Data Clustering Using Group Search Optimization with Alternative Fitness Functions

Luciano D. S. Pacifico[1,2]
Departamento de Estatistica e Informatica - DEInfo[1]
Universidade Federal Rural de Pernambuco - UFRPE
Recife, Pernambuco, Brazil
Email: ldsp@deinfo.ufrpe.br[1], ldsp@cin.ufpe.br[2]

Teresa B. Ludermir[2]
Centro de Informatica - CIn[2]
Universidade Federal de Pernambuco - UFPE
Recife, Pernambuco, Brazil
Email: tbl@cin.ufpe.br[2]

***Abstract*—Data clustering is an important tool for statistical data analysis and exploration, and it has been successfully applied in many fields like image understanding, bioinformatics, big data mining, and so on. From the past few decades, Evolutionary Algorithms (EAs) have been introduced to deal with clustering task, given their global search capabilities and their mechanisms to escape from local minima points. EAs execution is driven in an attempt to optimize a criterion function, also known as fitness function. In this work, we evaluate the influence of the fitness function on Group Search Optimization (GSO) meta-heuristic when applied to data clustering. Three different fitness function are proposed to GSO. Experiments are performed on twelve benchmark data sets obtained from UCI Machine Learning Repository to evaluate the performance of all alternative GSO models in comparison to other well-known partitional clustering methods from literature.**

I. Introduction

The amount of data daily produced from the past years has grown exponentially. Systems based on human analysis only are unpractical in real life applications, given that the need for precise and reliable information in a short period of time has become mandatory. Fast and robust computational techniques are required to extract relevant information from raw big data sets automatically [1].

As one of the most primitive pattern recognition tasks, data clustering, also referred as unsupervised learning, consists in an attempt to separate a set of observations in groups (clusters) according to their inner properties (similarities and dissimilarities). It is expected that observations belonging in the same cluster present a higher degree of similarity than observations belonging in different clusters [2]. No prior knowledge about the data patterns to be clustered is required. Many real world applications have been developed employing clustering techniques as the main tool for exploratory data analysis in many fields, such as medicine, social sciences, engineering, and so on.

Clustering methods are categorized in two main groups: hierarchical and partitional. Hierarchical clustering methods provide a series of nested partitions of the data set based on an iterative process. Partitional methods provide a partition of the data set into a prefixed number of clusters resulting from an attempt to minimize a criterion function [3], [4].

One of the most popular clustering algorithm is the partitional K-Means [5]. K-Means is a two-steps method for real-valued data that employs the Euclidean distance as a dissimilarity measure. The final solution furnished by K-Means is represented by the centroid vectors obtained for each cluster. In K-Means, centroid vectors are computed as the mean value of the data patterns currently associated to a cluster. K-Means main drawback is related to the fact that its convergence and final solutions depend on the initial partition generated. Since the initial partition is obtained generally for a random process, if the initial clusters are not good, the quality of the final partition may be dramatically affected for a bad starting point.

Clustering task can be seen, from an optimization perspective, as a NP-hard combinatorial optimization problem [6]. Many optimization meta-heuristics have been applied in literature to tackle clustering problem, such as Evolutionary Algorithms (EAs) and Swarm Intelligence (SI) methods [2], [7]. In both EAs and SIs, a set (*population*) of candidate solutions for the problem at hand is kept and evolved according to evolutionary operators, in an attempt to optimize an objective function (*fitness function*). Examples of EAs are Genetic Algorithm (GA) [8] and Differential Evolution (DE) [9], [10]. Exemplas of SI methods are Ant Colony Optimization (ACO) [11] and Particle Swarm Optimization (PSO) [12]. EAs and SI algorithms are known for their capabilities to deal with complex optimization scenarios, and their abilities to escape from local minima points from the problem search space.

Since the fitness function is the only measure guiding the search performed by the generational process employed by EAs and SIs, it must be designed as a good representation for the problem at hands, aggregating as much contextual information as possible, so the final solutions furnished by the adopted technique could be considered acceptable according to the limitations of the intended application. If the adopted fitness function is not adequate, the final solution obtained by the meta-heuristic may be compromised.

In this work, we evaluate the influence of the fitness function on the behavior of Group Search Optimization (GSO) meta-heuristic [13] when dealing with clustering task. GSO is employed as a partitional clustering algorithm for real-valued data. Three fitness functions are proposed and applied to GSO. This work is organized as follows. Section II presents briefly GSO algorithm. Next (Section III), the proposed GSO

algorithm for partitional clustering and three alternative fitness functions are described. Experimental results are shown in Section IV. Section V presents the conclusions and some tendencies for future works.

II. Group Search Optimization (GSO)

GSO is a Swarm Intelligence meta-heuristic inspired by animal social searching behavior and group living theory. GSO employs the Producer-Scrounger (PS) model as a framework [14].

