An Evaluation on Competitive and Cooperative Evolutionary Algorithms for Data Clustering

Luciano D. S. Pacifico
Departamento de Computacao – DC
Universidade Federal Rural de Pernambuco – UFRPE
Recife, Pernambuco, Brazil
ldsp@cin.ufpe.br

Teresa B. Ludermir
Centro de Informatica – CIn
Universidade Federal de Pernambuco – UFPE
Recife, Pernambuco, Brazil
tbl@cin.ufpe.br

***Abstract*—Data clustering methods are important tools for exploratory data analysis in many real world applications, such as data mining, image understanding, text analysis, engineering, medicine, and so on. Partitional clustering models are the most popular clustering methods, but these approaches suffer from some limitations, like the sensibility to algorithm initialization and the lack of mechanisms to help them escaping from local minima points. Evolutionary Algorithms (EAs) are global optimization meta-heuristics known for their capabilities to find optimal solutions even when dealing with hard and complex problems. Although many EAs are based on competitive behavior among individuals, its is known that cooperation may lead to better solutions then sheer competition. In this work, we perform a comparative analysis among four state-of-the-art EAs (Genetic Algorithm, Differential Evolution, Particle Swarm Optimization and Group Search Optimization), implemented in both competitive and cooperative frameworks, in the context of data clustering problem. Experiments are executed using eleven real world benchmark datasets as the testing bed, so we could access whether competitive or cooperative behaviors would prevail. The experimental results showed that cooperative algorithms are able to find better solutions, in average, when dealing with clustering problems, than their corresponding competitive approaches, and such models also require less storage memory to keep their population in comparison to competitive methods.**

***Index Terms*—Evolutionary Computing, Data Clustering, Cooperative Algorithms, Cooperative Coevolutionary Algorithms**

I. Introduction

In the past few decades, the amount of daily produced data (in electronic devices, such as smartphones, tablets, notebook and desktop computers, cars, GPS, smart TVs, and so on) has increased exponentially, in such a way that automatic and scalable computational systems are even more required, so useful information would be extracted from such big data scenario. Nowadays, it is impossible to a real world system rely in human analysis only, once the need for precise and reliable information in a short period of time has become mandatory [1].

As one of the most fundamental exploratory data analysis tools, clustering algorithms consists in an attempt to categorize observations (*data patterns*) in groups (*clusters*) based only on their inner properties, in such a way that observations belonging in the same group present a higher degree of similarity than observations belonging in different groups, which must present a high degree of dissimilarity. Clustering techniques require no prior assumptions concerning the problem to be solved, representing the unsupervised learning category of algorithms in pattern recognition, finding applications in many challenging knowledge discovery in databases (KDD) tasks, in fields like medicine, social sciences, engineering, bioinformatics, and so on.

The most popular clustering approaches are the partitional clustering algorithms, such as K-Means [2] and K-Medoids [3]. Partitional clustering methods provide a partition of the dataset into a prefixed number of clusters. Each cluster is represented by its centroid vector, and the clustering process is driven in an effort to optimize a criterion function iteratively, and, in each step of the execution, all centroids are updated in an attempt to improve the quality of the final solution. Partitional methods are known for their sensibility to the centroid initialization process, since they only perform local searches on the problem search space, what may lead to poor solutions, once the initial partition could have been placed in a region containing local minima points.

Natural-inspired global search meta-heuristics, such as Evolutionary Algorithms (EAs) and Swarm Intelligence (SIs) methods, that are extensions of EAs, have been increasingly applied to solve a great variety of difficult problems, including data clustering [4], [5], which is, from an optimization perspective, considered as a particular kind of NP-hard grouping problem [6]. In EAs, a population of candidate solutions to the problem at hand is kept and evolved, according to a generational process to optimize a criterion function (known as the *fitness function*). In EAs such as Genetic Algorithm (GA) [7] and Differential Evolution (DE) [8] (which are genetic-based approaches), the searching schema is driven by operators that simulate biological processes like mutation, recombination and selection. The searching process in SIs is based in an attempt to simulate self-organizing collective behavior of social animals, like swarming, flocking and herding [9]. Examples of SI algorithms are the Ant Colony Optimization (ACO) [10], Particle Swarm Optimization (PSO) [11] and Group Search Optimization (GSO) [12].

Although the main framework driving evolutionary algorithms is based on sheer competition among population indi-

This work was supported by CNPq, CAPES and FACEPE

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

viduals, sometimes cooperation may lead to better situations than competitive behavior. In this work, we extend the cooperative coevolutionary (CC) [13] framework for evolutionary algorithms (CCEAs) to the context of partitional clustering algorithms. In CCEAs, the population and the problem search space are split into k groups using a divide-and-conquer approach, in such a way that each sub-population is responsible for the optimization of only a reduced set of the global problem features each time. This work aims to evaluate the effectivity and advantages (if any) of cooperative coevolutionary methods in comparison to competitive approaches when dealing with real world clustering problems.

This work is organized as follows. Section II offers a brief introduction on Cooperative Evolutionary Algorithms (Section II-A), Evolutionary Algorithms and Cooperative Coevolutionary Evolutionary Algorithms in the context of partitional data clustering (Section II-B and Section II-C, respectively). The experimental analysis is presented in Section III, and, after that, some conclusions and leads for future works are presented (Section IV).

