorithms, when adapted as partitional clustering methods (Chen and Ye 2004).

Once more, consider a partition P_C of a data set with N patterns $\mathbf{x}_j \in \Re^m$ $(j = 1, 2, \ldots, N)$ in C clusters. Each cluster is represented by its centroid vector $\mathbf{g}_c \in \Re^m$ $(c = 1, 2, \ldots, C)$. Each population individual $\mathbf{X}_i \in \Re^n$ (where $n = m \times C$) in population G represents C cluster centroids at the same time, one for each cluster (Chen and Ye 2004). For instance, if $m = 3$ and $C = 5$, each individual will be a vector $\mathbf{X}_i \in \Re$, where the first three features will represent the centroid $\mathbf{g}_1$, features 4-th to 6-th will represent the centroid $\mathbf{g}_2$, features 7-th to 9-th will represent the centroid $\mathbf{g}_4$, features 10-th to 12-th will represent the centroid $\mathbf{g}_4$, and the last three features (features 13-th to 15-th) will represent the centroid $\mathbf{g}_5$, just like in Fig. 1.

The population of evolutionary algorithms is generally initialized by a random process, but in the context of partitional data clustering, an initialization by the random choice of C patterns from the data set in analysis to compose the initial cluster centroids (just like in K-Means - see Sect. 2.1), for each individual, leads to a faster exploration of the problem search space.

As the fitness function, many works adopt the Within-Cluster Sum of Squares [Eq. (3)] or some alternative function that takes such criterion as its main component, just like in (Chen and Ye 2004; Ahmadyfard and Modares 2008; Hruschka et al. 2009; Prabha and Visalakshi 2014; Pacifico and Ludermir 2019, 2019b). But sometimes, different functions are adopted as the fitness function (Das et al. 2007; Liu et al. 2011; Wong et al. 2011; He and Tan 2012; José-García and Gómez-Flores 2016; Pacifico and Ludermir 2016).

Once the initial population is obtained and the fitness value for each individual $\mathbf{X}_i^{(0)}$ in population G is computed, the evolutionary operators for the selected evolutionary algorithm are applied to the population, evolving the cluster centroids represented by each individual through a generational process, until a termination condition is reached. The global best individual found by the EA is provided as the clustering solution. A generic evolutionary algorithm for partitional data clustering is presented in Algorithm 2.

Algorithm 2 Generic Partitional Evolutionary Algorithm

```
$t \leftarrow 0$.
**Initialize** each individual $\mathbf{X}_i^{(0)} \in G^{(0)}$ by randomly picking $C$ patterns from the current data set as its initial cluster centroids.
**Generate** the initial partition $\mathbf{X}_i^{(0)}.P_C^{(0)}$, assigning each pattern $\mathbf{x}_j$ to its closest cluster, for each individual $\mathbf{X}_i^{(0)}$.
**Calculate** the fitness function for each individual $\mathbf{X}_i^{(0)}$.
**while** (termination conditions are not met) **do**
    **Execute** all evolutionary operators, according to the selected evolutionary algorithm, on current population $G^t$.
    **Assign** each pattern $\mathbf{x}_j$ to its closest cluster in $\mathbf{X}_i^t.P_C^t$, for each $\mathbf{X}_i^t$.
    **Calculate** the new fitness value for each population individual $\mathbf{X}_i^t \in G^t$.
    $t \leftarrow t + 1$.
**end while**
**Return** $\mathbf{X}_{best}^{t_{max}}$.
```

Although EAs are able to find global solutions even when dealing with complex optimization problems, the searching process performed by such strategies may be to slow, and in many applications (just like in Cui et al. 2005; Abdel-Kader 2010; Ahmadi et al. 2010; Chen et al. 2014a; Pacifico and Ludermir 2018, 2019), the EAs are hybridized with K-Means in such a way that the EAs are used to find a

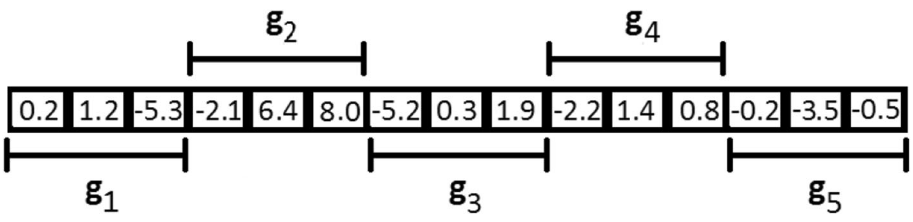

Fig. 1 Individual representation: $\mathbf{g}_1$, $\mathbf{g}_2$, $\mathbf{g}_3$, $\mathbf{g}_4$ and $\mathbf{g}_5$ are 3-dimensional cluster centroids

better set of starting points (i.e., cluster centroids) to K-Means. Algorithm 3 illustrates such process.

Algorithm 3 Hybrid Generic EA-k-means Algorithm

```
$t \leftarrow 0$.
Initialize each individual $\mathbf{X}_i^{(0)} \in G^{(0)}$ by randomly picking $C$ patterns from the current data set as its initial cluster centroids.
Generate the initial partition $\mathbf{X}_i^{(0)}.P_C^{(0)}$, assigning each pattern $\mathbf{x}_j$ to its closest cluster, for each individual $\mathbf{X}_i^{(0)}$.
Calculate the fitness function for each individual $\mathbf{X}_i^{(0)}$.
while (termination conditions are not met) do
    Execute all evolutionary operators, according to the selected evolutionary algorithm, on current population $G^t$.
    Assign each pattern $\mathbf{x}_j$ to its closest cluster in $\mathbf{X}_i^t.P_C^t$, for each $\mathbf{X}_i^t$.
    Calculate the new fitness value for each population individual $\mathbf{X}_i^t \in G^t$.
    $t \leftarrow t+1$.
end while
Use the best individual found by the generational process $\mathbf{X}_{best}^{t_{max}}$ as the input set o cluster centroids to K-Means algorithm.
Execute K-Means on $\mathbf{X}_{best}^{t_{max}}$ for $t_{maxkmeans}$ iterations.
Return the best set of cluster centroids found by K-Means $\mathbf{X}_{best}^{new}$.
```

2.3 Group search optimization

Group search optimization is inspired by animal social searching behavior and group living theory. GSO employs the Producer-Scrounger (PS) model as a framework. The PS model was firstly proposed by Barnard and Sibly (1981) to analyze social foraging strategies of group living animals. PS model assumes that there are two foraging strategies within groups: *producing* (*e.g.*, searching for food); and joining (*scrounging*, *e.g.*, joining resources uncovered by others). Foragers are assumed to use producing or joining strategies exclusively. Under this framework, concepts of resource searching from animal visual scanning mechanism are used to design optimum searching strategies in GSO algorithm (He et al. 2009).

In GSO, the population G of S individuals is called *group*, and each individual is called a *member*. In a n-dimensional search space, the i-th member at the t-th searching iteration (*generation*) has a current position $\mathbf{X}_i^t \in \Re^n$ and a head angle $\alpha_i^t \in \Re^{n-}$. The search direction of the i-th member, which is a vector $\mathbf{D}_i^t(\alpha_i^t) = (d_{i1}^t, \ldots, d_{in}^t)$ can be calculated from α_i^t via a polar to Cartesian coordinate transformation:

$$
\begin{aligned}
d_{i1}^t &= \prod_{q=1}^{n-1} \cos(\alpha_{iq}^t),\\
d_{ij}^t &= \sin(\alpha_{i(j-1)}^t) \prod_{q=1}^{n-1} \cos(\alpha_{iq}^t) \quad (j = 1, \ldots, n-1),\\
d_{in}^t &= \sin(\alpha_{i(n-1)}^t)
\end{aligned}
\tag{4}
$$

A group in GSO consists of three types of members: producers, scroungers and dispersed members (or *rangers*) (He et al. 2009). The rangers are introduced by GSO model, extending standard PS framework.

During each GSO search iteration, a group member which has found the best fitness value so far (most promising area form the problem search space) is chosen as the producer ($\mathbf{X}_p$) (Couzin et al. 2005), and the remaining members are scroungers or rangers. Standard GSO admits only one producer in each iteration, but there are some GSO variants that use multiple producers at the same time (Junaed et al. 2013; Pacifico and Ludermir 2013).

The producer employs a scanning strategy (*producing*) based on its vision field, generalized to a n-dimensional space, which is characterized by maximum pursuit angle $\theta_{max} \in \Re^{n-}$ and maximum pursuit distance $l_{max} \in \Re$, given by Eq. (5).

$$l_{max} = \|\mathbf{U} - \mathbf{L}\| = \sqrt{\sum_{k=1}^{n} (U_k - L_k)^2} \tag{5}$$

where U_k and L_k denote the upper bound and lower bound of the k-th dimension from the problem space, respectively.

In GSO, at the t-th iteration the producer $\mathbf{X}_p^t$ will scan laterally by randomly sampling three points in the scanning field: one at zero degree [Eq. (6)], one in the right hand side hypercube [Eq. (7)] and one in the left hand side hypercube [Eq. (8)].

$$\mathbf{X}_z = \mathbf{X}_p^t + r_1 l_{max} \mathbf{D}_p^t(\alpha_p^t) \tag{6}$$

$$\mathbf{X}_r = \mathbf{X}_p^t + r_1 l_{max} \mathbf{D}_p^t(\alpha_p^t + \frac{\mathbf{r}_2 \theta_{max}}{2}) \tag{7}$$

$$\mathbf{X}_l = \mathbf{X}_p^t + r_1 l_{max} \mathbf{D}_p^t(\alpha_p^t - \frac{\mathbf{r}_2 \theta_{max}}{2}) \tag{8}$$

where $r_1 \in \Re$ is a normally distributed random number (mean 0 and standard deviation 1) and $\mathbf{r}_2 \in \Re^{n-}$ is a uniformly distributed random sequence in the range (0, 1).

If the producer is able to find a better resource than its current position, it will fly to this point; if no better point is found, the producer will stay in its current position, then it will turn its head to a new generated angle [Eq. (9)].

$$\alpha_p^{t+1} = \alpha_p^t + \mathbf{r}_2 \alpha_{max} \tag{9}$$

where $\alpha_{max} \in \Re$ is the maximum turning angle.

If after $a \in \Re$ iterations the producer cannot find a better area, it will turn its head back to zero degree [Eq. (10)].

$$\alpha_p^{k+a} = \alpha_p^k \tag{10}$$

All scroungers will join the resource found by the producer, performing *scrounging* strategy according to Eq. (11).

$$\mathbf{X}_i^{t+1} = \mathbf{X}_i^t + \mathbf{r}_3 \circ (\mathbf{X}_p^t - \mathbf{X}_i^t) \tag{11}$$

where $\mathbf{r}_3 \in \Re^n$ is a uniform random sequence in the range (0, 1) and $\circ$ is the Hadamard product or the Schur product, which calculates the entrywise product of two vectors.

The rangers will perform random walks through the problem space (Higgins and Strauss 2004), according to Eq. (12).

$$\mathbf{X}_i^{t+1} = \mathbf{X}_i^t + l_i \mathbf{D}_i^t(\alpha_i^{t+1}) \tag{12}$$

where

$$l_i = a r_1 l_{max} \tag{13}$$

In GSO, when a member escapes from the search space bounds, it will turn back to its previous position inside the search space (Dixon 1959). Some studies considering alternative treatments to deal with out-bounded population individuals can be found in (Xu and Shu 2006; Silva et al. 2011; Pacifico et al. 2018).

GSO algorithm is presented in Algorithm 4.

Algorithm 4 GSO

```
$t \leftarrow 0$.
Initialize randomly position $\mathbf{X}_i^{(0)}$ and head angles $\alpha_i^{(0)}$ of all members $\mathbf{X}_i^{(0)} \in G$.
Calculate the fitness value ($fit(\mathbf{X}_i^{(0)})$) for each member $\mathbf{X}_i^{(0)}$.
while (termination conditions are not met) do
    Pick the best group member as the $\mathbf{X}_p^t$ for the current iteration.
    Execute producing ($\mathbf{X}_p^t$ only) by evaluating three random points in its visual scanning field, $\mathbf{X}_z^t$ (eq. (6)), $\mathbf{X}_r^t$ (eq. (7)) and $\mathbf{X}_l^t$ (eq. (8)).
    Choose a percentage from the members (but the $\mathbf{X}_p^t$) to perform scrounging (eq. (11)).
    Ranging: The remaining members will perform ranging through random walks (eq. (12)).
    Calculate the new fitness value $fit(\mathbf{X}_i^t)$ for each group member $\mathbf{X}_i^t$.
    $t \leftarrow t + 1$.
end while
Return $\mathbf{X}_p^{t_{max}}$.
```

GSO scrounging operator focuses the search performed by the group in the most promising areas from the problem space, corresponding to the main exploration/exploitation strategy employed by many EAs (like crossover strategy in Genetic Algorithms and particle movement in Particle Swarm Optimization).

Producing and ranging are the main mechanisms employed by GSO for escaping local minima points. When the producer of one generation is trapped in a local minimum point (situation in which all scroungers would be trapped by following the producer, resulting in a premature convergence of the group), producing operator may find a better point in the search space, escaping from that local minimum. Even if that situation does not happen so easily, rangers will keep performing random walks that do not depend on the results found by the producer, which may lead to most promising areas than the ones found so far by the whole group, evading local minima points.

Some works have already adapted GSO to the context of partitional clustering, just like in (Pacifico and Ludermir 2014a, 2016, 2019b).

3 Proposed approaches

This section presents the proposed hybrid memetic GSO and K-Means models, elaborated to test the influence of K-Means when employed as a local searcher to improve the capabilities of partitional Group Search Optimizer. Three memetic algorithms are presented: FMKGSO, MKGSO and TMKGSO. In this work, we opt to combine GSO and K-Means due to some advantages promoted by these algorithms, such as:

1. GSO has been proven to be a good global optimizer in many real-world applications;
2. GSO presents good mechanisms to escape from local optima points, represented by the *producing* behavior of producers, and by the *ranging* behavior from dispersed members;
3. *Ranging* and *producing* behaviors are also employed to prevent the premature convergence of the group, that is a known and hard to deal problem that affects some global search natural-inspired metaheuristics, such as Particle Swarm Optimization;
4. Standard GSO implements an effective mechanism to prevent out-bounded members, avoiding individuals that would not codify an actual solution to the problem at hand, by extrapolating the problem search space boundaries;
5. K-Means is a good local searcher, performing fast exploitation on problem space regions.

The proposed models are described as follows.

Firstly, as in generic partitional framework for EAs (see Sect. 2.2), for all three memetic algorithms, each group member is represented as a continuous vector $\mathbf{X}_i \in \Re^n$ (with $n = m \times C$), codifying C cluster centroids at the same time, and at the t-th generation, $\mathbf{X}_i$ will determine its own partition $\mathbf{X}_i^t.P_C$.

For each algorithm, the initial population will be obtained by the random selection of C patterns from the data set being analyzed to compose each member $\mathbf{X}_i^{(0)}$. After random initialization, each member $\mathbf{X}_i^{(0)}$ has its fitness value computed to determine the quality of the partition $\mathbf{X}_i^{(0)}.P_C$ it represents, by associating each data pattern $\mathbf{x}_j$ to its closest centroid vector $\mathbf{X}_i^{(0)}.\mathbf{g}_c$ in $\mathbf{X}_i^{(0)}$.

After that, GSO generational process is initialized. Each memetic algorithm will employ K-Means for some degree during its generational process to improve the quality of the group.

As in standard GSO, for all three memetic approaches, the generational process starts, in each generation t, by the choice of the best member found so far (according to the selected fitness function) as the producer $\mathbf{X}_p^t$. The producer will perform *producing* operator as in standard GSO, by the evaluation of three random points from its visual scanning field [see Eqs. (6), (7) and (8)]. If a better point is found, the $\mathbf{X}_p^t$ will migrate to that point in the next generation.

The *scrounging* and *ranging* for the memetic approaches will be executed following Eqs. (11) and (12), respectively. After executing all standard GSO operators, each memetic model will adopt K-Means to improve GSO group by different ways, as described bellow.