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 has a current position $\vec{X}_i^t \in \Re^n$ and a head angle $\vec{\alpha}_i^t \in \Re^{n-1}$. The search direction of the i-th member, which is a vector $\vec{D}_i^t(\vec{\alpha}_i^t) = (d_{i1}^t, \ldots, d_{in}^t)$ can be calculated from $\vec{\alpha}_i^t$ via a polar to Cartesian coordinate transformation:

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

A group in GSO consists of three types of members: producers, scroungers and dispersed members (or $rangers$) [13].

During each GSO search iteration, a group member which has found the best fitness value so far (most promising area) is chosen as the producer ($\vec{X}_p$) [15], and the remaining members are scroungers or rangers. The producer employs a scanning strategy (*producing*) based on its vision field. In GSO, at the t-th iteration the producer $\vec{X}_p^t$ will scan laterally by randomly sampling three points in the scanning field: one at zero degree (eq. (2)), one in the right hand side hypercube (eq. (3)) and one in the left hand side hypercube (eq. (4)).

$$\vec{X}_z = \vec{X}_p^t + r_1 l_{max} \vec{D}_p^t(\vec{\alpha}_p^t) \quad (2)$$

$$\vec{X}_r = \vec{X}_p^t + r_1 l_{max} \vec{D}_p^t(\vec{\alpha}_p^t + \frac{\vec{r}_2 \theta_{max}}{2}) \quad (3)$$

$$\vec{X}_l = \vec{X}_p^t + r_1 l_{max} \vec{D}_p^t(\vec{\alpha}_p^t - \frac{\vec{r}_2 \theta_{max}}{2}) \quad (4)$$

where $r_1 \in \Re$ is a normally distributed random number (mean 0 and standard deviation 1), $\vec{r}_2 \in \Re^{n-1}$ is a uniformly distributed random sequence in the range (0, 1), $\theta_{max} \in \Re^{n-1}$ is a maximum pursuit angle and $l_{max} \in \Re$ is a maximum pursuit distance given by eq. (5):

$$l_{max} = \left\| \vec{U} - \vec{L} \right\| = \sqrt{\sum_{k=1}^{n} (U_k - L_k)^2} \quad (5)$$

where U_k and L_k denote the upper and lower bounds for the k-ith dimension, respectively.

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. (6)).

$$\vec{\alpha}_p^{t+1} = \vec{\alpha}_p^t + \vec{r}_2 \beta_{max} \quad (6)$$

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

If after $a \in \aleph$ iterations the producer cannot find a better area, it will turn its head back to zero degree (eq. (7)).

$$\vec{\alpha}_p^{t+a} = \vec{\alpha}_p^t \quad (7)$$

All scroungers will join the resource found by the producer, performing *scrounging* strategy according to eq. (8).

$$\vec{X}_i^{t+1} = \vec{X}_i^t + \vec{r}_3 \circ (\vec{X}_p^t - \vec{X}_i^t) \quad (8)$$

where $\vec{r}_3 \in \Re^n$ 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 (*ranging*) [16], according to eq. (9).

$$\vec{X}_i^{t+1} = \vec{X}_i^t + l_i \vec{D}_i^t(\vec{\alpha}_i^{t+1}) \quad (9)$$

where

$$l_i = a r_1 l_{max} \quad (10)$$

When a member escapes from the search space bounds, it will turn back to its previous position inside the search space [17]. GSO algorithm is presented in Algorithm 1.

Algorithm 1 Group Search Optimizer

$t \leftarrow 0$
Initialize randomly position $\vec{X}_i^{(0)} \in G$.
Calculate the fitness function for each member $\vec{X}_i^{(0)}$.
while (termination conditions are not met) **do**
  Pick the best group member ($\vec{X}_p^t$) to execute producing.
  Choose a percentage from the members (but the $\vec{X}_p^t$) to perform scrounging (eq. (8)).
  Ranging: The remaining members will perform ranging through random walks (eq. (9)).
  Calculate the new fitness value for each group member.
  $t := t + 1$
end while
Return $\vec{X}_p^t$

GSO scrounging operator focuses the search performed by the group in the most promising areas from the problem space, while producing and ranging operators are the main mechanisms employed by GSO for escaping from local minima.

III. Proposed Approach

In this section, we present a GSO-based partitional clustering algorithm for real-valued data. We also apply three different fitness functions to the proposed method, generating GSO$_J$, GSO$_{J_e}$ and GSO$_{J_{e2}}$ variants.

Consider a partition P of a data set with N patterns (each pattern represented by a vector $\vec{p}_j \in \Re^m$, where $j = 1, 2, ..., N$) in C clusters. Each cluster is represented by its centroid vector $\vec{g}_c \in \Re^m$ (where $c = 1, 2, ..., C$). Each group member $\vec{X}_i = \{\vec{g}_{i1}, ..., \vec{g}_{iC}\} \in \Re^n$, where $n = m$xC, represents C cluster centroids, one for each cluster [18].

Each member is initialized by the random choice of C patterns from the data set as their initial cluster centers.