II. Background

A. Cooperative Evolutionary Algorithms

Cooperation involves a collection (population) of agents (individuals) that interact by communicating information to each other while solving problems [14]. In the context of Evolutionary Algorithms, there are two main ways to implement cooperation among individuals [15]: the Evolution Islands (EI) framework and the Cooperative Coevolutionary (CC) framework.

In EI, each population is split into sub-populations (i.e., *evolution islands*) that are geographically isolated from each other. Each evolution island will execute its search as an independent population, executing all steps for the corresponding EA (like it would do for the whole population). After a prefixed number of iterations, the islands will send and receive some individuals from other sub-populations, promoting an information exchange among islands. The way islands interact is determined by their topology of communication (such as *star*, *ring*, *Von Neumann*, *cellular*, etc) [16], [17].

In CC framework, the population is also divided into sub-populations, but instead of having each sub-population attempting to solve the global problem as a whole (just like in IE), the problem search space is also divided among the sub-populations. Thus, the original n-dimensional search space is split into $1 \leq k \leq n$ partitions of size d, with $k \times d = n$. Although the n-dimensional search space has been divided into k d-dimensional partitions, in which local searches are executed, the global problem remains n-dimensional. The k sub-populations of d-dimensional individuals need to cooperate, offering their best set of features found so far, so each individual from the other sub-populations would be able to complete the information necessary to evaluate their fitness.

The original CC framework was introduced by Potter and De Jong [13] by means of a cooperative coevolutionary Genetic Algorithm (CCGA). Potter and De Jong suggested that the search space should be partitioned by splitting the solution vectors into smaller ones, and each of these smaller vectors would then be searched by a separate GA (*different species*). CCGA method achieved significant improvement in performance over the basic GA when dealing with continuous optimization problems [13]. CC model has also been extended to the context of other evolutionary and swarm intelligence methods, such as Evolutionary Programming [18], Differential Evolution [19], [20], Particle Swarm Optimization [21], [22] and Group Search Optimization [23], [24].

An interesting survey on Cooperative Coevolutionary approaches can be found in [25].

B. Evolutionary Algorithms for Partitional Data Clustering

This section explains the most commonly adopted representation schema for evolutionary algorithms when such techniques are used as partitional clustering methods [26].

Firstly, consider a partition P_C of a dataset with N_O patterns (each pattern represented by a vector $\mathbf{o}_j \in \Re^m$, where $j = 1, 2, ..., N_O$) in C clusters (where C is a given input parameter for the partitional algorithm). Each cluster is represented by its centroid vector $\mathbf{g}_c \in \Re^m$ (where $c = 1, 2, ..., C$). 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 [26]. For example, if $m = 4$ and $C = 3$, each individual will be a vector $\mathbf{X}_i \in \Re^{12}$, where the first four features will represent the centroid $\mathbf{g}_1$, features 5-th to 8-th will represent the centroid $\mathbf{g}_2$, and the last four features will represent the centroid $\mathbf{g}_3$. In Fig. 1, $\mathbf{X}_i = \{1.2, 4.6, 0.3, 5.9, 2.1, 5.2, 0.8, 2.0, 0.3, 6.1, 2.2, 0.9\}$, and it codifies centroids $\mathbf{g}_1 \in \Re^4$, $\mathbf{g}_2 \in \Re^4$ and $\mathbf{g}_3 \in \Re^4$, such that $\mathbf{g}_1 = \{1.2, 4.6, 0.3, 5.9\}$, $\mathbf{g}_2 = \{2.1, 5.2, 0.8, 2.0\}$ and $\mathbf{g}_3 = \{0.3, 6.1, 2.2, 0.9\}$.

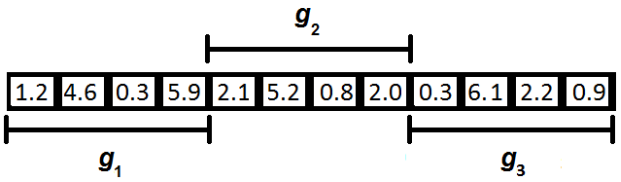

Fig. 1. Individual representation ($\mathbf{g}_1$, $\mathbf{g}_2$ and $\mathbf{g}_3$ represent cluster centroids).

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 dataset in analysis to compose the initial cluster centroids, 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. (1)) or some alternative function that takes such criterion as its main component, just like in [4], [24], [26]–[29].

$$J(P_C) = \sum_{c=1}^{C} \sum_{\forall i \in c} d(\mathbf{o}_i, \mathbf{g}_c) \quad (1)$$

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 evolve 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 furnished as the clustering solution. A generic evolutionary algorithm for partitional data clustering is presented in Algorithm 1.

Algorithm 1 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 dataset as its initial cluster centroids.
Generate the initial partition $\mathbf{X}_i^{(0)}.P_C^{(0)}$, assigning each pattern $\mathbf{o}_i$ 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 $\mathbf{o}_i$ 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 $\mathbf{X}_{best}^{t_{max}}$.

C. Cooperative Coevolutionary Algorithms for Partitional Data Clustering

Although the most popular way to codify the population for EAs when dealing with partitional data clustering is by representing each individual as a set of cluster centroids (see Section II-B), such representation may increase the computational cost in terms of storage space significantly, if the number of intended clusters is to high, or if the data patterns are composed of a large set of features. To reduce the space complexity of partitional EAs, the cooperative coevolutionary framework can be easily adapted to the context of partitional data clustering. Instead of having a single population G, where each individual represents a set of C cluster centroids in parallel, we have C sub-populations composed by individuals which represent only one cluster centroid each time [22], [24]. That way, each sub-population will perform local searches in an attempt to optimize just one cluster centroid as well.