In FMKGSO, the algorithm keeps track on the quality of each member separately. For each new member $\mathbf{X}_i^{t+1}$ in the t-th generation (after the execution of *producing*, *scrounging* and *ranging* operators), determine its partition $\mathbf{X}_i^{t+1}.P_C$ by associating each data pattern $\mathbf{x}_j$ to its closest centroid vector $\mathbf{X}_i^{t+1}.\mathbf{g}_c$. Compute $\mathbf{X}_i^{t+1}$ fitness. If $\mathbf{X}_i^{t+1}$ has not been able to improve (in terms of the fitness value) by a predefined number of consecutive generations ($t_{maxkmeans}$), its cluster centroids are refined by the application of $t_{maxkmeans}$ K-Means steps. FMKGSO will only apply K-Means to the members that are unable to improve their partition for a period of time. This combination seeks out to speed up the search performed by each member, once GSO operators could present slow convergence rates, just like in all EAs. K-Means execution can speed up considerably the convergence rates of GSO members, promoting a better exploration and exploitation of the problem search space. The generational process executes until a termination condition is met, and the final best improved solution $\mathbf{X}_p^{t_{max}}$ is returned. FMKGSO algorithm is presented in Algorithm 5.

Algorithm 5 FMKGSO

```
$t \leftarrow 0$
**Initialization:** For each member $\mathbf{X}_i{}^{(0)}$, pick $C$ patterns randomly as the initial cluster centroids $\mathbf{g}_c$. After that, assign each pattern $\mathbf{x}_j$ to its closest cluster.
**Calculate** the initial fitness function for each member $\mathbf{X}_i^{(0)}$.
**while** (termination conditions are not met) **do**
    **Pick** the best group member as the $\mathbf{X}_p^t$ for the current iteration.
    **Execute** *producing* ($\mathbf{X}_p^t$ only). For each evaluated point ($\mathbf{X}_z^t, \mathbf{X}_r^t$ and $\mathbf{X}_l^t$), determine its partition by assigning each data pattern to the cluster with the nearest centroid.
    **Choose** a percentage from the members (but the $\mathbf{X}_p^t$) to perform *scrounging*.
    **Ranging**: The remaining members will perform *ranging* through random walks.
    **Determine** the new partitions represented by each member $\mathbf{X}_i^{t+1}$, by assigning each data pattern to the cluster with the nearest centroid, for each member.
    **Calculate** the new fitness value for each group member.
    **for** each $\mathbf{X}_i^{t+1}$ **do**
        **if** $\mathbf{X}_i^{t+1}$ has not improved for the last $t_{maxkmeans}$ generations **then**
            **Execute** K-Means for $t_{maxkmeans}$ steps to refine $\mathbf{X}_i^{t+1}$.
        **end if**
    **end for**
    $t := t + 1$
**end while**
**Return** improved $\mathbf{X}_p^{t_{max}}$.
```

In MKGSO, after the determination of the new group by the execution of *producing*, *scrounging* and *ranging* operators, each member $\mathbf{X}_i^{t+1}$ will be improved by the execution of exactly one K-Means step in each generation before its fitness evaluation. All members are refined that way, independently if GSO operators have been able to improve or not. This operation seeks out to speedup the exploitation performed by each GSO member, by aggregating local information into GSO generational process smoothly. After that, the partitions represented by each member will be determined, and their fitness value will be computed. The generational process executes until a termination condition is met, and the final best improved solution $\mathbf{X}_p^{t_{max}}$ is returned. MKGSO algorithm is presented in Algorithm 6.

Algorithm 6 MKGSO

```
$t \leftarrow 0$
**Initialization:** For each member ${\mathbf{X}_i}^{(0)}$, pick $C$ patterns randomly as the initial cluster centroids $\mathbf{g}_c$. After that, assign each pattern $\mathbf{x}_j$ to its closest cluster.
**Calculate** the initial fitness function for each member $\mathbf{X}_i^{(0)}$.
**while** (termination conditions are not met) **do**
    **Pick** the best group member as the $\mathbf{X}_p^t$ for the current iteration.
    **Execute** *producing* ($\mathbf{X}_p^t$ only). For each evaluated point ($\mathbf{X}_z^t, \mathbf{X}_r^t$ and $\mathbf{X}_l^t$), determine its partition by assigning each data pattern to the cluster with the nearest centroid.
    **Choose** a percentage from the members (but the $\mathbf{X}_p^t$) to perform *scrounging*.
    **Ranging**: The remaining members will perform *ranging* through random walks.
    **Determine** the new partitions represented by each member $\mathbf{X}_i^t$, by assigning each data pattern to the cluster with the nearest centroid, for each member.
    **Execute** one K-Means step to refine each group member.
    **Calculate** the new fitness value for each group member.
    $t := t + 1$
**end while**
**Return** improved $\mathbf{X}_p^{t_{max}}$
```

The last proposed approach, TMKGSO, will apply K-Means to each member after a period, for a limited number of steps. After a prefixed number of generations ($t_{maxkmeans}$, an input parameter for the algorithm), each new group member $\mathbf{X}_i^{t+1}$ (obtained by the application of *producing*, *scrounging* or *ranging* operator) is refined by the application of $t_{maxkmeans}$ K-Means steps. By alternating some executions of GSO and K-Means, the new groups are improved by the interchanging of global and local information, in such a way that when GSO is executing, its operators will prevent the group from getting trapped in local optma regions, promoting the global exploration of the problem space, while during K-Means steps, the group will be guided through fast local exploitations on the region of the problem space where each member is placed. Each algorithm will complement the search performed by the other. The generational process executes until a termination condition is met, and the final best improved solution $\mathbf{X}\,_p^{t_{max}}$ is returned. TMKGSO algorithm is presented in Algorithm 7.

Algorithm 7 TMKGSO

```
$t \leftarrow 0$
**Initialization:** For each member ${\mathbf{X}_i}^{(0)}$, pick $C$ patterns randomly as the initial cluster centroids $\mathbf{g}_c$. After that, assign each pattern $\mathbf{x}_j$ to its closest cluster.
**Calculate** the initial fitness function for each member $\mathbf{X}_i^{(0)}$.
**while** (termination conditions are not met) **do**
    **Pick** the best group member as the $\mathbf{X}_p^t$ for the current iteration.
    **Execute** *producing* ($\mathbf{X}_p^t$ only). For each evaluated point ($\mathbf{X}_z^t, \mathbf{X}_r^t$ and $\mathbf{X}_l^t$), determine its partition by assigning each data pattern to the cluster with the nearest centroid.
    **Choose** a percentage from the members (but the $\mathbf{X}_p^t$) to perform *scrounging*.
    **Ranging**: The remaining members will perform *ranging* through random walks.
    **Determine** the new partitions represented by each member $\mathbf{X}_i^t$, by assigning each data pattern to the cluster with the nearest centroid, for each member.
    **if** t $mod\ t_{maxkmeans} = 0$ **then**
        **Execute** K-Means for $t_{maxkmeans}$ steps to refine each group member.
    **end if**
    **Calculate** the new fitness value for each group member.
    $t := t + 1$
**end while**
**Return** improved $\mathbf{X}_p^{t_{max}}$
```

By employing three different strategies to use K-Means during GSO generational process, we can evaluate the actual influence of that technique as a local searcher. In that sense, in this work the total number of intended clusters C is supposed to be known *a priori* (Niu et al. 2017), so no other factors would influence on the behavior of the proposed models. The automatic determination of the optimal number of cluster for each problem is ongoing research, and that issue will be addressed in a future work. For more information on automatic clustering issue, please refer to Das et al. (2007), José-García and Gómez-Flores (2016), Elaziz et al. (2019).

4 Experimental results

In this section, we test the clustering capabilities of the proposed approaches, in comparison to other partitional evolutionary algorithms from literature, by means of 20 (twenty) real-world and 10 synthetic data sets. All real-world data sets are benchmark classification and clustering problems acquired from UCI Machine Learning Repository (Asuncion and Newman 2007). The selected real data set features are shown in Table 2, presenting different degrees of difficulties, such as unbalanced and overlapping classes, different number of classes and features, and so on.

Table 2 Real-world data set features

Data set	Inst.	Attr.	Classes
Abalone	4177	8	3
Banknote authentication	1372	4	2
Blood transfusion	748	4	2
Cancer	699	9	2
Diabetes	768	8	2
E. coli	336	7	8
Glass	214	9	6
Heart	270	13	2
Image segmentation	2310	18	7
Ionosphere	351	34	2
Iris	150	4	3
Landsat satellite images	6435	36	7
Letter recognition	20000	16	26
Optical recognition	5620	64	10
Page blocks classification	5473	10	5
Pen-based recognition of handwritten digits (Pen-digits)	10992	16	10
Seeds	210	7	3
Waveform	5000	21	3
Wine	178	13	3
Yeast	1484	8	10

The proposed synthetic data sets are split into to main groups: *Disp* group (containing five data sets), where all classes varies in relation to its degree of dispersion, from a well-separated scenario (Disp01) to a scenario where all classes present some degree of overlapping; *Prox* group (containing five data sets), where the overlapping is obtained by the approximation of class centers, from a scenario where all class centers are well-split into the problem search space (Prox01), to a scenario where all class centers are close to each other (Prox05). The configurations for the proposed synthetic data sets are presented in Table 3 and illustrated in Fig. 2.

For comparison purposes, five clustering measures are employed: the Within-Cluster Sum of Squares [Eq. (3)], the Intra-Cluster Distance [D_{max}, Eq. (14)] and the Inter-Cluster Separation [D_{min}, Eq. (15)] (Wong et al. 2011), the Weighted Quantization Error (J_{e_2}, Eq. 16) (Esmin et al. 2008) and the Corrected Rand Index [CR, Eq. (17)] (Hubert and Arabie 1985).

$$D_{max}(P_C) = \max_{c=1,\ldots,C}\{\sum_{\forall \mathbf{x}_j \in c} d(\mathbf{x}_j, \mathbf{g}_c)/|N_c|\} \tag{14}$$

$$D_{min}(P_C) = \min_{\forall c_1, c_2, c_1 \neq c_2}\{d(\mathbf{g}_{c_1}, \mathbf{g}_{c_2})\} \tag{15}$$

$$J_{e_2}(P_C) = \sum_{c=1}^{C}[(\sum_{\forall \mathbf{x}_j \in c} d(\mathbf{p}_j, \mathbf{g}_c)/|N_c|) \times (|N_c|/N)] \tag{16}$$

The *CR* assesses the degree of similarity between an *a priori* partition and a partition provided by the clustering algorithm. Given that all data sets adopted in this work are real-valued classification problems (i.e., all data patters are labeled in each data set), *CR* represents a robust comparison metric for clustering studies, since its analysis takes into consideration only the relationships among the data patterns from an *a priori* and an *a posteriori* partitions, without taking into consideration the categories themselves.

Considering a partition $U_R = \{u_1, u_2, \ldots, u_R\}$ provided by a clustering algorithm and an *a priori* partition defined by classification $V_C = \{v_1, v_2, \ldots, v_C\}$, *CR* is defined as by Eq. (17).

$$CR = \frac{a-b}{c-d} \tag{17}$$

where

$$a = \sum_{i=1}^{R}\sum_{j=1}^{C}\binom{n_{ij}}{2} \tag{18}$$

$$b = \binom{n}{2}^{-1}\sum_{i=1}^{R}\binom{n_{i.}}{2} \tag{19}$$

$$c = 1/2\left[\sum_{i=1}^{R}\binom{n_{i.}}{2} + \sum_{j=1}^{C}\binom{n_{.j}}{2}\right] \tag{20}$$

Table 3 Disp and Prox synthetic data sets description: each Disp and Prox data set consists of 1000 3-dimensional patterns, equally distributed among 5 classes

Data set	Classes	Instances per class	Total no. of instances	Attributes	μ_k (1 to 5)	Σ_k (1 to 5)
Disp01	5	200	1000	3	(− 10, − 5, 0, 5, 10)	(1, 1, 1, 1, 1)
Disp02	5	200	1000	3	(− 10, − 5, 0, 5, 10)	(4, 1, 1, 1, 4)
Disp03	5	200	1000	3	(− 10, − 5, 0, 5, 10)	(4, 1, 4, 1, 4)
Disp04	5	200	1000	3	(− 10, − 5, 0, 5, 10)	(4, 4, 1, 4, 4)
Disp05	5	200	1000	3	(− 10, − 5, 0, 5, 10)	(4, 4, 4, 4, 4)
Prox01	5	200	1000	3	(− 8, − 4, 0, 4, 8)	(1, 1, 1, 1, 1)
Prox02	5	200	1000	3	(− 8, − 6, 0, 4, 8)	(1, 1, 1, 1, 1)
Prox03	5	200	1000	3	(− 8, − 6, 0, 6, 8)	(1, 1, 1, 1, 1)
Prox04	5	200	1000	3	(− 8, − 6, − 4, 6, 8)	(1, 1, 1, 1, 1)
Prox05	5	200	1000	3	(− 8, − 6, − 4, − 2, 0)	(1, 1, 1, 1, 1)

The mean μ and covariance matrix Σ are the same for each attribute in each class, but varies from class to class (the values are presented from class 1–5, respectively)

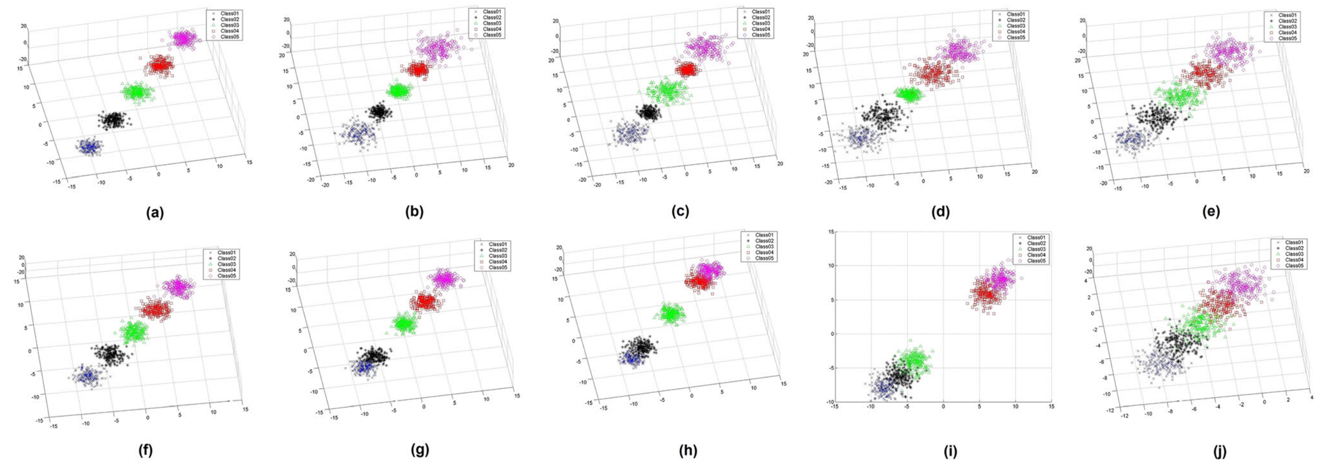

Fig. 2 Synthetic data sets representation: **a**–**e** Disp01 to Disp05, respectively, **f**–**j** Prox01-Prox05, respectively

$$d = \binom{n}{2}^{-1} \sum_{i=1}^{R} \binom{n_{i.}}{2} \sum_{j=1}^{C} \binom{n_{.j}}{2} \qquad (21)$$

where n_{ij} represents the number of objects that are in clusters u_i and v_i; $n_{i.}$ indicates the number of objects in cluster u_i; $n_{.j}$ indicates the number of objects in cluster v_j; and n is the total number of objects. CR takes its values from the interval [− 1,1], in which the value 1 indicates perfect agreement between partitions, whereas values near 0 (or negatives) correspond to cluster agreement found by chance (Arabie et al. 1996).