The three fitness function adopted in this work are the Within-Cluster Sum of Squares (J) (eq. (11)), the Quantization Error (J_e) (eq. (12)) [19], [20] and the Weighted Quantization Error (J_{e2}) (eq. (13)) [21].

$$J(P_i) = \sum_{c=1}^{C} \sum_{\forall \vec{p}_j \in c} d(\vec{p}_j, \vec{g}_{ic}) \tag{11}$$

$$J_e(P_i) = \frac{\sum_{c=1}^{C} \sum_{\forall \vec{p}_j \in c} d(\vec{p}_j, \vec{g}_{ic})/|N_{ic}|}{C} \tag{12}$$

$$J_{e2}(P_i) = \sum_{c=1}^{C} [(\sum_{\forall \vec{p}_j \in c} d(\vec{p}_j, \vec{g}_{ic})/|N_{ic}|) \times (|N_{ic}|/N)] \tag{13}$$

where $|N_{ic}|$ is the number of patterns in cluster c, and

$$d(\vec{p}_j, \vec{g}_{ic}) = \sqrt{\sum_{k=1}^{m} (p_{jk} - g_{ick})^2} \tag{14}$$

is the Euclidean distance.

GSO algorithm for partitional clustering is presented in Algorithm 2.

Algorithm 2 GSO for Partitional Clustering

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

IV. Experimental Results

In this section, we evaluate our proposed GSO-*clustering* with three alternative fitness functions using twelve well-known real benchmark data sets from UCI Machine Learning Repository [22]. The selected real data sets are: Diabetes, E. Coli, Glass, Heart, Image Segmentation, Ionosphere, Iris, Landsat Satellite (Statlog), Optical Recognition, Page Blocks Classification, Wine and Yeast. These data sets present different degrees of difficulties, exploring aspects as unbalanced classes, overlapping among classes, different number of characteristics, different number of classes, and so on. All UCI Machine Learning data sets are described in Table I.

TABLE I
Benchmark Data sets Description.

Data set	Attributes	Classes	Patterns (N)
Diabetes	8	2	768
E. coli	7	8	336
Glass	9	6	214
Heart	13	2	270
Image Segmentation	18	7	2310
Ionosphere	33	2	351
Iris	4	3	150
Landsat Satellite	36	7	6435
Optical Recognition	64	10	5620
Page Blocks Classification	10	5	5473
Wine	13	3	178
Yeast	8	10	1484

We compare our three partitional GSO variants with five partitional clustering methods from literature: standard K-Means, standard Particle Swarm Optimizer (PSO) [12] using J as its fitness function, PSO-*clustering* [18], PSO variant presented in [20], which adopts J_e as its fitness function, PSO variant presented in [21], which adopts J_{e2} as its fitness function. Parameters for all tested algorithms are presented Table II. GSO-based methods used a population of 18 members, while PSO-based methods used a population of 20 particles, given that GSO *producing* operator executes two more evaluations for the fitness function. For all algorithms, the adopted number of clusters C is equal to the number of classes in each tested data set.

For comparison purposes, three clustering measures are employed: the quantization error (eq. (12)), the intra-cluster distance (eq. (15)) and the inter-cluster separation (eq. (16)) [23].

$$D_{max}(\vec{X}_i) = \max_{c=1,...,C} \{ \sum_{\forall \vec{p}_j \in c} d(\vec{p}_j, \vec{g}_{ic})/|N_{ic}| \} \tag{15}$$

$$D_{min}(\vec{X}_i) = \min_{\forall c_1, c_2, c_1 \neq c_2} \{d(\vec{g}_{ic_1}, \vec{g}_{ic_2})\} \tag{16}$$

All algorithms run in a MATLAB 7.6 environment. Fifty independent tests were executed for each data set, and all

TABLE II
FIXED PARAMETERS FOR ALL ALGORITHMS.

Algorithm	Parameter	Value
K-Means	*maxKMeansIt*	4000
All SI methods	t_{max}	200
PSO-based methods	S	20
PSO-*clustering*	c_1	1.5
	c_2	1.5
	w	0.75
	k_0	50
	J_0	0.1
Other PSO-based methods	c_1	2.0
	c_2	2.0
	w	0.9 to 0.4
GSO-based methods	S	18
	θ_{max}	π/a^2
	α_0	$\pi/4$
	β_{max}	$\theta_{max}/2$
	Scroungers Percentage	80%

evolutionary methods started with the same initial population in each test, obtained by a random process, as explained in GSO for partitional clustering algorithm (see Algorithm 2).

The evaluation criterion includes a rank system employed through the application of Friedman test [24], [25] 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 [26] 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

$$CD = q_a \sqrt{\frac{n_{alg}(n_{alg}+1)}{6n_{data}}} \tag{17}$$

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 $\sqrt{2}$ [27]. Since we evaluated eight algorithms through twelve data sets, using a significance value $\alpha = 0.05$ for Friedman test, we have $q_a = 3.031$. Given that J_e and D_{max} values are minimization metrics, the best methods will obtain lower ranks for the Friedman test, while for D_{min} (a maximization metric), the best methods will keep higher average ranks for the Friedman test.