Formally, consider a partition P_C of a dataset with N_O patterns $\mathbf{o}_j \in \Re^m$ in C clusters. Each cluster is represented by its centroid vector $\mathbf{g}_c \in \Re^m$. The population G of S individuals is divided into C sub-populations. The i-th individual of the k-th ($1 \leq k \leq C$) sub-population $G_k.\mathbf{X}_i \in \Re^m$ represents only one cluster centroid.

Although the search space has been divided in C m-dimensional sub-regions, the original problem remains ($m \times C$)-dimensional. The cooperation is employed in such a way that each sub-population will contribute with the best set of features they have found so far (i.e., the best cluster centroid they represent), so the individuals of other sub-populations would be able to evaluate their fitness: the i-th individual of the k-th ($1 \leq k \leq C$) sub-population will be concatenated to the best centroids found so far by the other sub-populations, originating its own partition $G_k.\mathbf{X}_i.P_C$ of the original dataset. The fitness value $f : \Re^{(m \times C)} \rightarrow \Re$ for $G_k.\mathbf{X}_i$ is computed as in eq. (2):

$$f(\mathbf{X}_i) = f([G_1.\mathbf{X}_{best_1}, ..., G_k.\mathbf{X}_i, ..., G_C.\mathbf{X}_{best_C}]) \quad (2)$$

The final solution furnished by the CCEA is obtained by the concatenation of the local best solutions found so far by each sub-population (eq.(3)):

$$\mathbf{X}_{best} = [G_1.\mathbf{X}_{best_1}, ..., G_k.\mathbf{X}_{best_k}, ..., G_C.\mathbf{X}_{best_C}] \quad (3)$$

A generic cooperative coevolutionary algorithm for partitional data clustering is presented in Algorithm 2.

Algorithm 2 Generic Partitional Cooperative Coevolutionary Algorithm

$t \leftarrow 0$.
Divide the population in C sub-populations.
for all sub-population $G_k^{(0)} (k = 1, \ldots, C)$ **do**
  Initialize each individual from $G_k^{(0)}$ by the random choice of one pattern from the original dataset as its initial cluster centroid.
end for
Generate the initial partition $G_k^{(0)}.\mathbf{X}_i^{(0)}.P_C^{(0)}$, assigning each pattern $\mathbf{o}_i$ to its closest cluster, for each individual $G_k^{(0)}.\mathbf{X}_i^{(0)}$.
Calculate the fitness value for each individual $G_k^{(0)}.\mathbf{X}_i^{(0)}$, according to eq. (2).
while (termination conditions are not met) **do**
  for all sub-population $G_k^t (k = 1, \ldots, C)$ **do**
    Execute all evolutionary operators, according to the selected evolutionary algorithm, on current sub-population G_k^t.
    Assign each pattern $\mathbf{o}_i$ to its closest cluster in $G_k^t.\mathbf{X}_i^t.P_C^t$, for each $\mathbf{X}_i^t \in G_k^t$.
    Calculate the new fitness value for each individual $\mathbf{X}_i^t \in G_k^t$, according to eq. (2).
  end for
  Determine the global best member $\mathbf{X}_{best}^t$ according to eq. (3).
  $t \leftarrow t + 1$.
end while
return $\mathbf{X}_{best}^t$.

III. Experimental Evaluation

In this section, we evaluate the behavior of both competitive and cooperative evolutionary algorithms when dealing with

TABLE I
BENCHMARK DATASETS DESCRIPTION.

Dataset	Attributes	Classes	Patterns
Blood Transfusion	4	2	748
Banknote Authentication	4	2	1372
Cancer	9	2	699
Diabetes	8	2	768
E. Coli	7	8	336
Glass	9	6	214
Heart	13	2	270
Ionosphere	34	2	351
Iris	4	3	150
Seeds	7	3	210
Wine	13	3	178

TABLE II
PARAMETERS FOR ALL ALGORITHMS.

Algorithm	Parameter	Value
All EAs	t_{max} S	200 $5 \times C$
GA and CCGA	C_r *and* M_r S_r	0.8 and 0.05 0.2
DE and CCDE	F and C_r	0.8 and 0.9
PSO and CCPSO	w c_1 and c_2	0.9 to 0.4 2.0 and 2.0
GSO and CCGSO	Scroungers % and θ_{max} α_0 and α_{max}	80% and π/a^2 $\pi/4$ and $\theta_{max}/2$

data clustering problem. Eleven well-known real world benchmark datasets from UCI Machine Learning Repository [30] are selected as the testing bed. The selected real datasets are presented in Table I. These datasets present different degrees of difficulties, exploring aspects as unbalanced classes, overlapping among classes, different number of features, different number of classes, and so on.

Two evolutionary algorithms with genetic inspiration (Genetic Algorithm and Differential Evolution) and two swarm intelligence algorithms (Particle Swarm Optimization and Group Search Optimization) are selected for comparison purposes. The selected approaches represent state-of-the-art models on evolutionary algorithms and data clustering literature, being successfully applied in many applications [24], [29], [31]–[36]. All selected algorithms have been implemented as both competitive and cooperative partitional clustering models, as described in Section II-B and Section II-C, respectively.