The evaluation criterion includes a rank system employed through the application of Friedman hypothesis test (Friedman 1937, 1940) for all the comparison clustering measures. The Friedman test is a non-parametric hypothesis test that ranks all algorithms for each data set separately. If the null-hypothesis (all ranks are not significantly different) is rejected, Nemenyi test (Nemenyi 1962) is adopted as the *post-hoc* test. According to Nemenyi test, the performance of two algorithms are considered significantly different if the corresponding average ranks differ by at least the critical difference

$$CD = q_a \sqrt{\frac{n_{alg}(n_{alg}+1)}{6n_{data}}} \qquad (22)$$

where n_{data} represents the number of data sets, n_{alg} represents the number of compared algorithms and q_a are critical values based on a Studentized range statistic divided by $\sqrt{2}$ (Demšar 2006). Since J, J_{e_2} and D_{max} are *minimization metrics* (indicated by ↓), the best methods will obtain lower ranks for the Friedman-Nemenyi test, while for D_{min} and CR (*maximization metrics*, indicated by ↑), the best methods will find higher average ranks for the Friedman-Nemenyi hypothesis test.

The proposed hybrid memetic GSO approaches are compared to five other EAs: Genetic Algorithm (GA), Differential Evolution (DE), Particle Swarm Optimization (PSO), standard Group Search Optimization and Backtracking Search Optimization (BSA). The selected approaches are state-of-the-art models from evolutionary computing and data clustering literature, being successfully applied in many applications recently: GA (Shi and Xu 2018; Akbari et al. 2019; Islam et al. 2019; Mortezanezhad and Daneshifar 2019; Toman et al. 2020), DE (Wang 2018; Li and Dong 2019; Cho and Nyunt 2020; Zhang and Cao 2020; Zhu et al. 2020), PSO (Souza et al. 2018; Wang et al. 2018; Li et al. 2019; Pacifico and Ludermir 2019; Miranda and Prudêncio 208), GSO (Pacifico and Ludermir 2018; Lin and Huang 2019; Pacifico and Ludermir 2019b; Taj and Basu 2019; Abualigah 2020), and BSA (Latiff et al. 2016; Günen et al. 2017; Li et al. 2019; Toz et al. 2019; Hassan and Rashid 2020).

All selected algorithms have been adapted as hybrid partitional clustering models according to the schema presented in Algorithm 3. In this work, all algorithms used the Within-Cluster Sum of Squares [Eq. (3)] as the fitness function, for simplicity, once our main objective is to evaluate the impact of K-Means on the behavior of memetic GSO, and not the influence of the fitness function on the behavior of EAs. For that issue, please, refer to Wong et al. (2011), Pacifico and Ludermir (2016).

All algorithms run in a MATLAB 7.6 environment. Thirty independent tests have been executed for each data set, and all evolutionary methods have started with the same initial population in each test, obtained by a random process, as explained in Sect. 2), as a manner to guarantee a fair evaluation and comparison among the selected techniques. Also, once the computational costs have been evaluated in terms of the average execution time of each model in each data set, each algorithm has been run and tested in a computer with an i7-7700K CPU, NVIDIA GeForce GTX 1060 6GB GPU and 32 GB RAM, independently (one algorithm each time), and no other programs, but the Operating System, were executed during the tests, granting the same environmental conditions to each method. For all tests, the adopted number of clusters C is equal to the number of classes per data set.

The hyperparameters for each evolutionary algorithm are presented in Table 4, and have been obtained from the literature: GA (Abdel-Kader 2010), DE (Das et al. 2007), PSO (Abdel-Kader 2010; Pacifico and Ludermir 2019), GSO (He et al. 2009; Pacifico and Ludermir 2014a, 2018), and BSA (Civicioglu 2013). The population size (S) and the maximum number of generations (t_{max}) have been chosen after a trial-and-error process, using values from Pacifico and Ludermir (2018),Pacifico and Ludermir (2019) as the staring point. Once the selected approaches did not improve significantly with higher population sizes nor higher number of generations, we kept the same parameter values as presented in Pacifico and Ludermir (2018), Pacifico and Ludermir (2019) to perform the current evaluation, once the computational costs associated with such parameters may grow very fast for high parameter values. The maximum number of K-Means executions ($t_{maxkmeans}$) has been obtained from Pacifico and Ludermir (2018).

Table 4 Hyperparameters for each EA

Algorithm	Parameter	Value
All EAs	t_{max}	100
	S	20
GA-k-means	Crossover rate	0.8
	Mutation rate	0.5
	Selection rate	0.5
	$t_{maxkmeans}$	100
DE-k-means	F	80%
	Crossover rate	0.9
	$t_{maxkmeans}$	100
PSO-k-means	c_1	2.0
	c_2	2.0
	w	0.9 to 0.4
	$t_{maxkmeans}$	100
BSA-k-means	*Mixrate*	1
	F	$3N(0, 1)$
	$t_{maxkmeans}$	100
GSO-based approaches	Scroungers rate	0.8
	θ_{max}	π/a^2
	α_0	$\pi/4$
	α_{max}	$\theta_{max}/2$
GSO-k-means	$t_{maxkmeans}$	100
FMKGSO and TMKGSO	$t_{maxkmeans}$	10

4.1 Discussion

The discussion is divided into three parts: first, we evaluate the experimental results for the real-world data sets (Sect. 4.1.1), followed by the evaluation on the experiments using the synthetic data sets (Sect. 4.1.2); An overall evaluation is also performed, based on the Friedman-Nemenyi hypothesis tests and the computational costs (in terms of the average execution times) obtained by each model (Sect. 4.1.3).

4.1.1 Real-world data sets

The experimental results for the real-world data sets are presented from Tables 5, 6, 7, 8, 9. The best results for each metric in each data set are bold faced.

In an empirical analysis, we can observe that the proposed hybrid memetic GSO approaches are able to find better values in relation to the fitness function (J) in most cases than hybrid partitional models that only use K-Means to refine the best solution found by the generational process of the EA. As illustrated in Fig. 3, sometimes the comparison hybrid partitional approaches are too slow while performing the global search that the only significant improvement is obtained after the generational process, when their final solution is refined by K-Means method (which presents fast local search capabilities). Fig. 3 also showed that even the use of just one K-Means step during the generational process for each member (MKGSO approach) is enough to improve significantly the convergence speed of GSO. The empirical analysis also showed that the hybrid memetic GSO models are more stable than the comparison hybrid partitional EA and K-Means algorithms.

The overall evaluation on the real-world problems obtained by the Friedman-Nemenyi hypothesis tests (Table 10) showed that the proposed memetic approaches are able to find clusters that are, in average, better represented by their final centroids (according to J and J_{e_2} indices), the final clusters are more similar to the actual classes (CR index), and the clusters are well-split in the problem space (according to D_{min}). The hypothesis tests also showed that, although MKGSO and TMKGSO use about the same number of K-Means steps for each member on current experimentation, MK-GSO presented more smooth convergence rates than TMKGSO, what may help MKGSO to avoid local minima points, but yet according to the current set of experiments, the performance of TMKGSO has been considered significantly better than the average results for MKGSO (in terms of the optimization of the fitness function - J).

According to the Friedman-Nemenyi tests, considering all five clustering metrics, the best results have been found by TMKGSO, followed by MKGSO and FMKGSO (second and third places, respectively).

When considering the average execution time obtained by each method in each data set (Table 11), we can see that FMKGSO and TMKGSO (which use K-Means refinements only in determined conditions) presented average times compatible with the other state-of-the-art EAs from the literature. By the other hand, MKGSO presented average execution times about four times higher than the other evaluated approaches, what may limit its applications when the data set is big (in terms of the number of data patterns). Such limitation may have occurred due to the fact that such method uses K-Means calls in each generation for each individual of the population, requiring many calls for allocation and deallocation of computational resources by the compiler.

4.1.2 Synthetic data sets

The results for both sets of synthetic data sets are presented from Tables 12, 13, 14, 15, 16. The best results for each metric in each data set are bold faced.

The empirical analysis considering both Disp and Prox sets showed that all algorithms have been significantly affected when the degree of class overlapping increased, specially when the class overlapping has been obtained by approximating the class centers of distribution (Prox data sets). In spite of that, the memetic approaches have been able, once again, to find the best average performances and degree of stability among all approaches. In particular, FMKGSO was able to find the best values for almost all tests, considering the five evaluation metrics, showing its capabilities and competitiveness. Fig. 4 illustrates the obtained partitions for each algorithm.

Considering the ranks provided by Friedman-Nemenyi hypothesis test (Table 17), the memetic approaches have found the best values for all evaluated indices. Although, in an overall analysis on the synthetic data sets, FMKGSO has been able to present ranks slightly better then the ranks of MKGSO, the Friedman-Nemenyi tests showed that there is no statistical significantly differences between FMKGSO performance and the performances of MKGSO (their differences are not greater than the Critical Distance), so both memetic algorithms are considered to have achieved the best results together. TMKGSO has been capable of reaching the third place on the evaluation considering the synthetic approaches, but, for all five clustering metrics, this method has found better performances than all the selected state-of-the-art comparison approaches, reinforcing the advantages of memetic models.

The evaluation considering the average execution times for each model showed that (Table 18) showed that, once more, FMKGSO and TMKGSO computational costs, in terms of time, are close to the computational costs of state-of-the-art algorithms, but MKGSO, although presenting better performances when considering the clustering metrics than the non-memetic comparison approaches, is still a little bit slower in execution.

4.1.3 Overall evaluation

At this point, an overall evaluation considering both the real-world and synthetic data sets is provided. For that, an

Table 5 Experimental results for real-world data sets: within-cluster sum of squares ($J^{\downarrow}$)

Data set	GA-k-means	DE-k-means	PSO-k-means	GSO-k-means	BSA-k-means	FMKGSO	MKGSO	TMKGSO
Abalone	$1.01\times 10^4 \pm 298.2$	$1.00\times 10^4 \pm 98.5$	$1.01\times 10^4 \pm 298.2$	9960.8 ± 102.6	$1.00\times 10^4 \pm 86.37$	9889.8 ± 0	**9889.8 ± 0**	**9889.8 ± 0**
Banknote authentication	**$4\cdot 40 \times 10^4 \pm 0$**	**$4\cdot 40 \times 10^4 \pm 0$**	**$4\cdot 40 \times 10^4 \pm 0$**	**$4\cdot 40 \times 10^4)pm\ 0$**	**$4\cdot 40 \times 10^4) \pm 0$**	**$4\cdot 40 \times 10^4) \pm 0$**	**$4\cdot 40 \times 10^4) \pm 0$**	**$4\cdot 40 \times 10^4) \pm 0$**
Blood transfusion	$6.79\times 10^8 \pm 4.36\times 10^6$	**$6\cdot 78 \times 10^8 \pm 0$**	$6.79\times 10^8 \pm 4.36\times 10^6$	**$6\cdot 78 \times 10^8 \pm 0$**	$6.78\times 10^8 \pm 3.16\times 10^6$	**$6\cdot 78 \times 10^8 \pm 0$**	**$6\cdot 78 \times 10^8 \pm 0$**	**$6\cdot 78 \times 10^8 \pm 0$**
Cancer	**243.44 ± 0**	**243.44 ± 0**	**243.44 ± 0**	**243.44 ± 0**	**243.44 ± 0**	**243.44 ± 0**	**243.44 ± 0**	**243.44 ± 0**
Diabetes	**$5\cdot 14 \times 10^6 \pm 0$**	**$5\cdot 14 \times 10^6 \pm 0$**	**$5\cdot 14 \times 10^6 \pm 0$**	**$5\cdot 14 \times 10^6 \pm 0$**	**$5\cdot 14 \times 10^6 \pm 0$**	**$5\cdot 14 \times 10^6 \pm 192.7$**	**$5\cdot 14 \times 10^6 \pm 0$**	**$5\cdot 14 \times 10^6 \pm 0$**
E. Coli	15.101 ± 0.758	14.989 ± 0.687	15.012 ± 0.776	14.808 ± 0.757	14.968 ± 0.729	**13.855 ± 0.0365**	14.042 ± 0.306	**13.875 ± 0.061**
Glass	372.46 ± 22.66	364.73 ± 28.92	372.46 ± 22.66	363.44 ± 24.42	378.61 ± 24.64	419.28 ± 43.71 **11.92**	369.82 ± 30.15	**344.48 ± 14.19**
Heart	549327.1 ± 12.76	549325.4 ± 12.61	549327.1 ± 12.76	549328.8 ± 12.74	549322.8 ± 11.85	**549314.7 ± 0**	**549314.7 ± 0**	**549314.7 ± 0**
Image segmentation	$1.49\times 10^7 \pm 2.00\times 10^6$	$1.46\times 10^7 \pm 1.72\times 10^6$	$1.49\times 10^7 \pm 2.00\times 10^6$	$1.53\times 10^7 \pm 2.48\times 10^6$	$1.50\times 10^7 \pm 2.27\times 10^6$	$1.52\times 10^7 \pm 1.05\times 10^6$	$1.45\times 10^7 \pm 1.10\times 10^6$	**$1.40 \times 10^7 \pm 5.13 \times 10^5$**
Ionosphere	**2419.4 ± 0.022**	2446.8 ± 150.4	**2419.4 ± 0.022**	**2419.4 ± 0.022**	2419.3 ± 0.022	**2419.4 ± 0**	**2419.4 ± 0**	**2419.4 ± 0**
Iris	78.854 ± 0.0020	78.854 ± 0.0020	78.854 ± 0.0020	78.855 ± 0.0014	78.854 ± 0.001	**78.851 ± 0**	**78.851 ± 0**	**78.851 ± 0**
Landsat satellite images	$1.53\times 10^7 \pm 3.29\times 10^5$	$1.52\times 10^7 \pm 1.92\times 10^5$	$1.53\times 10^7 \pm 3.71\times 10^5$	$1.53\times 10^7 \pm 2.93\times 10^5$	$1.54\times 10^7 \pm 4.23\times 10^5$	**$1\cdot 50 \times 10^7 \pm 3\cdot 61 \times 10^4$**	$1.52\times 10^7 \pm 1.69\times 10^5$	$1.52\times 10^7 \pm 1.28\times 10^5$
Letter recognition	$6.20\times 10^5 \pm 4273.2$	$6.20\times 10^5 \pm 3596.7$	$6.19\times 10^5 \pm 4354.8$	$6.20\times 10^5 \pm 4587.3$	$6.20\times 10^5 \pm 4002.8$	**$6\cdot 12 \times 10^5 \pm 1281\cdot 5$**	$6.17\times 10^5 \pm 3575.1$	$6.16\times 10^5 \pm 3842.6$
Optical recognition	$3.75\times 10^6 \pm 5.36\times 10^4$	**$3\cdot 73 \times 10^6 \pm 7\cdot 37 \times 10^4$**	$3.75\times 10^6 \pm 5.36\times 10^4$	$3.75\times 10^6 \pm 6.29\times 10^4$	$3.71\times 10^6 \pm 4.30\times 10^4$	$3.89\times 10^6 \pm 1{,}07\times 10^5$	$3.90\times 10^6 \pm 1.16\times 10^5$	$3.78\times 10^6 \pm 6.60\times 10^4$
Page blocks classification	$3.33\times 10^{10} \pm 1.40\times 10^{10}$	$2.03\times 10^{10} \pm 5.16\times 10^9$	$2.37\times 10^{10} \pm 9.25\times 10^9$	$2.31\times 10^{10} \pm 6.16\times 10^9$	$2.09\times 10^{10} \pm 3.92\times 10^9$	$2.02\times 10^{10} \pm 4.90\times 10^9$	$1.80\times 10^{10} \pm 5.56\times 10^9$	**$1\cdot 62 \times 10^{10} \pm 4\cdot 80 \times 10^9$**
Pen-digits	$5.07\times 10^7 \pm 8.52\times 10^5$	$5.04\times 10^7 \pm 1.10\times 10^6$	$5.06\times 10^7 \pm 8.28\times 10^5$	$5.07\times 10^7 \pm 8.19\times 10^5$	$5.07\times 10^7 \pm 1.08\times 10^6$	**$4\cdot 97 \times 10^7 \pm 1\cdot 02 \times 10^6$**	$5.01\times 10^7 \pm 8.86\times 10^5$	$5.01\times 10^7 \pm 4.81\times 10^5$
Seeds	587.82 ± 0.7154	587.95 ± 0.7376	587.82 ± 0.7154	587.71 ± 0.6582	587.72 ± 0.682	**587.32 ± 0**	**587.32 ± 0**	**587.32 ± 0**
Waveform	**133119.2 ± 0**	**133119.2 ± 0**	**133119.2 ± 0**	**133119.2 ± 0**	**133119.2 ± 0**	**133119.2 ± 0**	**133119.2 ± 0**	**133119.2 ± 0**
Wine	$2.41\times 10^6 \pm 9.71\times 10^4$	**$2\cdot 37 \times 10^6 \pm 0$**	$2.41\times 10^6 \pm 9.71\times 10^4$	**$2\cdot 37 \times 10^6 \pm 0$**	$2.42\times 10^6 \pm 1.04\times 10^5$	**$2\cdot 37 \times 10^6 \pm 0$**	**$2\cdot 37 \times 10^6 \pm 0$**	**$2\cdot 37 \times 10^6 \pm 0$**
Yeast	47.497 ± 2.509	48.500 ± 3.255	47.852 ± 2.342	46.333 ± 0.408	47.878 ± 2.487	**$45\cdot 331 \pm 0\cdot 1788$**	45.402 ± 0.246	45.544 ± 0.383