Experimental results are presented in Table III and Table IV. The best results for each metric in each data set are bold faced. From the experimental results, based on an empirical analysis, we can observe that GSO$_{J_e}$ variant was able to obtain better performances than other GSO variants in nine out of twelve data sets. GSO$_{J_e}$ also outperformed all PSO variants in all tests concerning J_e and D_{min} metrics. In relation to K-Means, GSO$_{J_e}$ obtained the best results for J_e and D_{min} in ten out of twelve data sets, but in relation to D_{max}, K-Means obtained better results than GSO$_{J_e}$ in five out of twelve data sets.

TABLE III
EXPERIMENTAL RESULTS. AVERAGE VALUES FOR EACH METRIC ± STANDARD DEVIATION.

Data set	Algorithm	J_e	D_{max}	D_{min}
Diabetes	K-Means	1.04x10^4 ± 0.0	1.67x10^4 ± 0.0	5.0x10^4 ± 0.0
	PSO-*clust.*	1.03x10^4 ± 422.9	1.67x10^4 ± 582.4	3.97x10^4 ± 9.1x10^3
	PSO$_J$	1.04x10^4 ± 627.5	1.68x10^4 ± 788.4	4.0x10^4 ± 1.0x10^4
	PSO$_{J_e}$	8.05x10^3 ± 662.3	1.58x10^4 ± 423.4	7.93x10^4 ± 5.5x10^4
	PSO$_{J_{e2}}$	1.04x10^4 ± 627.5	1.68x10^4 ± 788.4	4.00x10^4 ± 1.1x10^4
	GSO$_J$	1.00x10^4 ± 384.2	1.66x10^4 ± 542.0	4.04x10^4 ± 5.6x10^3
	GSO$_{J_e}$	**7.58x10^3 ± 13.00**	**1.52x10^4 ± 26.01**	**1.02x10^6 ± 1.6x10^6**
	GSO$_{J_{e2}}$	1.00x10^4 ± 378.0	1.65x10^4 ± 541.1	4.05x10^4 ± 5.5x10^3
E. Coli	K-Means	0.050 ± 0.003	**0.103 ± 0.015**	0.044 ± 0.007
	PSO-*clust.*	0.057 ± 0.006	0.123 ± 0.023	0.047 ± 0.015
	PSO$_J$	0.057 ± 0.006	0.122 ± 0.025	0.050 ± 0.015
	PSO$_{J_e}$	0.045 ± 0.006	**0.100 ± 0.011**	0.039 ± 0.018
	PSO$_{J_{e2}}$	0.057 ± 0.006	0.122 ± 0.025	0.050 ± 0.015
	GSO$_J$	0.058 ± 0.006	0.129 ± 0.034	0.041 ± 0.010
	GSO$_{J_e}$	**0.033 ± 0.005**	**0.100 ± 0.011**	**0.072 ± 0.041**
	GSO$_{J_{e2}}$	0.058 ± 0.006	0.129 ± 0.034	0.041 ± 0.010
Glass	K-Means	3.747 ± 0.400	9.84 ± 1.23	1.95 ± 1.21
	PSO-*clust.*	4.520 ± 1.429	11.65 ± 5.90	0.761 ± 0.673
	PSO$_J$	4.520 ± 1.429	11.65 ± 5.90	0.761 ± 0.673
	PSO$_{J_e}$	3.266 ± 0.298	10.50 ± 2.15	0.477 ± 0.662
	PSO$_{J_{e2}}$	4.520 ± 1.429	11.65 ± 5.90	0.761 ± 0.673
	GSO$_J$	5.279 ± 1.086	15.43 ± 4.824	1.992 ± 1.203
	GSO$_{J_e}$	**1.058 ± 0.023**	**6.344 ± 0.125**	**59.72 ± 52.68**
	GSO$_{J_{e2}}$	5.279 ± 1.086	15.43 ± 4.824	1.992 ± 1.203
Heart	K-Means	2.16x10^3 ± 5.27	**2.72x10^3 ± 2.08**	6.64x10^3 ± 36.2
	PSO-*clust.*	2.63x10^3 ± 236.7	3.24x10^3 ± 337.5	6.40x10^3 ± 3.5x10^3
	PSO$_J$	2.63x10^3 ± 236.7	3.24x10^3 ± 337.5	6.40x10^3 ± 3.5x10^3
	PSO$_{J_e}$	2.53x10^3 ± 165.2	3.15x10^3 ± 207.6	5.52x10^3 ± 2.4x10^3
	PSO$_{J_{e2}}$	2.63x10^3 ± 236.7	3.24x10^3 ± 337.5	6.40x10^3 ± 3.5x10^3
	GSO$_J$	2.25x10^3 ± 127.5	2.84x10^3 ± 149.5	6.49x10^3 ± 1.9x10^3
	GSO$_{J_e}$	**1.87x10^3 ± 133.3**	3.54x10^3 ± 309.5	**4.46x10^5 ± 1.2x10^6**
	GSO$_{J_{e2}}$	2.26x10^3 ± 141.5	2.85x10^3 ± 175.9	6.52x10^3 ± 2.2x10^3
Image Seg.	K-Means	3.46x10^4 ± 1.2x10^4	2.13x10^5 ± 8.0x10^4	8.48x10^3 ± 1.3x10^3
	PSO-*clust.*	3.06x10^4 ± 2.3x10^4	1.68x10^5 ± 1.6x10^5	3.93x10^3 ± 2.9x10^3
	PSO$_J$	3.06x10^4 ± 2.3x10^4	1.68x10^5 ± 1.6x10^5	3.93x10^3 ± 2.9x10^3
	PSO$_{J_e}$	9.51x10^3 ± 587.9	2.46x10^4 ± 5.6x10^3	2.80x10^3 ± 3.4x10^3
	PSO$_{J_{e2}}$	3.06x10^4 ± 2.3x10^4	1.68x10^5 ± 1.6x10^5	3.93x10^3 ± 2.9x10^3
	GSO$_J$	6.35x10^4 ± 2.0x10^4	3.97x10^5 ± 1.4x10^5	7.16x10^3 ± 3.2x10^3
	GSO$_{J_e}$	**3.39x10^3 ± 181.7**	**2.36x10^4 ± 1.6x10^3**	**7.23x10^4 ± 6.3x10^4**
	GSO$_{J_{e2}}$	6.35x10^4 ± 2.0x10^4	3.97x10^5 ± 1.4x10^5	7.16x10^3 ± 3.2x10^3
Ionosp.	K-Means	7.103 ± 0.377	10.47 ± 0.329	320.5 ± 2.2x10^3
	PSO-*clust.*	7.827 ± 0.443	11.07 ± 0.714	11.58 ± 2.395
	PSO$_J$	7.847 ± 0.446	11.07 ± 0.723	11.78 ± 2.56
	PSO$_{J_e}$	6.439 ± 1.182	10.93 ± 1.379	16.68 ± 10.46
	PSO$_{J_{e2}}$	7.847 ± 0.446	11.07 ± 0.723	11.07 ± 0.723
	GSO$_J$	7.617 ± 0.329	10.55 ± 0.381	7.723 ± 1.106
	GSO$_{J_e}$	**4.656 ± 0.047**	**9.294 ± 0.058**	**1.47x10^3 ± 4.6x10^3**
	GSO$_{J_{e2}}$	7.617 ± 0.329	10.55 ± 0.381	7.724 ± 1.106