The parameters for each model (obtained from [12], [24], [37], [38]) are presented in Table II. The population size for all evolutionary algorithms is equal to five times the number of intended clusters for each dataset, as a manner to assure that CCEAs sub-populations (mainly, CCDE and CCGSO) will fit the algorithm restrictions (such as, minimum number of individuals [8], [39], at least one of each kind of individual [12], [24], etc.).

All algorithms have been implemented in a Python programming language. Thirty independent tests have been executed for each dataset, and all evolutionary methods started with the same initial random population in each test, as explained in Sections II-B and II-C. For all tests, the adopted number of clusters C is equal to the number of classes per dataset.

As comparison measures, four well-established clustering metrics from literature are employed: the Within-Cluster Sum of Squares (J, eq.(1)), the Quantization Error (J_e, eq. (4)), the Intra-Cluster Distance (D_{max}, eq. (5)), and the Inter-Cluster Separation (D_{min}, eq. (6)) [40], [41].

$$J_e(\mathbf{X}_i.P_C) = \frac{\sum_{c=1}^{C} \sum_{\forall \mathbf{o}_j \in c} d(\mathbf{o}_j, \mathbf{g}_{ic})/|N_{ic}|}{C} \tag{4}$$

$$D_{max}(\mathbf{X}_i.P_C) = \max_{c=1,\ldots,C} \{ \sum_{\forall \mathbf{o}_j \in c} d(\mathbf{o}_j, \mathbf{g}_{ic})/|N_{ic}| \} \tag{5}$$

$$D_{min}(\mathbf{X}_i.P_C) = \min_{\forall c_1, c_2, c_1 \neq c_2} \{ d(\mathbf{g}_{ic_1}, \mathbf{g}_{ic_2}) \} \tag{6}$$

where $|N_{ic}|$ is the cardinality of cluster $\mathbf{g}_c$ from individual $\mathbf{X}_i$. The within-cluster sum of squares gives an overall view on how close objects are in their clusters in a given partition P_C, while the quantization error gives us an average view on how objects are distant in relation to their cluster centroid. The intra-cluster distance shows the highest average degree of scattering in a cluster in P_C, and the inter-cluster separation shows how close the two closest clusters are. Once clustering models aim to find the best partition, where cluster objects in the same cluster are more similar to each other, and with the highest degree of separation among different clusters, its desirable that the final solution obtained by a clustering algorithm will present lower values for J, J_e and D_{max} and higher values in relation to D_{min}.

The evaluation criterion includes a rank system employed through the application of Friedman test [42], [43] 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 [44] 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

$$CD = q_a \sqrt{\frac{n_{alg}(n_{alg}+1)}{6 n_{data}}} \tag{7}$$

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 $\sqrt{2}$ [45]. Once our experiments are executed with $n_{data} = 11$ and $n_{alg} = 8$, we have a $CD = 3.1656$. Since J, J_e and D_{max} are minimization metrics, the best methods will obtain lower ranks for the Friedman/Nemenyi test, while for D_{min} (maximization metric), the best methods will find higher average ranks in the Friedman/Nemenyi test.

The experimental results are shown in Table III and Table IV.

The experimental results show that CCPSO and CCGSO are able to obtain better performances than their corresponding competitive models (PSO and GSO, respectively) in most of

TABLE III

EXPERIMENTAL RESULTS. FOR EACH CLUSTERING METRIC, *Mean* REPRESENTS THE AVERAGE VALUES OBTAINED IN THIRTY EXECUTIONS OF THE EXPERIMENTS, WHILE *Std* REPRESENTS THE STANDARD DEVIATION FOR THE THIRTY EXECUTIONS OF THE EXPERIMENTS.