Table 6 Experimental results for real-world data sets: correct Rand index ($CR^{\uparrow}$)

Data set	GA-k-means	DE-k-means	PSO-k-means	GSO-k-means	BSA-k-means	FMKGSO	MKGSO	TMKGSO
Abalone	0.1211 ± 0.017	0.1181 ± 0.014	0.1211 ± 0.017	0.1284 ± 0.010	0.1194 ± 0.013	**0.1331 ± 0**	**0.1331 ± 0**	**0.1331 ± 0**
Banknote authentication	**0.0485 ± 0**	**0.0485 ± 0**	**0.0485 ± 0**	**0.0485 ± 0**	**0.0485 ± 0**	**0.0485 ± 0**	**0.0485 ± 0**	**0.0485 ± 0**
Blood transfusion	0.0777 ± 0.004	**0.0795 ± 0**	0.0777 ± 0.004	**0.0795 ± 0**	0.0776 ± 0.003	**0.0795 ± 0**	**0.0795 ± 0**	**0.0795 ± 0**
Cancer	**0.8391 ± 0**	**0.8391 ± 0**	**0.8391 ± 0**	**0.8391 ± 0**	**0.8391 ± 0**	**0.8391 ± 0**	**0.8391 ± 0**	**0.8391 ± 0**
Diabetes	**0.0744 ± 0**	**0.0744 ± 0**	**0.0744 ± 0**	**0.0744 ± 0**	**0.0744 ± 0**	**0.0744 ± 0**	**0.0744 ± 0**	**0.0744 ± 0**
E. coli	0.4116 ± 0.055	0.4056 ± 0.047	0.4223 ± 0.052	0.4169 ± 0.052	0.4102 ± 0.053	0.4297 ± 0.019	**0.4629 ± 0.047**	0.4373 ± 0.030
Glass	0.2452 ± 0.029	0.2571 ± 0.023	0.2452 ± 0.029	0.2565 ± 0.020	0.2492 ± 0.025	0.2530 ± 0.008	0.2539 ± 0.006	**0.2589 ± 0.006**
Heart	0.0289 ± 0.001	0.0291 ± 0.001	0.0289 ± 0.001	0.0288 ± 0.001	0.0293 ± 0.001	**0.0302 ± 0**	**0.0302 ± 0**	**0.0302 ± 0**
Image segmentation	0.3586 ± 0.049	**0.3609 ± 0.045**	0.3586 ± 0.049	0.3596 ± 0.054	0.3526 ± 0.048	0.2867 ± 0.048	0.3377 ± 0.042	0.3422 ± 0.053
Ionosphere	0.1749 ± 0.002	0.1695 ± 0.032	0.1747 ± 0.002	0.1755 ± 0.002	0.1752 ± 0.002	**0.1776 ± 0**	**0.1776 ± 0**	**0.1776 ± 0**
Iris	0.7205 ± 0.006	0.7210 ± 0.007	0.7205 ± 0.006	0.7182 ± 0.005	0.7191 ± 0.005	**0.7302 ± 0**	**0.7302 ± 0**	**0.7302 ± 0**
Landsat satellite images	0.4975 ± 0.0591	0.4979 ± 0.0497	0.4938 ± 0.0678	0.5084 ± 0.0507	0.4895 ± 0.070	**0.5666 ± 0.0081**	0.5395 ± 0.0346	0.5377 ± 0.0238
Letter recognition	0.1328 ± 0.004	0.1314 ± 0.006	**0.1334 ± 0.005**	0.1327 ± 0.004	0.1330 ± 0.006	0.1317 ± 0.003	0.1309 ± 0.004	0.1324 ± 0.003
Optical recognition	0.6142 ± 0.058	**0.6360 ± 0.045**	0.6142 ± 0.058	0.6108 ± 0.053	0.6500 ± 0.071	0.5699 ± 0.049	0.5656 ± 0.049	0.6068 ± 0.047
Page blocks classification	0.0480 ± 0.041	**0.0491 ± 0.039**	0.0205 ± 0.035	0.0221 ± 0.026	0.0131 ± 0.019	0.0178 ± 0.021	0.0176 ± 0.020	0.0152 ± 0.015
Pen-digits	0.5504 ± 0.038	0.5414 ± 0.035	**0.5541 ± 0.033**	0.5499 ± 0.036	0.5514 ± 0.036	0.5375 ± 0.016	0.5415 ± 0.029	0.5519 ± 0.029
Seeds	0.7152 ± 0.003	0.7139 ± 0.003	0.7152 ± 0.003	0.7149 ± 0.003	0.7160 ± 0.002	**0.7166 ± 0**	**0.7166 ± 0**	**0.7166 ± 0**
Waveform	**0.2536 ± 0**	**0.2536 ± 0**	**0.2536 ± 0**	**0.2536 ± 0**	**0.2536 ± 0**	**0.2536 ± 0**	**0.2536 ± 0**	**0.2536 ± 0**
Wine	0.3662 ± 0.011	**0.3711 ± 0**	0.3662 ± 0.011	**0.3711 ± 0**	0.3645 ± 0.013	**0.3711 ± 0**	**0.3711 ± 0**	**0.3711 ± 0**
Yeast	0.1350 ± 0.0133	0.1323 ± 0.0132	0.1347 ± 0.0164	0.1354 ± 0.0147	0.1334 ± 0.0156	0.1460 ± 0.0039	**0.1498 ± 0.0058**	0.1470 ± 0.0058

evaluation based on the Friedman-Nemenyi hypothesis tests considering all data sets (real and synthetic) is performed (Table 19). By considering both sets of data sets, we can have a better understanding on the behavior of all selected algorithms (hybrid partitional approaches, and hybrid memetic partitional approaches) when dealing with either controlled and uncontrolled testing scenarios, showing how robust such approaches actually are.

The overall analysis ranked TMKGSO as the best approach among all the tested algorithms, considering the rank values for all metrics, followed by FMKGSO and MKGSO, respectively. The overall evaluation showed that including some K-Means during the generational process of EAs is quite advantageous (the memetic approach), and it is a preferred strategy than only use K-Means only to refine the best solution found so far by the EAs. Considering the selected five clustering metrics, we can concluded that, for the current testing bed, the memetric approaches are able to find final solutions with clusters that are, in average, better represented by their centroids vectors (according to J and J_{e_2} indices), more similar to the actual classes (CR index), more compact (according to D_{max} index), and well-split in the problem space in relation to each other (according to D_{min}). The proposed memetic

Table 7 Experimental results for real-world data sets: weighted quantization error ($J_{e_2}^{\downarrow}$)

Data set	GA-k-means	DE-k-means	PSO-k-means	GSO-k-means	BSA-k-means	FMKGSO	MKGSO	TMKGSO
Abalone	2.4191 ± 0.071	2.3979 ± 0.024	2.4191 ± 0.071	2.3847 ± 0.025	2.4076 ± 0.020	**2.3677 ± 0**	**2.3677 ± 0**	**2.3677 ± 0**
Banknote authentication	**32.106 ± 0**	**32.106 ± 0**	**32.106 ± 0**	**32.106 ± 0**	**32.106 ± 0**	**32.106 ± 0**	**32.106 ± 0**	**32.106 ± 0**
Blood transfusion	$9.08\times 10^{5} \pm 5.83\times 10^{3}$	$\mathbf{9 \cdot 06 \times 10^{5} \pm 0}$	$9.08\times 10^{5} \pm 5.83\times 10^{3}$	$\mathbf{9 \cdot 06 \times 10^{5} \pm 0}$	$9.07\times 10^{5} \pm 4.23\times 10^{3}$	$\mathbf{9 \cdot 06 \times 10^{5} \pm 0}$	$\mathbf{9 \cdot 06 \times 10^{5} \pm 0}$	$\mathbf{9 \cdot 06 \times 10^{5} \pm 0}$
Cancer	**0.3483 ± 0**	**0.3483 ± 0**	**0.3483 ± 0**	**0.3483 ± 0**	**0.3483 ± 0**	**0.3483 ± 0**	**0.3483 ± 0**	**0.3483 ± 0**
Diabetes	**6695.8 ± 0**	**6695.8 ± 0**	**6695.8 ± 0**	**6695.8 ± 0**	**6695.8 ± 0**	**6695.9 ± 0.251**	**6695.8 ± 0**	**6695.8 ± 0**
E. coli	0.0449 ± 0.002	0.0446 ± 0.002	0.0447 ± 0.002	0.0441 ± 0.002	0.0445 ± 0.002	**0.0412 ± 0.0001**	0.0418 ± 0.0009	**0.0413 ± 0.0002**
Glass	1.7405 ± 0.106	1.7044 ± 0.135	1.7405 ± 0.106	1.6983 ± 0.114	1.7692 ± 0.115	1.9592 ± 0.204	1.7281 ± 0.141	**1.6097 ± 0.066**
Heart	**2034.5 ± 0.047**	**2034.5 ± 0.047**	**2034.5 ± 0.047**	**2034.6 ± 0.047**	2034.5 ± 0.043	**2034.5 ± 0**	**2034.5 ± 0**	**2034.5 ± 0**
Image segmentation	6435.9 ± 867.5	6324.7 ± 744.7	6435.9 ± 867.5	6616.2 ± 1074.9	6525.9 ± 986.6	6567.5 ± 455.7	6273.2 ± 477.3	**6063.9 ± 222.2**
Ionosphere	**6.8928 ± 0.0001**	6.9711 ± 0.428	**6.8929 ± 0.0001**	**6.8928 ± 0.0001**	6.8928 ± 0.00006	**6.8928 ± 0**	**6.8928 ± 0**	**6.8928 ± 0**
Iris	**0.5257 ± 0.00001**	**0.5257 ± 0.00001**	0.5266 ± 0.0013	**0.5257 ± 0.00001**	0.5256 ± 0.00001	**0.5257 ± 0**	**0.5257 ± 0**	**0.5257 ± 0**
Landsat satellite images	2379.5 ± 51.14	2363.2 ± 29.74	2383.4 ± 57.65	2372.4 ± 45.57	2396.8 ± 65.87	**2333.9 ± 5.609**	2355.1 ± 26.30	2357.5 ± 19.91
Letter recognition	31.022 ± 0.213	31.049 ± 0.179	30.993 ± 0.217	31.005 ± 0.229	31.002 ± 0.200	**30.623 ± 0.064**	30.864 ± 0.178	30.817 ± 0.192
Optical recognition	666.47 ± 9.538	**665.05 ± 13.13**	666.47 ± 9.538	667.58 ± 11.20	661.57 ± 7.652	692.93 ± 1.107	694.88 ± 20.65	672.33 ± 11.75
Page blocks classification	$6.08\times 10^{6} \pm 2.57\times 10^{6}$	$3.72\times 10^{6} \pm 9.35\times 10^{5}$	$4.34\times 10^{6} \pm 1.69\times 10^{6}$	$4.22\times 10^{6} \pm 1.12\times 10^{6}$	$3.82\times 10^{6} \pm 7.16\times 10^{5}$	$3.69\times 10^{6} \pm 8.95\times 10^{5}$	$3.29\times 10^{6} \pm 1.02\times 10^{6}$	$\mathbf{2 \cdot 96 \times 10^{6} \pm 8.77 \times 10^{5}}$
Pen-digits	4619.3 ± 77.58	4594.1 ± 100.7	4610.2 ± 75.39	4619.0 ± 74.58	4617.5 ± 98.40	**4528.3 ± 93.61**	4566.0 ± 80.69	4566.9 ± 43.77
Seeds	2.7991 ± 0.0034	2.7998 ± 0.0035	2.7991 ± 0.0034	2.7986 ± 0.0031	2.7986 ± 0.003	**2.7968 ± 0**	**2.7968 ± 0**	**2.7968 ± 0**
Waveform	**26.624 ± 0**	**26.624 ± 0**	**26.624 ± 0**	**26.624 ± 0**	**26.624 ± 0**	**26.624 ± 0**	**26.624 ± 0**	**26.624 ± 0**
Wine	13558.4 ± 545.6	**13318.5 ± 0**	13558.4 ± 545.6	**13318.5 ± 0**	13607.6 ± 588.2	**13318.5 ± 0**	**13318.5 ± 0**	**13318.5 ± 0**
Yeast	0.0320 ± 0.0017	0.0327 ± 0.0022	0.0322 ± 0.0016	0.0312 ± 0.0003	0.0322 ± 0.0016	**0.0305 ± 0.0001**	**0.0306 ± 0.0002**	**0.0307 ± 0.0003**

approaches have also been able to find the best stability throughout the experimentation, reinforcing their reliability.

For the current testing bed, the best hybrid model has been the TMKGSO approach, which executes some K-Means steps after a prefixed number of generations for each group member, which either has found good performances, according to the clustering indices, and has been able to execute with a computational cost, in terms of average execution time, close to what is expected from any Evolutionary Algorithm from current state-of-the-art. By the other hand, although able to find good average performances, MKGSO approach has been quite slow in its execution, what may represent a limitation. But, if time is not the main concern when a new data clustering system is proposed, MKGSO would still be a better choice than non-memetic approaches, once it still preserves GSO good global searching capabilities and K-Means fast convergence speed, so the designer of such system should have this trade-off in mind when making the decision.