Table V shows the average Friedman/Nemenyi ranks obtained for each method in an overall evaluation. Fig. 1, Fig. 2 and Fig. 3 present all algorithms ordered by their average ranks for each metric, from the best method in the left to the worst method at right. The Friedman/Nemenyi test shows that GSO-based meta-heuristics were able to obtain better results than PSO-based methods when adopting the same fitness function, showing GSO potential as an optimization meta-heuristic. The overall evaluation also showed that the quantization error is a good option as a fitness function for GSO when dealing with clustering task.

The Friedman/Nemenyi tests also showed that there is no significant difference between J and J_{e2} when adopted as

TABLE IV
EXPERIMENTAL RESULTS (CONT.).

Data set	Algorithm	J_e	D_{max}	D_{min}
Iris	K-Means	0.550 ± 0.063	0.733 ± 0.215	3.45 ± 2.59
	PSO-*clust.*	0.741 ± 0.083	1.011 ± 0.154	3.721 ± 1.98
	PSO_J	0.741 ± 0.083	1.011 ± 0.154	3.721 ± 1.98
	PSO_{J_e}	0.714 ± 0.066	1.122 ± 0.240	2.770 ± 1.89
	$PSO_{J_{e2}}$	0.741 ± 0.083	1.011 ± 0.154	3.721 ± 1.982
	GSO_J	0.543 ± 0.019	**0.702 ± 0.066**	3.049 ± 0.190
	GSO_{J_e}	0.557 ± 0.027	1.041 ± 0.308	**4.669 ± 3.639**
	$GSO_{J_{e2}}$	**0.543 ± 0.019**	0.703 ± 0.068	3.054 ± 0.186
Landsat	K-Means	**$2.72x10^3$ ± 258.2**	**$5.41x10^3$ ± $1.3x10^3$**	**$4.00x10^3$ ± 429.9**
Sat.	PSO-*clust.*	$3.86x10^3$ ± 491.9	$7.55x10^3$ ± $1.6x10^3$	$2.79x10^3$ ± $1.4x10^3$
	PSO_J	$3.94x10^3$ ± 484.3	$7.49x10^3$ ± $1.5x10^3$	$3.10x10^3$ ± $1.6x10^3$
	PSO_{J_e}	$3.50x10^3$ ± 257.4	$6.57x10^3$ ± $1.3x10^3$	$2.31x10^3$ ± $1.1x10^3$
	$PSO_{J_{e2}}$	$3.94x10^3$ ± 484.3	$7.49x10^3$ ± $1.5x10^3$	$3.10x10^3$ ± $1.6x10^3$
	GSO_J	$3.90x10^3$ ± 491.2	$7.415x10^3$ ± $1.4x10^3$	$2.24x10^3$ ± 491.2
	GSO_{J_e}	$3.11x10^3$ ± 360.2	$7.05x10^3$ ± $1.8x10^3$	$3.35x10^3$ ± $1.4x10^3$
	$GSO_{J_{e2}}$	$3.90x10^3$ ± 491.2	$7.41x10^3$ ± $1.4x10^3$	$2.24x10^3$ ± 998.6
Optical	K-Means	652.3 ± 6.430	**779.5 ± 34.12**	480.1 ± 128.0
Rec.	PSO-*clust.*	$1.13x10^3$ ± 42.36	$1.40x10^3$ ± 82.66	783.8 ± 320.9
	PSO_J	$1.13x10^3$ ± 41.26	$1.40x10^3$ ± 80.1	767.2 ± 326.5
	PSO_{J_e}	$1.12x10^3$ ± 30.40	$1.41x10^3$ ± 94.52	720.7 ± 257.9
	$PSO_{J_{e2}}$	$1.13x10^3$ ± 41.26	$1.40x10^3$ ± 80.14	767.2 ± 326.5