Dataset	Algorithm	J		J_e		D_{max}		D_{min}	
		Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
Banknote Authentication	GA	45053.3467	820.1455	33.7543	0.4798	37.8211	1.9842	142.1931	19.1010
	CCGA	45284.2152	1096.0931	33.9297	0.7868	38.0610	2.3412	145.2388	17.9120
	DE	44052.5239	14.4696	33.0608	0.0072	35.9766	0.0103	149.3565	0.2159
	CCDE	44688.7940	1100.0716	33.5348	0.7556	37.2892	2.2269	145.1725	12.7306
	PSO	44049.4561	0.0257	33.0591	0.0	35.9779	0.0	149.3964	0.0471
	CCPSO	44049.4431	0.0004	33.0591	0.0	35.9779	0.0	149.3832	0.0056
	GSO	44060.2157	14.2291	33.0703	0.0180	35.9978	0.0706	149.1498	1.0336
	CCGSO	44052.3708	3.6081	33.0616	0.0040	35.9811	0.0100	149.2133	0.4496
Blood Transfusion	GA	686116142.1	12417025.2	2152217.8	98709.4	3923789.7	160775.4	9533964.7	857713.7
	CCGA	697062207.7	23405975.0	2137727.1	141926.9	3893675.4	220196.6	9248724.0	1622835.7
	DE	677749653.3	295.0	2108932.0	0.7748	3859382.4	1.3294	9268164.1	2975.5
	CCDE	686685421.4	36314502.5	2097425.0	74565.4	3832792.4	149633.4	9307484.2	398442.2
	PSO	677757987.6	20008.8	2108942.7	19.0838	3859392.5	19.1046	9267824.9	1100.9
	CCPSO	677755370.2	16431.6	2108938.8	14.3657	3859388.0	15.3665	9267537.6	47.4385
	GSO	677776016.8	25303.6	2108981.7	41.9005	3859453.1	68.6110	9261178.1	15352.6
	CCGSO	677822811.7	137698.8	2109054.7	152.2089	3859541.2	226.5976	9263743.2	11043.0
Cancer	GA	251.39556	4.22867	0.47533	0.00830	0.82229	0.01067	2.46162	0.17984
	CCGA	250.84247	2.90897	0.47445	0.00530	0.82516	0.01094	2.41455	0.18860
	DE	249.35570	14.69270	0.47326	0.03236	0.82597	0.06268	2.36323	0.14057
	CCDE	266.57653	25.67322	0.50960	0.05658	0.88883	0.10486	2.62479	0.45659
	PSO	254.06655	11.87852	0.47495	0.01575	0.82114	0.02087	2.55737	0.14402
	CCPSO	246.07225	2.47213	0.46491	0.00504	0.81067	0.00963	2.44748	0.05851
	GSO	245.20692	1.72360	0.46478	0.00267	0.80580	0.00450	2.25988	0.05812
	CCGSO	243.75571	0.35020	0.46254	0.00083	0.80578	0.00131	2.30797	0.02957
Diabetes	GA	5337576.5	169125.5	10372.6	543.86	16797.4	637.21	49910.0	8457.1
	CCGA	5346897.1	158120.6	10391.7	543.32	16780.5	626.41	49314.0	9992.1
	DE	5162293.3	71008.5	10305.4	175.78	16713.5	226.66	49427.9	1703.5
	CCDE	5319990.4	194186.1	10155.4	361.97	16500.0	492.67	48198.9	4798.1
	PSO	5155270.0	36696.0	10369.0	35.794	16787.7	72.028	49979.5	336.03
	CCPSO	5171653.6	98411.4	10319.5	173.63	16719.8	252.80	49905.5	450.99
	GSO	5149710.3	9466.6	10382.5	46.970	16808.7	51.406	49993.3	518.19
	CCGSO	5148361.0	5459.0	10366.2	13.543	16786.4	23.303	50001.3	323.14
E. Coli	GA	18.17744	0.80208	0.06955	0.00740	0.15747	0.03541	0.05761	0.02364
	CCGA	15.45975	0.40937	0.05666	0.00260	0.13250	0.01144	0.04753	0.01611
	DE	20.72997	0.76786	0.06627	0.00581	0.12051	0.02841	0.03337	0.02261
	CCDE	15.05779	0.82346	0.05526	0.00369	0.12767	0.02106	0.05195	0.00806
	PSO	15.14878	0.71929	0.05132	0.00350	0.11253	0.02440	0.04803	0.00884
	CCPSO	14.49937	0.60067	0.05058	0.00142	0.12305	0.01732	0.04946	0.00643
	GSO	15.15042	0.41485	0.05473	0.00296	0.12473	0.02007	0.04823	0.01123
	CCGSO	14.28233	0.35180	0.05121	0.00153	0.12634	0.01209	0.04697	0.00881
Glass	GA	421.24000	21.10890	3.71332	0.72592	8.05231	2.12889	3.99240	3.02430
	CCGA	379.40753	15.83310	3.34341	0.64554	7.82575	1.51687	3.19245	1.37435
	DE	538.75474	26.50355	4.82566	1.21121	11.89588	4.46998	1.52952	1.27193
	CCDE	360.88782	21.10689	3.71055	0.70515	9.19618	2.85249	3.24326	0.82053
	PSO	364.34764	20.67760	3.80791	0.58070	9.37955	1.87793	3.33399	0.52022
	CCPSO	351.93828	14.82960	3.49792	0.68658	8.71667	2.09562	3.23621	0.66857
	GSO	378.47152	17.51526	3.76134	0.48432	9.07024	1.75713	2.84770	1.25433
	CCGSO	353.48994	15.09140	3.28826	0.61295	7.98469	1.79633	2.85007	0.72967

the cases in terms of the final best fitness value (J). CCGA presented performances (in terms of the best fitness value) in most cases at least as good as GA algorithm. Also, CCGA showed an overall average performance (according to the Friedman/Nemenyi test - see Table V) significantly better than GA considering the within-cluster sum of squares.

An evaluation considering the convergence rate (Fig. 2 to Fig. 4) of competitive and cooperative EAs reveals that CCEAs present a faster convergence speed than their corresponding competitive EAs, once they may perform better exploitations of the problem space than sheer competitive methods. The CCEAs also presented better capabilities to escape and avoid local minima points than their corresponding competitive models.

An overall evaluation considering the Friedman/Nemenyi ranks showed that CCPSO was able to achieve the best average performances considering J, J_e and D_{max} measures. CCGA and CCGSO have also presented better average performances (considering J, J_e and D_{max}) than their corresponding competitive approaches (GA and GSO, respectively). The only situation where a CCEA presented a slightly worse average performance considering these three metrics than its corresponding competitive model was for Differential Evolution. In terms of Intra-Cluster Separation, both CCDE and CCGSO presented better average performances than their corresponding competitive methods, what means that CCEAs are able

TABLE IV
EXPERIMENTAL RESULTS (CONT.).