Table 8 Experimental results for real-world data sets: intra-cluster distance ($D^{\downarrow}_{max}$)

Data set	GA-k-means	DE-k-means	PSO-k-means	GSO-k-means	BSA-k-means	FMKGSO	MKGSO	TMKGSO
Abalone	6.8652 ± 0.501	**6.6643 ± 0.352**	6.8652 ± 0.501	6.8051 ± 0.279	6.7650 ± 0.484	6.7044 ± 0	6.7044 ± 0	6.7044 ± 0
Banknote authentication	**35.978 ± 0**	**35.978 ± 0**	**35.978 ± 0**	**35.978 ± 0**	**35.978 ± 0**	**35.978 ± 0**	**35.978 ± 0**	**35.978 ± 0**
Blood transfusion	$3.93\times 10^6 \pm 1.68\times 10^5$	$\mathbf{3 \cdot 86 \times 10^6 \pm 0}$	$3.93\times 10^6 \pm 1.68\times 10^5$	$\mathbf{3 \cdot 86 \times 10^6 \pm 0}$	$3.91\times 10^6 \pm 1.33\times 10^5$	$\mathbf{3 \cdot 86 \times 10^6 \pm 0}$	$\mathbf{3 \cdot 86 \times 10^6 \pm 0}$	$\mathbf{3 \cdot 86 \times 10^6 \pm 0}$
Cancer	**0.8050 ± 0**	**0.8050 ± 0**	**0.8050 ± 0**	**0.8050 ± 0**	**0.8050 ± 0**	**0.8050 ± 0**	**0.8050 ± 0**	**0.8050 ± 0**
Diabetes	$\mathbf{1 \cdot 68 \times 10^4 \pm 0}$	$\mathbf{1 \cdot 68 \times 10^4 \pm 0}$	$\mathbf{1 \cdot 68 \times 10^4 \pm 0}$	$\mathbf{1 \cdot 68 \times 10^4 \pm 0}$	$\mathbf{1 \cdot 68 \times 10^4 \pm 0}$	$\mathbf{1 \cdot 68 \times 10^4 \pm 11.56}$	$\mathbf{1 \cdot 68 \times 10^4 \pm 0}$	$\mathbf{1 \cdot 68 \times 10^4 \pm 0}$
E. Coli	**0.1029 ± 0.015**	0.1048 ± 0.018	0.1118 ± 0.019	0.1069 ± 0.016	0.1083 ± 0.017	0.1330 ± 0	0.1307 ± 0.013	0.1330 ± 0
Glass	9.0743 ± 1.639	9.2814 ± 1.764	9.0743 ± 1.639	7.9628 ± 2.163	8.6093 ± 1.730	8.4604 ± 1.825	**7.3165 ± 0.912**	7.4499 ± 1.109
Heart	**2718.0 ± 2.542**	**2718.5 ± 2.425**	**2718.0 ± 2.542**	**2718.0 ± 2.323**	2718.9 ± 2.423	**2720.7 ± 0**	**2720.7 ± 0**	**2720.7 ± 0**
Image segmentation	231386.2 ± 58399.2	235047.8 ± 54517.2	231386.2 ± 58399.2	213276.6 ± 75049.5	222034.2 ± 68870.5	**183540.4 ± 60964.4**	206166.7 ± **57345.6**	209843.8 ± 51638.3
Ionosphere	10.547 ± 0.0127	**10.505 ± 0.2394**	10.546 ± 0.0126	10.550 ± 0.0127	10.548 ± 0.012	10.561 ± 0	10.561 ± 0	10.561 ± 0
Iris	0.6488 ± 0.004	0.6485 ± 0.004	0.6488 ± 0.004	0.6504 ± 0.003	0.6497 ± 0.003	**0.6423 ± 0**	**0.6423 ± 0**	**0.6423 ± 0**
Landsat satellite images	5769.5 ± 1306.0	5729.4 ± 1424.2	5738.7 ± 1364.3	5678.1 ± 1309.2	6343.7 ± 1548.2	**4726.1 ± 6.984**	5383.9 ± 1105.2	5507.9 ± 1033.7
Letter recognition	48.858 ± 1.816	49.113 ± 2.401	**48.541 ± 2.159**	48.644 ± 2.008	49.019 ± 2.272	48.739 ± 0.639	49.035 ± 1.715	48.682 ± 1.439
Optical recognition	800.70 ± 51.17	**781.49 ± 46.10**	800.70 ± 51.17	795.23 ± 49.98	772.15 ± 20.07	829.63 ± 66.73	827.20 ± 64.67	798.93 ± 44.25
Page blocks classification	$1.20\times 10^9 \pm 4.42\times 10^7$	$\mathbf{6.28 \times 10^8 \pm 3.31 \times 10^8}$	$1.18\times 10^9 \pm 0$	$1.16\times 10^9 \pm 1.42\times 10^8$	$1.18\times 10^9 \pm 0$	$1.01\times 10^9 \pm 3.27\times 10^8$	$8.30\times 10^8 \pm 3.84\times 10^8$	$6.76\times 10^8 \pm 3.65\times 10^8$
Pen-digits	6534.3 ± 427.0	**6358.0 ± 438.2**	6455.9 ± 463.5	6560.1 ± 427.8	6490.3 ± 451.4	6411.8 ± 262.2	6423.8 ± 377.0	6525.4 ± 484.0
Seeds	**3.0182 ± 0**	**3.0182 ± 0**	**3.0182 ± 0**	**3.0182 ± 0**	**3.0182 ± 0**	**3.0182 ± 0**	**3.0182 ± 0**	**3.0182 ± 0**
Waveform	**27.140 ± 0**	**27.140 ± 0**	**27.140 ± 0**	**27.140 ± 0**	**27.140 ± 0**	**27.140 ± 0**	**27.140 ± 0**	**27.140 ± 0**
Wine	**27559.4 ± 3196.2**	28956.4± 0	**27559.4 ± 3196.2**	28956.4± 0	27364.0 ± 3253.5	28956.4± 0	28956.4± 0	28956.4± 0
Yeast	0.0638 ± 0.0107	**0.0598 ± 0.0082**	0.0681 ± 0.0161	0.0662 ± 0.0118	0.0305 ± 0.0043	0.0931 ± 0.0127	0.0980 ± 0	0.0907 ± 0.0150

5 Conclusions

This paper presented an evaluation on the behavior of partitional Group Search Optimization when employing K-Means algorithm as a local search operator, to deal with data clustering problem. In that sense, three hybrid memetic algorithms are proposed (FMKGSO, MKGSO and TMKGSO), which use different ways to combine GSO and K-Means in an attempt to improve GSO population during the generational process of the method. Although K-Means is easily trapped in local optima points, it is known that it would represent a good local searcher when combined to global search metaheuristics, like evolutionary algorithms. Also, GSO, which is known to present good global searching capabilities and mechanisms to escape from local optima points, would benefit considerably by adopting K-Means to speedup the algorithm convergence when dealing with data clustering problems.

To evaluate the proposed methods, five state-of-the-art partitional clustering evolutionary algorithms are tested:

Table 9 Experimental results for real-world data sets: inter-cluster separation ($D_{min}^{\uparrow}$)

Data set	GA-k-means	DE-k-means	PSO-k-means	GSO-k-means	BSA-k-means	FMKGSO	MKGSO	TMKGSO
Abalone	12.272 ± 1.314	12.654 ± 0.800	12.272 ± 1.314	12.538 ± 0.734	12.401 ± 1.065	**12.926 ± 0**	**12.926 ± 0**	**12.926 ± 0**
Banknote authentication	**149.38 ± 0**	**149.38 ± 0**	**149.38 ± 0**	**149.38 ± 0**	**149.38 ± 0**	**149.38 ± 0**	**149.38 ± 0**	**149.38 ± 0**
Blood transfusion	$\mathbf{9 \cdot 61 \times 10^6 \pm 8 \cdot 40 \times 10^5}$	$9.27\times 10^6 \pm 0$	$\mathbf{9 \cdot 61 \times 10^6 \pm 8 \cdot 40 \times 10^5}$	$9.27\times 10^6 \pm 0$	$\mathbf{9 \cdot 54 \times 10^6 \pm 6 \cdot 65 \times 10^5}$	$9.27\times 10^6 \pm 0.057$	$9.27\times 10^6 \pm 0$	$9.27\times 10^6 \pm 0$
Cancer	**2.3366 ± 0**	**2.3366 ± 0**	**2.3366 ± 0**	**2.3366 ± 0**	**2.3366 ± 0**	**2.3366 ± 0**	**2.3366 ± 0**	**2.3366 ± 0**
Diabetes	$\mathbf{5 \cdot 00 \times 10^4 \pm 0}$	$\mathbf{5 \cdot 00 \times 10^4 \pm 0}$	$\mathbf{5 \cdot 00 \times 10^4 \pm 0}$	$\mathbf{5 \cdot 00 \times 10^4 \pm 0}$	$\mathbf{5 \cdot 00 \times 10^4 \pm 0}$	$\mathbf{5 \cdot 00 \times 10^4 \pm 0.0001}$	$\mathbf{5 \cdot 00 \times 10^4 \pm 0}$	$\mathbf{5 \cdot 00 \times 10^4 \pm 0}$
E. coli	0.0450 ± 0.007	0.0462 ± 0.009	0.0486 ± 0.010	0.0472 ± 0.010	0.0459 ± 0.006	0.0512 ± 0.0003	**0.0539 ± 0.007**	0.0516 ± 0.004
Glass	2.2558 ± 1.747	2.7011 ± 0.960	2.2558 ± 1.747	2.8926 ± 1.530	2.4239 ± 1.155	**6.6358 ± 4.960**	3.2817 ± 0.066	3.2733 ± 0.125
Heart	6643.5 ± 34.85	6650.0 ± 35.45	6643.5 ± 34.85	6640.3 ± 34.96	6657.3 ± 33.94	**6683.5 ± 0**	**6683.5 ± 0**	**6683.5 ± 0**
Image segmentation	8732.9 ± 1575.8	8793.0 ± 1076.8	8732.9 ± 1575.8	8904.6 ± 1330.5	8491.9 ± 1401.4	**10740.8 ± 1678.7**	9098.7 ± 1032.1	9179.6 ± 789.3
Ionosphere	9.4467 ± 0.005	**11.117 ± 9.143**	9.4464 ± 0.005	9.4479 ± 0.005	9.4472 ± 0.004	9.4519 ± 0	9.4519 ± 0	9.4519 ± 0
Iris	3.2079 ± 0.015	3.2089 ± 0.015	3.2079 ± 0.015	3.2026 ± 0.011	3.2047 ± 0.012	**3.2299 ± 0**	**3.2299 ± 0**	**3.2299 ± 0**
Landsat satellite images	4054.4 ± 387.6	4097.7 ± 300.3	4057.7 ± 407.9	4050.0 ± 363.7	3947.5 ± 432.8	**4290.3 ± 423.6**	4174.8 ± 225.9	4214.5 ± 131.8
Letter recognition	24.711 ± 2.504	23.533 ± 3.363	24.742 ± 2.466	24.912 ± 1.955	24.288 ± 3.441	**26.951 ± 0.726**	25.133 ± 2.200	25.773 ± 2.355
Optical recognition	435.01 ± 163.3	410.02 ± 107.7	435.01 ± 163.3	443.51 ± 159.3	449.38 ± 159.7	**646.48 ± 151.1**	**645.66 ± 151.8**	486.11 ± 185.4
Page blocks classification	$1.97\times 10^7 \pm 1.71\times 10^7$	$\mathbf{1 \cdot 87 \times 10^8 \pm 2 \cdot 05 \times 10^8}$	$1.08\times 10^7 \pm 6.18\times 10^6$	$2.38\times 10^7 \pm 1.88\times 10^7$	$1.17\times 10^7 \pm 7.54\times 10^6$	$4.60\times 10^7 \pm 9.03\times 10^7$	$4.42\times 10^7 \pm 8.87\times 10^7$	$3.90\times 10^7 \pm 8.36\times 10^7$
Pen-digits	5334.6 ± 1261.6	6096.3 ± 1639.2	5158.1 ± 1109.4	5407.8 ± 1375.5	5520.1 ± 922.0	**6477.1 ± 882.0**	6115.9 ± 1300.3	5705.8 ± 1316.9
Seeds	13.377 ± 0.0367	**13.388 ± 0.0412**	13.377 ± 0.0367	13.375 ± 0.0368	13.370 ± 0.032	13.353 ± 0	13.353 ± 0	13.353 ± 0
Waveform	**52.632 ± 0**	**52.632 ± 0**	**52.632 ± 0**	**52.632 ± 0**	**52.632 ± 0**	**52.632 ± 0**	**52.632 ± 0**	**52.632 ± 0**
Wine	83986.4 ± 24870.4	73087.8 ± 0	83986.4 ± 24870.4	73087.8 ± 0	**85486.2 ± 25291.7**	73087.8 ± 0.0005	73087.8 ± 0	73087.8 ± 0
Yeast	0.0300 ± 0.0038	0.0286 ± 0.0035	0.0305 ± 0.0033	0.0301 ± 0.0032	0.0305 ± 0.0043	**0.0311 ± 0.0008**	0.0304 ± 0.0023	0.0296 ± 0.0020

GA, DE, PSO, standard GSO and BSA. As a current and usual practice in data clustering literature, the comparison approaches are hybridized with K-Means in such a way that their best final solutions are refined by K-Means after the generational process, so we could access the actual influence of such technique on the behavior of EAs.

The experimental testing bed adopted in this work included 20 real-world and 10 synthetic data sets. To access the potential of the proposed memetic models, five clustering metrics have been employed (within-cluster sum of squares, correct Rand index, weighted quantization error, intra-cluster distance and inter-cluster separation), and the experimental analysis has been obtained by means of both an empirical analysis, and by an overall evaluation obtained through the application of a rank system considering the Friedman-Nemenyi hypothesis tests on each clustering metric.

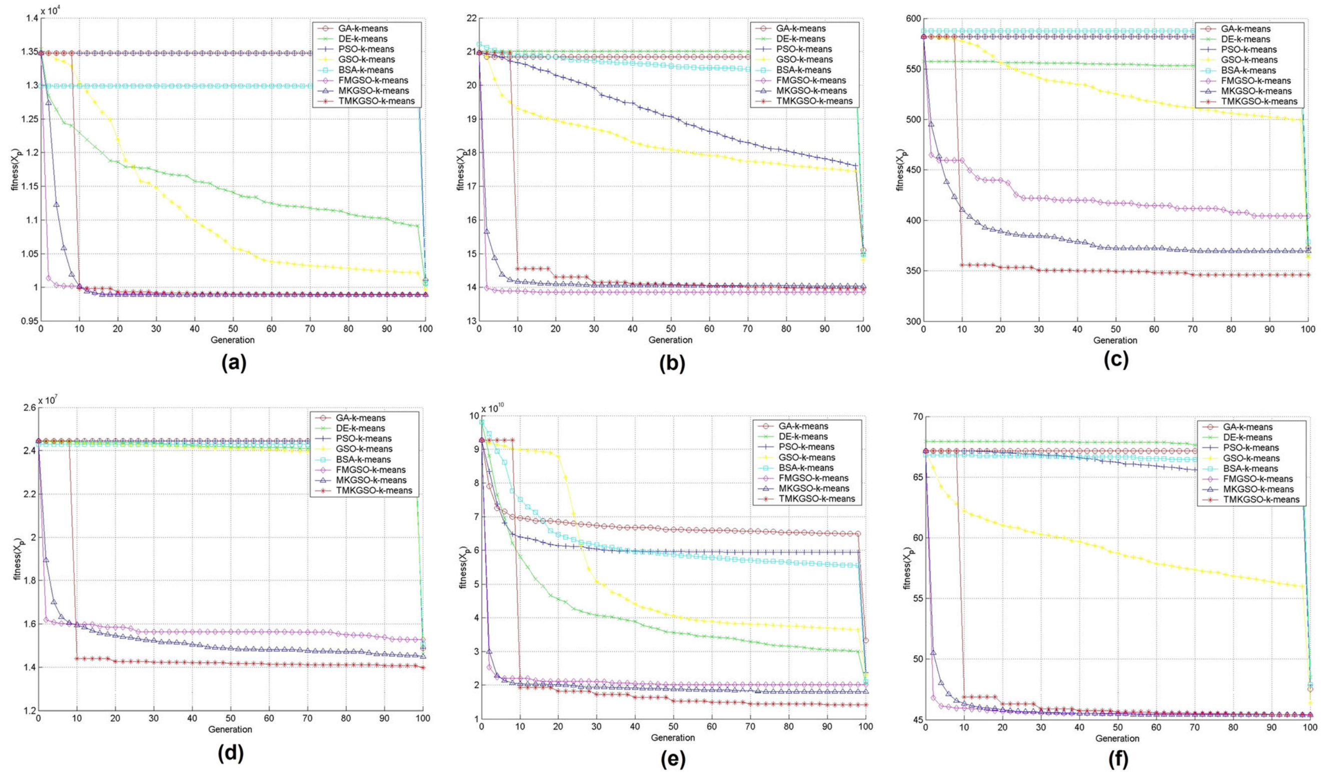

Fig. 3 Average convergence graph for the best solution found so far $\mathbf{X}_p^t$ for **a** Abalone, **b** E. coli, **c** Glass, **d** Image segmentation, **e** Page blocks classification, **f** Yeast

Table 10 Overall evaluation on real-world data sets: average Friedman–Nemenyi ranks (and position) for each metric. The best results are bold faced

Algorithm	$J^{\downarrow}$	$CR^{\uparrow}$	$J_{e_2}^{\downarrow}$	$D_{max}^{\downarrow}$	$D_{min}^{\uparrow}$	Total
GA-k-means	141.0858 (8)	112.6358 (6)	141.0883 (8)	121.0775 (5)	106.7317 (7)	34
DE-k-means	130.5425 (4)	115.8492 (5)	130.5425 (4)	**111.9875 (1)**	116.8375 (4)	18
PSO-k-means	138.2092 (7)	111.3783 (8)	138.2092 (7)	121.4925 (6)	105.0050 (8)	36
GSO-k-means	132.0817 (5)	117.3917 (4)	132.0783 (5)	118.5375 (2)	108.2017 (5)	21
BSA-k-means	136.5800 (6)	112.1883 (7)	136.5792 (6)	120.5900 (4)	106.9175 (6)	29
FMKGSO	95.6808 (2)	127.4767 (2)	95.7883 (2)	125.5967 (8)	**148.2208 (1)**	15
MKGSO	100.6042 (3)	132.1517 (3)	100.6300 (3)	124.8192 (7)	136.4942 (2)	18
TMKGSO	**89.2158 (1)**	**134.9283 (1)**	**89.0842 (1)**	119.8992 (3)	135.5917 (3)	**9**

The critical distance is CD = 2.3477

The experimental results showed that, when K-Means is executed within the generational process of GSO at least by a few steps, it is able to aggregate more local information than when it is adopted only to improve the final solution found so far by an EA, speeding up the exploration/exploitation promoted by the memetic system. EAs are known to present slow convergence speeds, so when K-Means is only employed after the generational process of the EA, given K-Means lack of recovering mechanisms from local optima points, the final solution found by the hybrid model may still be in a region of the problem space that do not contain the global optimum point, so K-Means will still be trapped in such region.