	GSO_J	$1.11x10^3$ ± 64.74	$1.39x10^3$ ± 98.16	792.7 ± 328.4
	GSO_{J_e}	**193.1 ± 63.34**	$1.58x10^3$ ± 123.1	**$3.10x10^3$ ± $6.0x10^3$**
	$GSO_{J_{e2}}$	$1.11x10^3$ ± 64.74	$1.39x10^3$ ± 98.16	792.7 ± 328.4
Page	K-Means	$2.62x10^8$ ± 0.0	$1.18x10^9$ ± 0.0	$6.71x10^6$ ± 0.0
Blocks	PSO-*clust.*	$3.54x10^8$ ± $9.6x10^7$	$1.51x10^9$ ± $4.7x10^8$	$2.05x10^6$ ± $4.5x10^6$
Class.	PSO_J	$3.61x10^8$ ± $8.7x10^7$	$1.52x10^9$ ± $4.7x10^8$	$2.49x10^6$ ± $5.1x10^6$
	PSO_{J_e}	$6.11x10^6$ ± $1.5x10^5$	$3.06x10^7$ ± $7.7x10^5$	$4.59x10^5$ ± $1.2x10^6$
	$PSO_{J_{e2}}$	$3.61x10^8$ ± $8.7x10^7$	$1.52x10^9$ ± $4.7x10^8$	$2.49x10^6$ ± $5.1x10^6$
	GSO_J	$3.17x10^8$ ± $7.4x10^7$	$1.34x10^9$ ± $1.9x10^8$	$3.89x10^7$ ± $1.1x10^8$
	GSO_{J_e}	**$5.77x10^6$ ± $4.7x10^4$**	**$2.89x10^7$ ± $2.3x10^5$**	**$3.8x10^{10}$ ± $2.5x10^{11}$**
	$GSO_{J_{e2}}$	$3.17x10^8$ ± $7.4x10^7$	$1.34x10^9$ ± $1.9x10^8$	$3.89x10^7$ ± $1.1x10^8$
Wine	K-Means	**$1.50x10^4$ ± 442.6**	**$2.76x10^4$ ± $3.3x10^3$**	$8.31x10^4$ ± $2.5x10^4$
	PSO-*clust.*	$1.83x10^4$ ± $2.5x10^3$	$3.08x10^4$ ± $7.5x10^3$	$7.30x10^4$ ± $4.4x10^4$
	PSO_J	$1.83x10^4$ ± $2.5x10^3$	$3.08x10^4$ ± $7.5x10^3$	$7.30x10^4$ ± $4.4x10^4$
	PSO_{J_e}	$1.81x10^4$ ± $2.1x10^3$	$3.02x10^4$ ± $9.0x10^3$	$7.19x10^4$ ± $4.0x10^4$
	$PSO_{J_{e2}}$	$1.83x10^4$ ± $2.5x10^3$	$3.08x10^4$ ± $7.5x10^3$	$7.30x10^4$ ± $4.4x10^4$
	GSO_J	**$1.50x10^4$ ± 237.9**	$2.89x10^4$ ± $1.2x10^3$	$7.44x10^4$ ± $9.0x10^3$
	GSO_{J_e}	**$1.52x10^4$ ± 847.9**	**$2.75x10^4$ ± $4.4x10^3$**	**$9.78x10^4$ ± $7.0x10^4$**
	$GSO_{J_{e2}}$	**$1.50x10^4$ ± 223.7**	$2.89x10^4$ ± $1.2x10^3$	$7.45x10^8$ ± $8.9x10^3$
Yeast	K-Means	0.038 ± 0.002	**0.065 ± 0.015**	**0.028 ± 0.003**
	PSO-*clust.*	0.058 ± 0.009	0.124 ± 0.038	0.023 ± 0.012
	PSO_J	0.061 ± 0.012	0.133 ± 0.050	0.023 ± 0.011
	PSO_{J_e}	0.048 ± 0.002	0.076 ± 0.009	0.014 ± 0.007
	$PSO_{J_{e2}}$	0.061 ± 0.012	0.133 ± 0.050	0.023 ± 0.011
	GSO_J	0.049 ± 0.006	0.122 ± 0.040	0.023 ± 0.007
	GSO_{J_e}	**0.026 ± 0.005**	0.069 ± 0.012	**0.029 ± 0.018**
	$GSO_{J_{e2}}$	0.049 ± 0.006	0.122 ± 0.040	0.023 ± 0.007

fitness functions and applied to both PSO and GSO, for the evaluated data sets. GSO_{J_e} obtained the larger inter-cluster separation and the second best more compact clusters from all tested algorithms.