Dataset	Algorithm	J		J_e		D_{max}		D_{min}	
		Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
Heart	GA	558525.8	4550.3	2190.7	2214.4	2738.7	59.982	7043.7	976.57
	CCGA	564903.9	9443.7	2214.4	66.859	2751.2	56.246	6941.5	1288.2
	DE	551505.1	3371.8	2177.3	23.434	2731.5	22.271	6647.8	185.98
	CCDE	557510.8	10537.2	2195.8	37.920	2754.8	50.464	6751.6	442.00
	PSO	549643.0	284.51	2166.1	4.2396	2718.1	4.2788	6656.7	30.730
	CCPSO	549730.0	289.64	2166.2	3.8776	2717.8	3.9176	6658.5	26.112
	GSO	551582.3	2547.6	2178.5	15.425	2735.7	28.319	6685.1	82.991
	CCGSO	550639.1	1363.8	2172.1	7.8469	2724.9	11.878	6677.0	63.619
Ionosphere	GA	2930.71372	187.655	8.58926	1.2672	11.68574	1.85489	11.36489	3.66450
	CCGA	2720.01382	89.102	7.91464	0.41943	11.13668	0.68372	9.94826	2.51786
	DE	2678.01868	132.450	7.97840	0.66261	10.96695	0.74460	9.36215	1.92786
	CCDE	2636.94547	108.718	7.85082	0.45704	11.10343	0.52975	9.72613	2.00591
	PSO	2509.50569	87.622	7.38134	0.41593	10.46709	0.32101	10.93054	1.11969
	CCPSO	2463.46297	28.96170	7.23143	0.10799	10.54692	0.16089	10.25182	0.84376
	GSO	2550.97605	93.76644	7.55314	0.23765	10.59834	0.30448	7.86152	1.18567
	CCGSO	2482.01837	32.69989	7.37320	0.14909	10.67072	0.20967	8.78221	0.64195
Iris	GA	81.22521	1.18469	0.54166	0.00787	0.69786	0.04512	3.02611	0.31315
	CCGA	82.55626	1.64062	0.55035	0.01052	0.69955	0.03424	3.15443	0.49260
	DE	79.88360	2.12919	0.53309	0.01602	0.66097	0.03956	3.12798	0.26966
	CCDE	80.18489	2.35443	0.53512	0.01626	0.66589	0.03444	3.19794	0.30647
	PSO	79.03144	0.40391	0.52682	0.00308	0.64571	0.00459	3.21821	0.01732
	CCPSO	78.96015	0.32592	0.52672	0.00300	0.64696	0.00476	3.21388	0.01590
	GSO	79.40203	0.98442	0.53077	0.00684	0.66376	0.03141	3.17729	0.11815
	CCGSO	78.87785	0.03219	0.52656	0.00143	0.64839	0.00473	3.19795	0.03038
Seeds	GA	610.08371	12.47485	2.90747	0.05113	3.17072	0.12631	13.17701	1.63531
	CCGA	610.57837	13.78568	2.90862	0.06466	3.17043	0.15618	13.58649	1.69849
	DE	609.25667	27.82266	2.91437	0.13069	3.15599	0.17468	13.38245	0.87344
	CCDE	594.23178	14.68215	2.84236	0.06575	3.11244	0.30702	13.31207	0.51383
	PSO	589.37119	3.02828	2.82295	0.01458	3.02836	0.02927	13.40255	0.13293
	CCPSO	587.53163	0.50746	2.81255	0.00381	3.01836	0.00038	13.36232	0.02987
	GSO	591.87508	3.95263	2.82763	0.01939	3.07430	0.10329	13.11010	0.38706
	CCGSO	588.08559	0.77466	2.81156	0.00529	3.01942	0.00170	13.29216	0.12147
Wine	GA	2428579.2	88192.5	15179.8	484.96	27628.6	3116.5	85034.1	24907.4
	CCGA	2478201.0	99223.4	15474.5	596.21	27858.8	2969.1	85211.5	24882.3
	DE	2378836.1	4609.2	14884.3	25.236	29005.7	35.057	73099.7	1534.6
	CCDE	2433167.2	108972.6	15193.1	616.22	27074.7	3521.2	88177.3	27335.6
	PSO	2372065.9	6263.5	14845.9	30.248	28957.4	1.0027	73150.9	226.75
	CCPSO	2404824.1	87992.3	15034.0	502.70	27895.5	2752.3	81489.3	21769.2
	GSO	2379558.9	32743.0	14890.0	189.12	28905.5	357.80	72529.7	1549.9
	CCGSO	2438745.2	114474.3	15227.8	653.90	26834.2	3580.9	89865.6	28307.2

TABLE V
OVERALL EVALUATION: AVERAGE RANKS FOR THE FRIEDMAN/NEMENYI TEST FOR EACH METRIC, WITH $CD = 3.1656$. **BOLD**: THE BEST AVERAGE RANK BETWEEN AN EA AND THE CORRESPONDING CCEA; †: THE OVERALL BEST AVERAGE RANK.