All memetic approaches have been able to improve the comparison approaches, and the best evaluated memetic combination between K-Means and GSO (according to the Friedman-Nemenyi ranks and the evaluation on the average execution times) has been achieved when K-Means executes for a limited number of steps, for each population individual, after a prefixed number of GSO generations (TMKGSO algorithm), followed by FMKGSO (where individuals are improved by K-Means only if they fail to improve for a prefixed number of consecutive generations).

Table 11 Average execution times for the real-world data sets (in seconds)

Data set	GA-k-means	DE-k-means	PSO-k-means	GSO-k-means	BSA-k-means	FMKGSO	MKGSO	TMKGSO
Abalone	3.2324	3.1665	3.1848	3.7053	3.2208	3.2161	11.993	2.2860
Banknote authentication	0.6849	0.6814	0.6726	0.8054	0.6885	0.7939	2.5665	0.6990
Blood transfusion	0.3938	0.3792	0.3808	0.4633	0.3883	0.3616	1.4789	0.2945
Cancer	0.4844	0.4633	0.4654	0.5878	0.4986	0.4299	1.8070	0.4041
Diabetes	0.5166	0.4940	0.4947	0.6159	0.5155	0.5596	1.8619	0.5105
E. coli	0.5914	0.5820	0.5820	0.8138	0.7540	1.4727	3.0436	0.9399
Glass	0.3681	0.3524	0.3547	0.5540	0.5159	0.6873	1.8540	0.5448
Heart	0.2361	0.2202	0.2214	0.3206	0.2607	0.2866	0.9268	0.2626
Image segmentation	6.2777	6.3355	6.2147	8.2467	7.3594	10.891	21.707	7.0738
Ionosphere	0.4540	0.4443	0.4522	0.7103	0.6920	0.5505	1.8175	0.5118
Iris	0.1545	0.1413	0.1473	0.2002	0.1546	0.2267	0.6869	0.1886
Landsat satellite images	47.173	47.168	47.064	55.600	50.703	85.043	152.46	53.855
Letter recognition	254.40	254.61	253.51	294.70	263.82	553.67	809.80	334.92
Optical recognition	105.40	106.37	106.07	129.77	129.52	169.31	343.48	100.96
Page blocks classification	7.2430	7.8097	6.9642	8.0242	8.0102	11.203	24.978	7.1443
Pen-digits	52.908	52.925	52.971	61.555	54.205	87.005	174.22	55.801
Seeds	0.2066	0.1930	0.1937	0.2777	0.2206	0.2795	0.8892	0.2312
Waveform	9.8666	9.8409	9.8524	11.496	10.095	11.378	33.455	9.7679
Wine	0.2291	0.2132	0.2155	0.3475	0.3063	0.3398	0.9966	0.2940
Yeast	3.1202	3.0142	3.0257	3.6732	3.3673	8.0542	11.848	4.5985

Table 12 Experimental results for synthetic data sets: within-cluster sum of squares ($J^{\downarrow}$)

Data set	GA-k-means	DE-k-means	PSO-k-means	GSO-k-means	BSA-k-means	FMKGSO	MKGSO	TMKGSO
Disp01	3232.8 ± 1320.6	3468.9 ± 1815.9	**2991.7 ± 0**	**2991.7 ± 0**	3241.3 ± 1367.3	**2991.7 ± 0**	**2991.7 ± 0**	**2991.7 ± 0**
Disp02	7138.9 ± 1831.8	7820.9 ± 2646.7	7125.8 ± 1781.2	6905.3 ± 1355.5	7631.2 ± 2525.1	**6657.8 ± 0**	**6657.8 ± 0**	**6657.8 ± 0**
Disp03	8756.5 ± 1791.9	8602.0 ± 1646.6	8402.1 ± 1275.0	8363.8 ± 1065.8	8386.2 ± 1187.7	**8169.1 ± 0**	**8169.1 ± 0**	**8169.1 ± 0**
Disp04	9809.4 ± 1395.1	10904.6 ± 2466.6	**9442.8 ± 0**	10173.1 ± 1893.8	9825.9 ± 1458.2	**9442.8 ± 0**	**9442.8 ± 0**	**9442.8 ± 0**
Disp05	11942.3 ± 1042.3	12706.0 ± 2175.1	**11752.0 ± 0**	11937.5 ± 1015.7	11923.7 ± 940.7	**11752.0 ± 0**	**11752.0 ± 0**	**11752.0 ± 0**
Prox01	3407.4 ± 1186.0	3700.9 ± 1569.1	**3095.8 ± 0**	**3095.8 ± 0**	3251.8 ± 854.7	**3095.8 ± 0**	**3095.8 ± 0**	**3095.8 ± 0**
Prox02	3226.4 ± 526.8	3266.2 ± 545.1	3188.9 ± 508.8	3227.1 ± 528.0	3382.9 ± 577.1	**2887.3 ± 0**	**2887.3 ± 0**	2924.7 ± 205.2
Prox03	3304.6 ± 594.8	3530.1 ± 544.2	3266.1 ± 498.6	3267.0 ± 500.0	3393.3 ± 553.7	**2919.8 ± 0**	2987.5 ± 257.8	2956.1 ± 199.0
Prox04	3078.7 ± 446.8	3156.4 ± 460.2	3060.6 ± 411.5	3066.8 ± 422.3	2920.8 ± 240.9	**2841.9 ± 0**	2868.0 ± 143.1	2983.6 ± 327.1
Prox05	**2725.4 ± 0.2862**	2773.5 ± 182.9	**2725.4 ± 0.2564**	**2725.3 ± 0.2300**	**2725.4 ± 0.2785**	**2725.2 ± 0**	**2725.2 ± 0**	**2725.2 ± 0**

It means that interchange between the global search offered by EA operators and the local search promoted by K-Means (i.e., the memetic framework) may represent a robust approach to explore and exploit the problem search space, leading to better optimization performances. By the other hand, the use of just one K-Means step for each

Table 13 Experimental results for synthetic data sets: correct Rand index ($CR^{\uparrow}$)

Data set	GA-k-means	DE-k-means	PSO-k-means	GSO-k-means	BSA-k-means	FMKGSO	MKGSO	TMKGSO
Disp01	0.9902 ± 0.0536	0.9805 ± 0.0741	**1.0000 ± 0**	**1.0000 ± 0**	0.9906 ± 0.0512	**1.0000 ± 0**	**1.0000 ± 0**	**1.0000 ± 0**
Disp02	0.9598 ± 0.0682	0.9321 ± 0.1036	0.9601 ± 0.0669	0.9685 ± 0.0500	0.9435 ± 0.0889	**0.9777 ± 0**	**0.9777 ± 0**	**0.9777 ± 0**
Disp03	0.9462 ± 0.0993	0.9615 ± 0.0671	0.9695 ± 0.0499	0.9677 ± 0.0582	0.9696 ± 0.0512	**0.9775 ± 0**	**0.9775 ± 0**	**0.9775 ± 0**
Disp04	0.9370 ± 0.0800	0.8743 ± 0.1412	**0.9580 ± 0**	0.9157 ± 0.1097	0.9368 ± 0.0805	**0.9580 ± 0**	**0.9580 ± 0**	**0.9580 ± 0**
Disp05	0.9469 ± 0.0613	0.9081 ± 0.1154	**0.9581 ± 0**	0.9454 ± 0.0691	0.9469 ± 0.0610	**0.9581 ± 0**	**0.9581 ± 0**	**0.9581 ± 0**
Prox01	0.9765 ± 0.0893	0.9616 ± 0.1000	**1.0000 ± 0**	**1.0000 ± 0**	0.9925 ± 0.0409	**1.0000 ± 0**	**1.0000 ± 0**	**1.0000 ± 0**
Prox02	0.8923 ± 0.1178	0.8839 ± 0.1211	0.9007 ± 0.1138	0.8923 ± 0.1179	0.8605 ± 0.1255	**0.9681 ± 0**	**0.9681 ± 0**	0.9597 ± 0.0463
Prox03	0.8533 ± 0.1294	0.8021 ± 0.1255	0.8588 ± 0.1191	0.8596 ± 0.1179	0.8386 ± 0.1213	**0.9416 ± 0**	0.9253 ± 0.0622	0.9326 ± 0.0495
Prox04	0.8298 ± 0.1068	0.8068 ± 0.1164	0.8276 ± 0.1103	0.8287 ± 0.1050	0.8594 ± 0.0839	**0.8874 ± 0**	0.8781 ± 0.0509	0.8433 ± 0.1005
Prox05	0.8452 ± 0.0055	0.8288 ± 0.0567	0.8435 ± 0.0051	0.8420 ± 0.0064	0.8453 ± 0.0052	**0.8459 ± 0**	**0.8459 ± 0**	**0.8459 ± 0**

Table 14 Experimental results for synthetic data sets: weighted quantization error ($J^{\downarrow}_{e_2}$)

Data set	GA-k-means	DE-k-means	PSO-k-means	GSO-k-means	BSA-k-means	FMKGSO	MKGSO	TMKGSO
Disp01	3.2328 ± 1.3206	3.4689 ± 1.8158	**2.9917 ± 0**	**2.9917 ± 0**	3.2413 ± 1.3673	**2.9917 ± 0**	**2.9917 ± 0**	**2.9917 ± 0**
Disp02	7.1389 ± 1.8318	7.8209 ± 2.6467	7.1258 ± 1.7812	6.9053 ± 1.3555	7.6312 ± 2.5251	**6.6578 ± 0**	**6.6578 ± 0**	**6.6578 ± 0**
Disp03	8.7565 ± 1.7919	8.6020 ± 1.6466	8.4021 ± 1.2750	8.3638 ± 1.0658	8.3862 ± 1.1877	**8.1691 ± 0**	**8.1691 ± 0**	**8.1691 ± 0**
Disp04	9.8094 ± 1.3951	10.905 ± 2.4666	**9.4428 ± 0**	10.173 ± 1.8938	9.8259 ± 1.4582	**9.4428 ± 0**	**9.4428 ± 0**	**9.4428 ± 0**
Disp05	11.942 ± 1.0423	12.706 ± 2.1751	**11.752 ± 0**	11.937 ± 1.0157	11.923 ± 0.9407	**11.752 ± 0**	**11.752 ± 0**	**11.752 ± 0**
Prox01	3.4074 ± 1.1860	3.7009 ± 1.5691	**3.0958 ± 0**	**3.0958 ± 0**	3.2518 ± 0.8547	**3.0958 ± 0**	**3.0958 ± 0**	**3.0958 ± 0**
Prox02	3.2264 ± 0.5268	3.2662 ± 0.5451	3.1889 ± 0.5088	3.2271 ± 0.5280	3.3829 ± 0.5771	**2.8873 ± 0**	**2.8873 ± 0**	2.9247 ± 0.2052
Prox03	3.3046 ± 0.5948	3.5301 ± 0.5442	3.2661 ± 0.4986	3.2670 ± 0.5000	3.3933 ± 0.5537	**2.9198 ± 0**	2.9875 ± 0.2578	2.9561 ± 0.1990
Prox04	3.0787 ± 0.4468	3.1564 ± 0.4602	3.0606 ± 0.4115	3.0668 ± 0.4223	2.9208 ± 0.2409	**2.8419 ± 0**	2.8680 ± 0.1431	2.9836 ± 0.3271
Prox05	**2.7254 ± 0.0003**	2.7735 ± 0.1829	**2.7254 ± 0.0003**	**2.7253 ± 0.0002**	**2.7254 ± 0.0002**	**2.7252 ± 0**	**2.7252 ± 0**	**2.7252 ± 0**

individual from the population in each generation (MKGSO approach), represented the worst combination for the memetic models, in terms of computational costs (measured by the average execution time), but such combination still kept the good performances of all other evaluated memetic approaches (FMKGSO and TMKGSO),

Table 15 Experimental results for synthetic data sets: intra-cluster distance ($D^{\downarrow}_{max}$)

Data set	GA-k-means	DE-k-means	PSO-k-means	GSO-k-means	BSA-k-means	FMKGSO	MKGSO	TMKGSO
Disp01	3.8415 ± 3.2624	4.4236 ± 4.4823	**3.2458 ± 0**	**3.2458 ± 0**	3.8622 ± 3.3763	**3.2458 ± 0**	**3.2458 ± 0**	**3.2458 ± 0**
Disp02	13.375 ± 3.4581	14.280 ± 4.2780	13.378 ± 3.4693	12.921 ± 2.4885	14.133 ± 4.4080	**12.466 ± 0**	**12.466 ± 0**	**12.466 ± 0**
Disp03	12.778 ± 1.9092	12.834 ± 2.6421	12.630 ± 2.5661	12.411 ± 1.3675	12.581 ± 2.3002	**12.162 ± 0**	**12.162 ± 0**	**12.162 ± 0**
Disp04	12.033 ± 1.9258	**13.876 ± 4.0761**	11.527 ± 0	12.576 ± 2.7213	12.225 ± 2.6688	11.527 ± 0	11.527 ± 0	11.527 ± 0
Disp05	13.070 ± 2.0900	14.538 ± 4.2234	**12.689 ± 0**	12.997 ± 1.6910	13.082 ± 2.1547	**12.689 ± 0**	**12.689 ± 0**	**12.689 ± 0**
Prox01	3.8775 ± 2.2080	4.7609 ± 3.7982	**3.2974 ± 0**	**3.2974 ± 0**	3.6763 ± 2.0760	**3.2974 ± 0**	**3.2974 ± 0**	**3.2974 ± 0**
Prox02	4.0095 ± 1.2998	4.1025 ± 1.3371	3.9165 ± 1.2543	4.0095 ± 1.2998	4.3813 ± 1.4055	**3.1729 ± 0**	**3.1729 ± 0**	3.2658 ± 0.5092
Prox03	3.9949 ± 1.3633	4.6991 ± 1.4728	3.9949 ± 1.3633	3.9949 ± 1.3633	4.3013 ± 1.4605	**3.0491 ± 0**	3.2318 ± 0.6953	3.1512 ± 0.5595
Prox04	3.5836 ± 1.0632	3.7080 ± 1.0301	3.5448 ± 0.9919	3.5292 ± 0.9737	3.1859 ± 0.4594	**3.0354 ± 0**	3.0927 ± 0.3139	3.3622 ± 0.7700
Prox05	2.8617 ± 0.0056	2.9290 ± 0.2535	2.8609 ± 0.0053	2.8612 ± 0.0055	2.8612 ± 0.0054	**2.8574 ± 0**	**2.8574 ± 0**	**2.8574 ± 0**