V. CONCLUSION

In this work, a GSO partitional clustering method for real-valued data was presented. We also evaluated the influence of three alternative fitness functions (the Within-Cluster Sum of Squares, the Quantization Error and the Weighted Quantization Error) on the behavior of GSO, generating GSO_J, GSO_{J_e} and $GSO_{J_{e2}}$ variants.

Experiments were performed on twelve real-valued classification data sets obtained from UCI Machine Learning

TABLE V
FRIEDMAN/NEMENYI AVERAGE RANKS FOR EACH METRIC.

Algorithm	J_e	D_{max}	D_{min}
K-Means	124.97	**126.20**	228.16
PSO-*clustering*	268.32	239.26	176.98
PSO_J	273.28	240.61	181.24
PSO_{J_e}	170.23	171.87	145.14
$PSO_{J_{e2}}$	273.28	240.61	181.24
GSO_J	221.28	221.26	190.19
GSO_{J_e}	**50.98**	143.14	**309.20**
$GSO_{J_{e2}}$	221.65	221.05	191.87

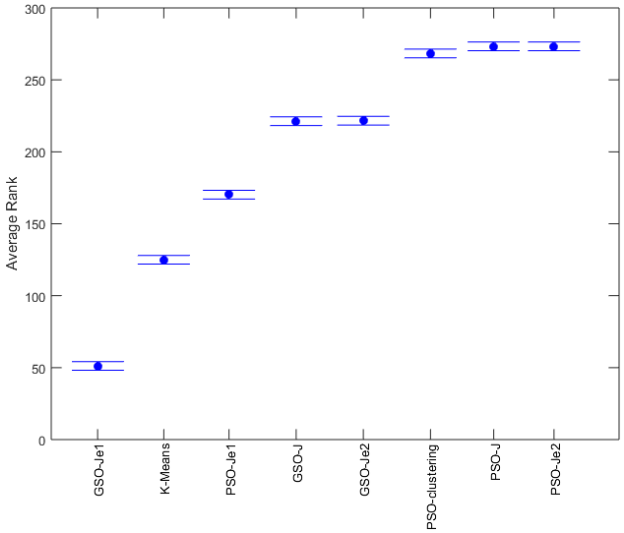

Fig. 1. Friedman test in relation to J_e (from the best method, in the left, to the worst, at right).

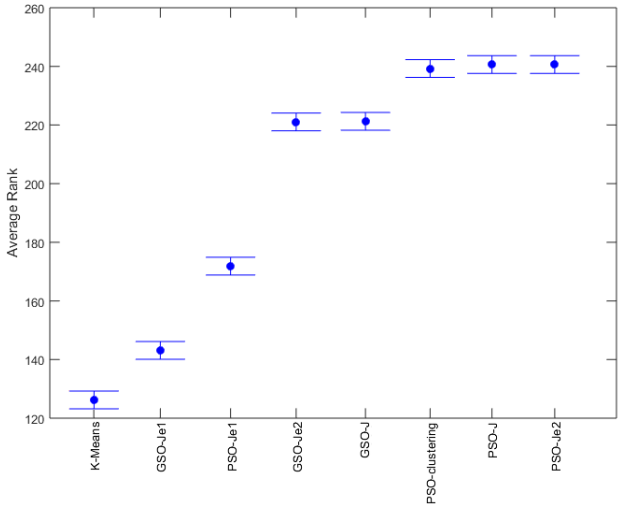

Fig. 2. Friedman test in relation to D_{max}.

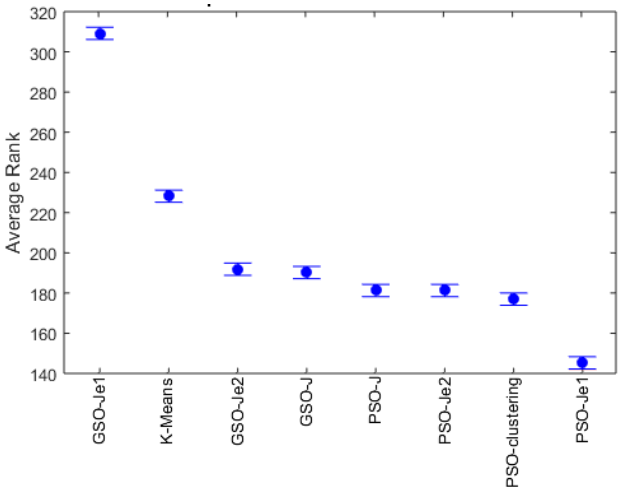

Fig. 3. Friedman test in relation to D_{min}.