Algorithm	Rank$_J$	Rank$_{J_e}$	Rank$_{D_{max}}$	Rank$_{D_{min}}$
GA	192.1273	174.4394	160.9561	**128.7606**
CCGA	**184.7848**	**163.8091**	**153.4045**	124.0545
DE	**136.1333**	**132.1939**	**130.6576**	106.2394
CCDE	139.5152	137.6970	134.1879	**126.7788**
PSO	74.0970	83.4758	92.1273	**140.5576**†
CCPSO	**49.1333**†	**59.6424**†	**69.5121**†	134.7242
GSO	111.3667	121.8242	122.4121	94.8394
CCGSO	**76.8424**	**90.9182**	**100.7424**	**108.0455**

to find more compact clusters than competitive EAs, but they may generate partitions with clusters that are not very much distant one from the other.

The experimental results showed that CCEAs are able to obtain better average performances than their corresponding competitive EAs is most of the cases, so they are good solutions to tackle clustering problems. Also, with the adopted encoding schema for the population individuals, such methodologies present huge advantages in relation to competitive methods in terms of storage memory when the number of intended clusters increases.

IV. CONCLUSION

In this work, a we evaluate the cooperative coevolutionary framework for evolutionary algorithms in the context of partitional clustering problems. We also compare the behavior of cooperative coevolutionary methods in relation to standard competitive evolutionary approaches when tackling clustering task.

In the adopted population encoding schema the population is split in several sub-populations, in such a way that each sub-population is responsible for the optimization of only one cluster centroid each time. The clustering problem (global problem) is solved by the combination of the best local solutions found so far by each sub-population. This representation requires less storage memory in comparison to the

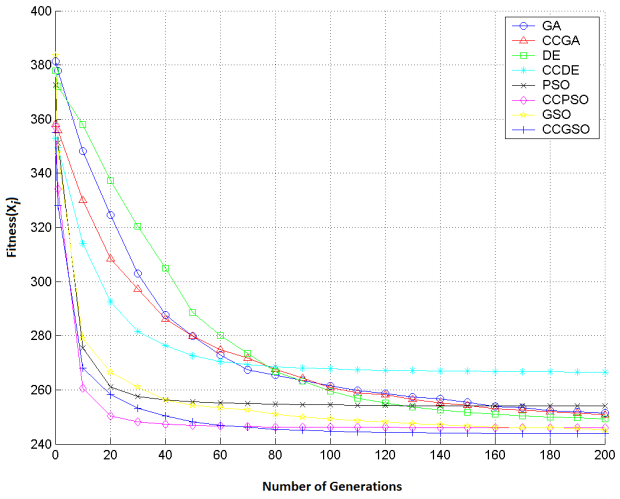

Fig. 2. Convergence graph for Cancer dataset.

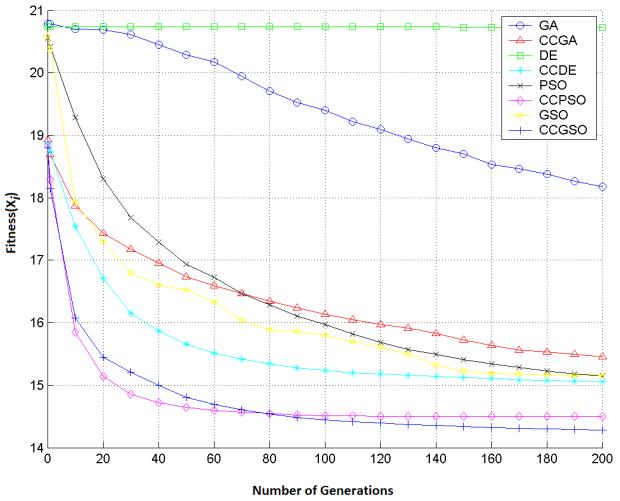

Fig. 3. Convergence graph for E. Coli dataset.

standard representation approach for partitional evolutionary algorithms, once each individual represents just one cluster centroid each time, instead of representing a complete set of cluster centroids.

The comparison framework took into consideration four well-established evolutionary and swarm intelligence algorithms from evolutionary computing literature: Genetic Algorithm, Differential Evolution, Particle Swarm Optimization and Group Search Optimization. Each one of the selected algorithms has been implemented as both competitive and cooperative partitional clustering models (GA/CCGA, DE/CCDE, PSO/CCPSO, and GSO/CCGSO). Eleven benchmark real world problems from UCI Machine Learning Repository have been selected as the testing bed for the experimentation. The evaluation included a rank system obtained by Friedman/Nemenyi hypothesis test in relation to four clustering quality measures (the Within-Cluster Sum of Squares, the Quantization Error, the Intra-Cluster Distance, and the Inter-

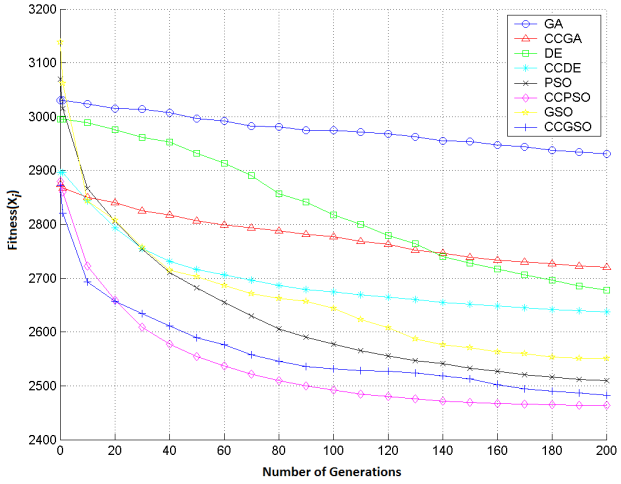

Fig. 4. Convergence graph for Ionosphere dataset.