Table 16 Experimental results for synthetic data sets: inter-cluster separation ($D^{\uparrow}_{min}$)

Data set	GA-k-means	DE-k-means	PSO-k-means	GSO-k-means	BSA-k-means	FMKGSO	MKGSO	TMKGSO
Disp01	70.431 ± 12.756	68.121 ± 17.653	**72.760 ± 0**	**72.760 ± 0**	70.417 ± 12.825	**72.760 ± 0**	**72.760 ± 0**	**72.760 ± 0**
Disp02	70.993 ± 17.191	64.584 ± 24.870	71.257 ± 16.153	73.106 ± 13.117	68.622 ± 21.143	**75.501 ± 0**	**75.501 ± 0**	**75.501 ± 0**
Disp03	65.536 ± 18.336	70.374 ± 12.010	69.540 ± 11.051	69.607 ± 10.889	69.506 ± 10.952	**71.723 ± 0**	**71.723 ± 0**	**71.723 ± 0**
Disp04	67.101 ± 13.943	55.316 ± 26.032	**70.749 ± 0**	62.980 ± 20.149	66.824 ± 14.937	**70.749 ± 0**	**70.749 ± 0**	**70.749 ± 0**
Disp05	71.039 ± 11.266	70.418 ± 15.102	**73.096 ± 0**	71.290 ± 9.8895	71.105 ± 10.899	**73.096 ± 0**	**73.096 ± 0**	**73.096 ± 0**
Prox01	44.267 ± 11.207	42.920 ± 13.593	**47.211 ± 0**	**47.211 ± 0**	**47.235 ± 0.1302**	**47.211 ± 0**	**47.211 ± 0**	**47.211 ± 0**
Prox02	9.9199 ± 4.5234	9.5492 ± 4.7216	10.244 ± 4.3635	9.8985 ± 4.5569	10.010 ± 8.1720	**12.831 ± 0**	**12.831 ± 0**	12.514 ± 1.7377
Prox03	8.9750 ± 4.0264	7.6806 ± 4.2878	8.9680 ± 4.0372	8.9385 ± 4.0787	9.0281 ± 4.0169	**11.774 ± 0**	11.205 ± 2.1630	11.497 ± 1.5176
Prox04	10.065 ± 3.2354	9.8634 ± 3.6289	9.6850 ± 3.6083	10.000 ± 3.4498	10.838 ± 2.4424	**11.638 ± 0**	11.365 ± 1.4917	10.253 ± 3.1510
Prox05	11.617 ± 0.0477	**12.002 ± 1.5962**	11.582 ± 0.0650	11.536 ± 0.0531	11.608 ± 0.0566	11.537 ± 0	11.537 ± 0	11.537 ± 0

showing its potential in terms of performance (considering all five selected clustering metrics) when dealing with data clustering problems, in comparison to all other state-of-the-art non-memetic hybrid algorithms. In a future investigation, some mechanisms, like a selection process on the population individuals that would be improved by K-Means in each GSO generation, would be performed in an attempt to improve MKGSO computational costs.

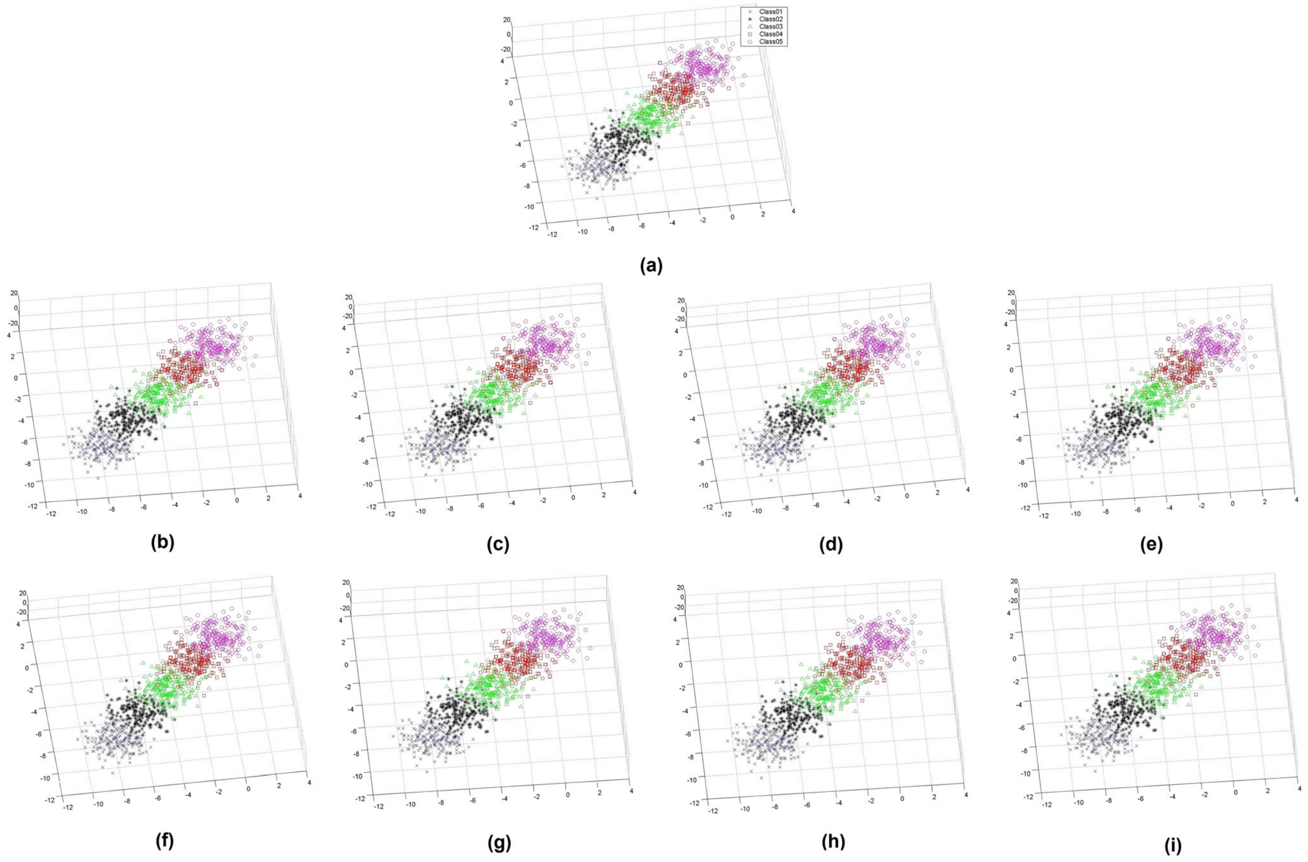

Fig. 4 Results for Prox05 data set: **a** Original data set, **b** GA-k-means, **c** DE-k-means, **d** PSO-k-means, **e** GSO-k-means, **f** BSA-k-means, **g** FMKGSO, **h** MKGSO, **i** TMKGSO

Table 17 Overall evaluation on synthetic data sets: average Friedman-Nemenyi ranks (and position) for each metric. The critical distance is CD = 3.3202. The best results are bold faced

Algorithm	$J^{\downarrow}$	$CR^{\uparrow}$	$J_{e_2}^{\downarrow}$	$D_{max}^{\downarrow}$	$D_{min}^{\uparrow}$	Total
GA-k-means	134.1333 (6)	119.8133 (6)	134.1333 (6)	126.6117 (6)	119.5967 (4)	24
DE-k-means	146.6967 (8)	106.8250 (8)	146.6950 (8)	138.5300 (8)	110.5567 (7)	39
PSO-k-means	127.0100 (4)	120.5133 (4)	127.0100 (4)	121.9817 (4)	119.0733 (5)	21
GSO-k-means	127.7867 (5)	110.0250 (7)	127.7883 (5)	125.5100 (5)	108.9217 (8)	30
BSA-k-means	136.6383 (7)	119.9783 (5)	136.6383 (7)	126.8050 (7)	118.5300 (6)	32
FMKGSO	**95.7500 (1)**	**130.4500 (1)**	**95.7500 (1)**	**106.8500 (1)**	**130.2500 (1)**	**5**
MKGSO	**96.2650 (2)**	**129.9283 (2)**	**96.2650 (2)**	**107.2517 (2)**	**129.8117 (2)**	10
TMKGSO	99.7200 (3)	126.4667 (3)	99.7200 (3)	110.4600 (3)	**127.2600 (3)**	15

As future works, we intend to evaluate the influence of the adopted distance function on the behavior of the proposed memetic algorithms, so the proposed approaches would be more robust to deal with clusters with different formats and shapes, overcoming some limitations of the standard Euclidean distance. Also, new fitness functions will be introduced and tested, using a larger testing bed, that will be obtained by the development of alternative synthetic data set configurations and scenarios. Finally, mechanisms for the automatic determination of the best number of clusters will be implemented, in such a way that such parameter would be estimated by the memetic evolutionary model itself, instead of being provided as an *a priori* input parameter to the algorithm.

Table 18 Average execution times for the synthetic data sets (in seconds)

Data set	GA-k-means	DE-k-means	PSO-k-means	GSO-k-means	BSA-k-means	FMKGSO	MKGSO	TMKGSO
Disp01	0.8063	0.7808	0.7798	0.9301	0.7955	0.7533	3.2231	0.5825
Disp02	0.7987	0.7803	0.7792	0.9301	0.7988	0.8456	3.2337	0.6636
Disp03	0.7968	0.7810	0.7832	0.9303	0.7990	0.9501	3.2367	0.7013
Disp04	0.8013	0.7821	0.7828	0.9320	0.8014	0.9412	3.2257	0.7482
Disp05	0.8005	0.7832	0.7817	0.9344	0.8016	1.0081	3.2252	0.8167
Prox01	0.7945	0.7813	0.7819	0.9320	0.7985	0.7967	3.2203	0.6293
Prox02	0.7999	0.7817	0.7782	0.9353	0.7990	0.9621	3.2320	0.7646
Prox03	0.8027	0.7832	0.7791	0.9371	0.8004	1.1342	3.2330	0.8668
Prox04	0.8018	0.7824	0.7819	0.9328	0.7988	1.2774	3.2399	0.9039
Prox05	0.8059	0.7888	0.7821	0.9352	0.8026	1.3448	3.2589	0.9195

Table 19 Overall evaluation on all data sets: average Friedman-Nemenyi ranks (and position) for each metric. The critical distance is CD = 1.9169. The best results are bold faced

Algorithm	$J^{\downarrow}$	$CR^{\uparrow}$	$J_{e_2}^{\downarrow}$	$D_{max}^{\downarrow}$	$D_{min}^{\uparrow}$	Total
GA-k-means	138.7683 (8)	115.0283 (4)	138.7700 (8)	122.9222 (8)	111.0200 (5)	33
DE-k-means	135.9272 (6)	112.8411 (8)	135.9267 (6)	120.8350 (4)	114.7439 (4)	28
PSO-k-means	134.4761 (5)	114.4233 (7)	134.4761 (5)	121.6556 (6)	109.6944 (7)	30
GSO-k-means	130.6500 (4)	114.9361 (5)	130.6483 (4)	120.8617 (5)	108.4417 (8)	26
BSA-k-means	136.5994 (7)	114.7850 (6)	136.5989 (7)	122.6617 (7)	110.7883 (6)	33
FMKGSO	95.7039 (2)	128.4678 (3)	95.7756 (2)	119.3478 (3)	**142.2306 (1)**	11
MKGSO	99.1578 (3)	**131.4106 (2)**	99.1750 (3)	118.9633 (2)	134.2667 (2)	12
TMKGSO	**92.7172 (1)**	**132.1078 (1)**	**92.6294 (1)**	**116.7528 (1)**	132.8144 (3)	**7**

Acknowledgements The authors would like to thank FACEPE, CNPq and CAPES (Brazilian Research Agencies) for their financial support.

References

Abdel-Kader RF (2010) Genetically improved pso algorithm for efficient data clustering. In: Machine Learning and Computing (ICMLC), 2010 Second International Conference on, pp. 71–75. IEEE

Abualigah L (2020) Group search optimizer: a nature-inspired meta-heuristic optimization algorithm with its results, variants, and applications. Neural Computing and Applications pp. 1–24

Ahmadi A, Karray F, Kamel MS (2010) Flocking based approach for data clustering. Nat Comput 9(3):767–791

Ahmadyfard A, Modares H (2008) Combining pso and k-means to enhance data clustering. In: Telecommunications, 2008. IST 2008. International Symposium on, pp. 688–691. IEEE

Akbari M, Izadkhah H (2019) Gakh: A new evolutionary algorithm for graph clustering problem. In: 2019 4th International Conference on Pattern Recognition and Image Analysis (IPRIA), pp. 159–162. IEEE

Arabie P, Hubert LJ, De Soete G (1996) Clustering and classification. World Scientific, Singapore

Asuncion A, Newman D (2007) Uci machine learning repository

Barnard C, Sibly R (1981) Producers and scroungers: a general model and its application to captive flocks of house sparrows. Anim Behav 29(2):543–550

BEDDAD B, HACHEMI K, POSTAIRE JG, JABLONCIK F, MESSAI O (2019) An improvement of spatial fuzzy c-means clustering method for noisy medical image analysis. In: 2019 6th International Conference on Image and Signal Processing and their Applications (ISPA), pp. 1–5. IEEE

Bhavani R, Sadasivam GS, Kumaran R (2011) A novel parallel hybrid k-means-de-aco clustering approach for genomic clustering using mapreduce. In: Information and Communication Technologies (WICT), 2011 World Congress on, pp. 132–137. IEEE

Bonabeau E, Dorigo M, Theraulaz G (1999) Swarm intelligence: from natural to artificial systems, vol 4. Oxford University Press, New York

Bruse JL, Zuluaga MA, Khushnood A, McLeod K, Ntsinjana HN, Hsia TY, Sermesant M, Pennec X, Taylor AM, Schievano S (2017) Detecting clinically meaningful shape clusters in medical image data: metrics analysis for hierarchical clustering applied to healthy and pathological aortic arches. IEEE Trans Biomed Eng 64(10):2373–2383

Canuto A, Neto AF, Silva HM, Xavier-Júnior JC, Barreto CA (2018) Population-based bio-inspired algorithms for cluster ensembles optimization. Natural Computing pp. 1–18

Chen CY, Ye F (2004) Particle swarm optimization algorithm and its application to clustering analysis. In: Networking, Sensing and Control, 2004 IEEE International Conference on, vol. 2, pp. 789–794. IEEE

Chen G, Luo W, Zhu T (2014) Evolutionary clustering with differential evolution. In: Evolutionary Computation (CEC), 2014 IEEE Congress on, pp. 1382–1389. IEEE

Chen J, Zheng J, Liu Y, Wu Q (2014) Dynamic economic dispatch with wind power penetration using group search optimizer with adaptive strategies. In: IEEE PES Innovative Smart Grid Technologies, Europe, pp. 1–6. IEEE

Cho PPW, Nyunt TTS (2020) Data clustering based on differential evolution with modified mutation strategy. In: 2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), pp. 222–225. IEEE

Civicioglu P (2013) Backtracking search optimization algorithm for numerical optimization problems. Appl Math Comput 219(15):8121–8144

Couzin ID, Krause J, Franks NR, Levin SA (2005) Effective leadership and decision-making in animal groups on the move. Nature 433(7025):513–516

Cui X, Potok TE, Palathingal P (2005) Document clustering using particle swarm optimization. In: Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005., pp. 185–191. IEEE

Darwish A, Ezzat D, Hassanien AE (2020) An optimized model based on convolutional neural networks and orthogonal learning particle swarm optimization algorithm for plant diseases diagnosis. Swarm Evol Comput 52:100616

Das S, Abraham A, Konar A (2007) Automatic clustering using an improved differential evolution algorithm. IEEE Trans Syst Man Cybern Part A Syst Hum 38(1):218–237

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

Dhiviya S, Sariga A, Sujatha P (2017) Survey on wsn using clustering. In: 2017 Second International Conference on Recent Trends and Challenges in Computational Models (ICRTCCM), pp. 121–125. IEEE