Repository. GSO variants were compared to K-Means, PSO-*clustering*, and PSO$_J$, PSO$_{J_e}$ and PSO$_{J_{e2}}$. Experimental results showed that GSO variants were able to outperform their corresponding PSO variants. Also, GSO$_{J_e}$ was able to obtain the best average results in relation to Quantization Error and Inter-Cluster Separation.

An overall evaluation was obtained by the application of Friedman/Nemenyi tests. The overall evaluation showed the potential of GSO$_{J_e}$ in relation to other GSO variants and all PSO variants studied. GSO$_{J_e}$ was able to obtain the best separation among clusters and the second best degree of compactness for the final clusters.

As future works, we intend to evaluate the influence of hybrid fitness function on the behavior of GSO as a manner to improve the degree of compactness of the final clusters. We also intend to extend our partitional GSO clustering method to the context of automatic clustering [28], so the algorithm would be able to determine the best number of clusters automatically (as part of its generational process).

ACKNOWLEDGMENT

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

REFERENCES

[1] M. C. Naldi and R. J. G. B. Campello, "Evolutionary k-means for distributed data sets," *Neurocomputing*, vol. 127, pp. 30–42, 2014.

[2] E. R. Hruschka, R. J. G. B. Campello, A. A. Freitas, and A. P. L. F. De Carvalho, "A survey of evolutionary algorithms for clustering," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 39, no. 2, pp. 133–155, 2009.

[3] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[4] R. Xu, D. Wunsch *et al.*, "Survey of clustering algorithms," *Neural Networks, IEEE Transactions on*, vol. 16, no. 3, pp. 645–678, 2005.

[5] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1. California, USA, 1967, pp. 281–297.

[6] M. C. Naldi, R. J. Campello, E. R. Hruschka, and A. Carvalho, "Efficiency issues of evolutionary k-means," *Applied Soft Computing*, vol. 11, no. 2, pp. 1938–1952, 2011.

[7] J. Yao, N. Kharma, and Y. Q. Zhu, "On clustering in evolutionary computation," in *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*. IEEE, 2006, pp. 1752–1759.

[8] A. E. Eiben and J. E. Smith, *Introduction to evolutionary computing*. Springer Berlin, 2010, vol. 2.

[9] R. Storn and K. Price, "Differential evolutiona simple and efficient adaptive scheme for global optimization over continuous spaces. international computer science institute, berkeley," CA, 1995, Tech. Rep. TR-95–012, Tech. Rep., 1995.

[10] ——, "Differential evolution–a simple , efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[11] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 26, no. 1, pp. 29–41, 1996.

[12] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4. IEEE, 1995, pp. 1942–1948.

[13] S. He, Q. H. Wu, and J. Saunders, "Group search optimizer: an optimization algorithm inspired by animal searching behavior," *Evolutionary Computation, IEEE Transactions on*, vol. 13, no. 5, pp. 973–990, 2009.

[14] C. Barnard and R. Sibly, "Producers and scroungers: a general model and its application to captive flocks of house sparrows," *Animal Behaviour*, vol. 29, no. 2, pp. 543–550, 1981.

[15] I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin, "Effective leadership and decision-making in animal groups on the move," *Nature*, vol. 433, no. 7025, pp. 513–516, 2005.

[16] C. L. Higgins and R. E. Strauss, "Discrimination and classification of foraging paths produced by search-tactic models," *Behavioral Ecology*, vol. 15, no. 2, pp. 248–254, 2004.

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

[18] C.-Y. Chen and F. Ye, "Particle swarm optimization algorithm and its application to clustering analysis," in *Networking, Sensing and Control, 2004 IEEE International Conference on*, vol. 2. IEEE, 2004, pp. 789–794.

[19] M. Omran, A. Salman, and A. P. Engelbrecht, "Image classification using particle swarm optimization," in *Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning*, vol. 1. Singapore, 2002, pp. 18–22.

[20] D. Van der Merwe and A. P. Engelbrecht, "Data clustering using particle swarm optimization," in *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, vol. 1. IEEE, 2003, pp. 215–220.

[21] A. A. A. Esmin, D. L. Pereira, and F. De Araujo, "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*. IEEE, 2008, pp. 1817–1822.

[22] A. Asuncion and D. Newman, "Uci machine learning repository," 2007.

[23] M. T. Wong, X. He, and W.-C. Yeh, "Image clustering using particle swarm optimization," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*. IEEE, 2011, pp. 262–268.

[24] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the american statistical association*, vol. 32, no. 200, pp. 675–701, 1937.

[25] ——, "A comparison of alternative tests of significance for the problem of m rankings," *The Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92, 1940.

[26] P. Nemenyi, "Distribution-free multiple comparisons," in *Biometrics*, vol. 18, no. 2. INTERNATIONAL BIOMETRIC SOC 1441 I ST, NW, SUITE 700, WASHINGTON, DC 20005-2210, 1962, p. 263.

[27] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[28] M. G. Omran, A. Salman, and A. P. Engelbrecht, "Dynamic clustering using particle swarm optimization with application in image segmentation," *Pattern Analysis and Applications*, vol. 8, no. 4, pp. 332–344, 2006.