Diderot PKG, Vasudevan N, Sankaran KS (2019) An efficient fuzzy c-means clustering based image dissection algorithm for satellite images. In: 2019 International Conference on Communication and Signal Processing (ICCSP), pp. 0806–0809. IEEE

Dixon A (1959) An experimental study of the searching behaviour of the predatory coccinellid beetle adalia decempunctata (l.). The Journal of Animal Ecology pp. 259–281

Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. Syst Man Cybern Part B Cybern IEEE Trans 26(1):29–41

Eiben AE, Smith JE (2015) Introduction to evolutionary computing. Springer, Berlin

Elaziz MA, Nabil N, Ewees AA, Lu S (2019) Automatic data clustering based on hybrid atom search optimization and sine-cosine algorithm. In: 2019 IEEE Congress on Evolutionary Computation (CEC), pp. 2315–2322. IEEE

Esmin AAA, Pereira DL, De Araujo F (2008) Study of different approach to clustering data by using the particle swarm optimization algorithm. In: Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on, pp. 1817–1822. IEEE

Figueiredo E, Macedo M, Siqueira HV, Santana CJ Jr, Gokhale A, Bastos-Filho CJ (2019) Swarm intelligence for clustering-a systematic review with new perspectives on data mining. Eng Appl Artif Intell 82:313–329

Fogel D (2009) Artificial intelligence through simulated evolution. Wiley-IEEE Press, New Jersy

Fogel DB (2006) Evolutionary computation: toward a new philosophy of machine intelligence, vol 1. Wiley, New Jersy

Fogel LJ, Owens AJ, Walsh MJ (1966) Artificial intelligence through simulated evolution. Wiley, New Jersy

Friedman M (1937) The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J Am Stat Assoc 32(200):675–701

Friedman M (1940) A comparison of alternative tests of significance for the problem of m rankings. Ann Math Stat 11(1):86–92

Geem ZW (2010) Recent advances in harmony search algorithm, vol 270. Springer, Berlin

Z.W Geem, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation 76(2):60–68

Günen MA, Atasever ÜH, Beşdok E (2017) A novel edge detection approach based on backtracking search optimization algorithm (bsa) clustering. In: 2017 8th International Conference on Information Technology (ICIT), pp. 116–122. IEEE

Hassan BA, Rashid TA (2020) Operational framework for recent advances in backtracking search optimisation algorithm: a systematic review and performance evaluation. Appl Math Comput 370:124919

He H, Tan Y (2012) A two-stage genetic algorithm for automatic clustering. Neurocomputing 81:49–59

He S, Wu Q, Saunders J (2006) A novel group search optimizer inspired by animal behavioural ecology. In: 2006 IEEE Congress on Evolutionary Computation (CEC), pp. 1272–1278. IEEE

He S, Wu QH, Saunders JR (2009) Group search optimizer: an optimization algorithm inspired by animal searching behavior. IEEE Trans Evol Comput 13(5):973–990

Higgins CL, Strauss RE (2004) Discrimination and classification of foraging paths produced by search-tactic models. Behav Ecol 15(2):248–254

Holland JH (1992) Genetic algorithms. Sci Am 267(1):66–72

Hruschka ER, Campello RJ, Freitas AA et al (2009) A survey of evolutionary algorithms for clustering. IEEE Trans Syst Man Cybern Part C Appl Rev 39(2):133–155

Hubert L, Arabie P (1985) Comparing partitions. J Classif 2(1):193–218

Idrissi MAJ, Ramchoun H, Ghanou Y, Ettaouil M (2016) Genetic algorithm for neural network architecture optimization. In: 2016 3rd International Conference on Logistics Operations Management (GOL), pp. 1–4. IEEE

Inkaya T, Kayalıgil S, Özdemirel NE (2016) Swarm intelligence-based clustering algorithms: A survey. In: Unsupervised learning algorithms, pp. 303–341. Springer

Islam MT, Basak PK, Bhowmik P, Khan M (2019) Data clustering using hybrid genetic algorithm with k-means and k-medoids algorithms. In: 2019 23rd International Computer Science and Engineering Conference (ICSEC), pp. 123–128. IEEE

Jain M, Singh V, Rani A (2019) A novel nature-inspired algorithm for optimization: Squirrel search algorithm. Swarm Evol Comput 44:148–175

José-García A, Gómez-Flores W (2016) Automatic clustering using nature-inspired metaheuristics: a survey. Appl Soft Comput 41:192–213

Junaed A, Akhand M, Murase K, et al (2013) Multi-producer group search optimizer for function optimization. In: 2013 International Conference on Informatics, Electronics and Vision (ICIEV), pp. 1–4. IEEE

Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. J Global Optim 39(3):459–471

Kennedy J (2006) Swarm intelligence. Handbook of nature-inspired and innovative computing. Springer, Berlin, pp 187–219

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Neural Networks, 1995. Proceedings., IEEE International Conference on, vol. 4, pp. 1942–1948. IEEE

Kennedy J, Eberhart RC, Shi Y (2001) Swarm intelligence. Kaufmann, San Francisco

Koza JR, Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection, vol 1. MIT press, Cambridge

Krishnaprabha R, Aloor G (2014) Group search optimizer algorithm in wireless sensor network localization. In: 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1953–1957. IEEE

Latiff NA, Malik NNA, Idoumghar L (2016) Hybrid backtracking search optimization algorithm and k-means for clustering in wireless sensor networks. In: 2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), pp. 558–564. IEEE

Li L, Liang Y, Li T, Wu C, Zhao G, Han X (2019) Boost particle swarm optimization with fitness estimation. Nat Comput 18(2):229–247

Li L, Xu S, Wang S, Ma X (2016) The diseases clustering for multi-source medical sets. In: 2016 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI), pp. 294–298. IEEE

Li T, Dong H (2019) Unsupervised feature selection and clustering optimization based on improved differential evolution. IEEE Access 7:140438–140450

Li Xl (2002) An optimizing method based on autonomous animats: fish-swarm algorithm. Syst Eng Theory Pract 22(11):32–38

Li Yz, Zheng Xw, Lu Dj (2015) Virtual network embedding based on multi-objective group search optimizer. In: 2015 10th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA), pp. 598–601. IEEE

Li Z, Hu Z, Miao Y, Xiong Z, Xu X, Dai C (2019) Deep-mining backtracking search optimization algorithm guided by collective wisdom. Mathematical Problems in Engineering 2019

Lin CJ, Huang ML (2019) Efficient hybrid group search optimizer for assembling printed circuit boards. AI EDAM 33(3):259–274

Liu F, Xiong L (2011) Survey on text clustering algorithm-research present situation of text clustering algorithm. In: 2011 IEEE 2nd International Conference on Software Engineering and Service Science, pp. 196–199. IEEE

Liu Y, Wu X, Shen Y (2011) Automatic clustering using genetic algorithms. Appl Math Comput 218(4):1267–1279

MacQueen J, et al (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, vol. 1, pp. 281–297. California, USA

Masoud MZ, Jaradat Y, Zaidan D, Jannoud I (2019) To cluster or not to cluster: A hybrid clustering protocol for wsn. In: 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), pp. 678–682. IEEE

Miranda PB, Prudêncio RB (2018) A novel context-free grammar for the generation of pso algorithms. Natural Computing pp. 1–19

Mirjalili S (2016) Sca: a sine cosine algorithm for solving optimization problems. Knowl-Based Syst 96:120–133

Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67

Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61

Misra S, Kumar R (2016) A literature survey on various clustering approaches in wireless sensor network. In: 2016 2nd international conference on communication control and intelligent systems (CCIS), pp. 18–22. IEEE

Mortezanezhad A, Daneshifar E (2019) Big-data clustering with genetic algorithm. In: 2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI), pp. 702–706. IEEE

Naldi MC, Campello RJGB (2014) Evolutionary k-means for distributed data sets. Neurocomputing 127:30–42

Nemenyi P (1962) Distribution-free multiple comparisons. Biometrics 18(2):263

Niu B, Duan Q, Liu J, Tan L, Liu Y (2017) A population-based clustering technique using particle swarm optimization and k-means. Nat Comput 16(1):45–59

Oliveira JFL, Pacifico LDS, Ludermir TB (2013) A hybrid group search optimization based on fish swarms. In: 2013 Brazilian Conference on Intelligent Systems, pp. 51–56. IEEE

Pacifico LDS, Ludermir TB (2013) Cooperative group search optimization. In: 2013 IEEE Congress on Evolutionary Computation, pp. 3299–3306. IEEE

Pacifico LDS, Ludermir TB (2014) A group search optimization method for data clustering. In: Intelligent Systems (BRACIS), 2014 Brazilian Conference on, pp. 342–347. IEEE

Pacifico LDS, Ludermir TB (2014) Improved cooperative group search optimization based on divide-and-conquer strategy. In: 2014 Brazilian Conference on Intelligent Systems, pp. 420–425. IEEE

Pacifico LDS, Ludermir TB (2016) Data clustering using group search optimization with alternative fitness functions. In: 2016 5th Brazilian Conference on Intelligent Systems (BRACIS), pp. 301–306. IEEE

Pacifico LDS, Ludermir TB (2018) Hybrid k-means and improved group search optimization methods for data clustering. In: 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE

Pacifico LDS, Ludermir TB (2019) Hybrid k-means and improved self-adaptive particle swarm optimization for data clustering. In: 2019 International Joint Conference on Neural Networks (IJCNN), pp. 1–7. IEEE

Pacifico LDS, Ludermir TB (2019) A partitional cooperative coevolutionary group search optimization approach for data clustering. In: 2019 8th Brazilian Conference on Intelligent Systems (BRACIS), pp. 347–352. IEEE

Pacifico LDS, Ludermir TB, Britto LFS (2018) A hybrid improved group search optimization and otsu method for color image segmentation. In: 2018 7th Brazilian Conference on Intelligent Systems (BRACIS), pp. 296–301. IEEE

Pacifico LDS, Ludermir TB, Oliveira JFL (2018) Evolutionary elms with alternative treatments for the population out-bounded individuals. In: 2018 7th Brazilian Conference on Intelligent Systems (BRACIS), pp. 151–156. IEEE

Parimalam T, Sundaram KM (2017) Efficient clustering techniques for web services clustering. In: 2017 ieee international conference on computational intelligence and computing research (iccic), pp. 1–4. IEEE

Passino KM (2002) Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Syst Mag 22(3):52–67

Prabha KA, Visalakshi NK (2014) Improved particle swarm optimization based k-means clustering. In: 2014 International Conference on Intelligent Computing Applications, pp. 59–63. IEEE

Premalatha P, Subasree S (2017) Performance analysis of clustering algorithms in medical datasets. In: 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT), pp. 1–6. IEEE

Rahamathunnisa U, Nallakaruppan M, Anith A, Kumar KS (2020) Vegetable disease detection using k-means clustering and svm. In: 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), pp. 1308–1311. IEEE

Ramos AC, Vellasco M (2018) Quantum-inspired evolutionary algorithm for feature selection in motor imagery eeg classification. In: 2018 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. IEEE

Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) Gsa: a gravitational search algorithm. Inf Sci 179(13):2232–2248

Rechenberg I (1973) Evolution strategy: optimization of technical systems by means of biological evolution. Fromman Holzboog Stuttgart 104:15–16

Ren Z, Zhang A, Wen C, Feng Z (2014) A scatter learning particle swarm optimization algorithm for multimodal problems. Cybern IEEE Trans 44(7):1127–1140

Sapkota N, Alsadoon A, Prasad P, Elchouemi A, Singh AK (2019) Data summarization using clustering and classification: Spectral clustering combined with k-means using nfph. In: 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), pp. 146–151. IEEE

Saraswathi S, Allirani A (2013) Survey on image segmentation via clustering. In: 2013 International Conference on Information Communication and Embedded Systems (ICICES), pp. 331–335. IEEE

Saremi S, Mirjalili S, Lewis A (2017) Grasshopper optimisation algorithm: theory and application. Adv Eng Softw 105:30–47

Schwefel HPP (1993) Evolution and optimum seeking: the sixth generation. Wiley, New Jersy

Shi H, Xu M (2018) A data classification method using genetic algorithm and k-means algorithm with optimizing initial cluster center. In: 2018 IEEE International Conference on Computer and Communication Engineering Technology (CCET), pp. 224–228. IEEE

Silva DNG, Pacifico LDS, Ludermir TB (2011) An evolutionary extreme learning machine based on group search optimization. In: 2011 IEEE Congress of Evolutionary Computation (CEC), pp. 574–580. IEEE

Simon D (2013) Evolutionary optimization algorithms. Wiley, New Jersy

Souza E, Santos D, Oliveira G, Silva A, Oliveira AL (2018) Swarm optimization clustering methods for opinion mining. Natural computing pp. 1–29

Sreepathi S, Kumar J, Mills RT, Hoffman FM, Sripathi V, Hargrove WW (2017) Parallel multivariate spatio-temporal clustering of large ecological datasets on hybrid supercomputers. In: 2017 IEEE International Conference on Cluster Computing (CLUSTER), pp. 267–277. IEEE

Storn R, Price K (1995) Differential evolution–a simple and efficient adaptive scheme for global optimization over continuous spaces. international computer science institute, berkeley. Tech. rep., CA, 1995, Tech. Rep. TR-95–012

Storn R, Price K (1997) Differential evolution-a simple, efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359

Taj N, Basu A (2019) Hybridization of genetic and group search optimization algorithm for deadline-constrained task scheduling approach. J Intell Syst 28(1):153–171

Taşci E, Gökalp O, Uğur A (2018) Development of a novel feature weighting method using cma-es optimization. In: 2018 26th Signal Processing and Communications Applications Conference (SIU), pp. 1–4. IEEE

Toman SH, Abed MH, Toman ZH (2020) Cluster-based information retrieval by using (k-means)-hierarchical parallel genetic algorithms approach. arXiv preprint arXiv:2008.00150

Toz G, Yücedağ İ, Erdoğmuş P (2019) A fuzzy image clustering method based on an improved backtracking search optimization algorithm with an inertia weight parameter. J King Saud Univ Comput Inf Sci 31(3):295–303

Wan C, Ye M, Yao C, Wu C (2017) Brain mr image segmentation based on gaussian filtering and improved fcm clustering algorithm. In: 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), pp. 1–5. IEEE

Wang F (2018) A weighted k-means algorithm based on differential evolution. In: 2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), pp. 1–2274. IEEE

Wang H, Zuo L, Liu J, Yi W, Niu B (2018) Ensemble particle swarm optimization and differential evolution with alternative mutation method. Natural Computing pp. 1–14

Wei Y, Niu C, Wang H, Liu D (2019) The hyperspectral image clustering based on spatial information and spectral clustering. In: 2019 IEEE 4th International Conference on Signal and Image Processing (ICSIP), pp. 127–131. IEEE

Wong MT, He X, Yeh WC (2011) Image clustering using particle swarm optimization. In: Evolutionary Computation (CEC), 2011 IEEE Congress on, pp. 262–268. IEEE

Xu D, Tian Y (2015) A comprehensive survey of clustering algorithms. Ann Data Sci 2(2):165–193

Xu H, Xue B, Zhang M (2020) A duplication analysis based evolutionary algorithm for bi-objective feature selection. IEEE Transactions on Evolutionary Computation

Xu Y, Shu Y (2006) Evolutionary extreme learning machine-based on particle swarm optimization. International Symposium on Neural Networks. Springer, Berlin, pp 644–652

Yang XS (2009) Firefly algorithms for multimodal optimization. In: International symposium on stochastic algorithms, pp. 169–178. Springer

Yang XS (2010) Firefly algorithm, levy flights and global optimization. In: Research and development in intelligent systems XXVI, pp. 209–218. Springer

Zhang M, Cao J (2020) An elitist-based differential evolution algorithm for multiobjective clustering. In: 2020 3rd International Conference on Artificial Intelligence and Big Data (ICAIBD), pp. 161–166. IEEE

Zhu L, Ma Y, Bai Y (2020) A self-adaptive multi-population differential evolution algorithm. Nat Comput 19(1):211–235