Cluster Separation). The experimental results showed the potential of CC approaches over sheer competitive methods, according to Friedman/Nemenyi tests, considering three out of four clustering metrics (J, J_e and D_{max}), except for Differential Evolution algorithm, where the competitive method showed performances slightly better than the corresponding cooperative coevolutionary model.

As future works, we intend to evaluate the behavior of other EAs and SIs when adapted to CC model as partitional clustering methods. We also intend to extend the testing bed by adding other real world problems, as much as synthetic datasets, so we could have a better evaluation on the generalization performances of such approaches and their behavior when dealing with specific clustering problems (such as unbalanced datasets, classes with different shapes, and so on). It is worth to mention that the main advantages offered by CCEAs when dealing with data clustering problems are highlighted when the number of intended clusters increases, so the future evaluations will also include tests with datasets presenting a high number of classes.

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] 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.

[3] L. Kaufman and P. Rousseeuw, *Clustering by means of medoids*. North-Holland, 1987.

[4] E. R. Hruschka, R. J. Campello, A. A. Freitas *et al.*, "A survey of evolutionary algorithms for clustering," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 39, no. 2, pp. 133–155, 2009.

[5] S. Alam, G. Dobbie, Y. S. Koh, P. Riddle, and S. U. Rehman, "Research on particle swarm optimization based clustering: a systematic review of literature and techniques," *Swarm and Evolutionary Computation*, vol. 17, pp. 1–13, 2014.

[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. H. Holland, "Genetic algorithms," *Scientific american*, vol. 267, no. 1, pp. 66–72, 1992.

[8] R. Storn and K. Price, "Differential evolution—a 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.

[9] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm intelligence: from natural to artificial systems*. Oxford university press New York, 1999, vol. 4.

[10] 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.

[11] J. Kennedy, R. C. Eberhart, and Y. Shi, "Swarm intelligence. 2001," *Kaufmann, San Francisco*, 2001.

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

[13] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *International Conference on Parallel Problem Solving from Nature*. Springer, 1994, pp. 249–257.

[14] S. H. Clearwater, T. Hogg, and B. A. Huberman, "Cooperative problem solving," in *Computation: The micro and the macro view*. World Scientific, 1992, pp. 33–70.

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

[16] E. Cantu-Paz, *Efficient and accurate parallel genetic algorithms*. Springer Science & Business Media, 2000, vol. 1.

[17] E. M. Figueiredo and T. B. Ludermir, "Investigating the use of alternative topologies on performance of the pso-elm," *Neurocomputing*, vol. 127, pp. 4–12, 2014.

[18] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Scaling up fast evolutionary programming with cooperative coevolution," in *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546)*, vol. 2. Ieee, 2001, pp. 1101–1108.

[19] Y.-j. Shi, H.-f. Teng, and Z.-q. Li, "Cooperative co-evolutionary differential evolution for function optimization," in *International Conference on Natural Computation*. Springer, 2005, pp. 1080–1088.

[20] G. A. Trunfio, "A cooperative coevolutionary differential evolution algorithm with adaptive subcomponents," *Procedia Computer Science*, vol. 51, pp. 834–844, 2015.

[21] F. Van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE transactions on evolutionary computation*, vol. 8, no. 3, pp. 225–239, 2004.

[22] K. Georgieva and A. P. Engelbrecht, "A cooperative multi-population approach to clustering temporal data," in *2013 IEEE Congress on Evolutionary Computation*. IEEE, 2013, pp. 1983–1991.

[23] L. D. S. Pacifico and T. B. Ludermir, "Cooperative group search optimization," in *2013 IEEE Congress on Evolutionary Computation*. IEEE, 2013, pp. 3299–3306.

[24] ——, "A partitional cooperative coevolutionary group search optimization approach for data clustering," in *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*. IEEE, 2019, pp. 347–352.

[25] X. Ma, X. Li, Q. Zhang, K. Tang, Z. Liang, W. Xie, and Z. Zhu, "A survey on cooperative co-evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 3, pp. 421–441, 2018.

[26] 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.

[27] A. Ahmadyfard and H. Modares, "Combining pso and k-means to enhance data clustering," in *Telecommunications, 2008. IST 2008. International Symposium on*. IEEE, 2008, pp. 688–691.

[28] K. A. Prabha and N. K. Visalakshi, "Improved particle swarm optimization based k-means clustering," in *2014 International Conference on Intelligent Computing Applications*. IEEE, 2014, pp. 59–63.

[29] L. D. S. Pacifico and T. B. Ludermir, "Hybrid k-means and improved self-adaptive particle swarm optimization for data clustering," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–7.

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

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

[32] A. Mortezanezhad and E. Daneshifar, "Big-data clustering with genetic algorithm," in *2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI)*. IEEE, 2019, pp. 702–706.

[33] V. S. Srinivas, A. Srikrishna, and B. E. Reddy, "Automatic feature subset selection for clustering images using differential evolution," in *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 2018, pp. 216–217.

[34] T. Li and H. Dong, "Unsupervised feature selection and clustering optimization based on improved differential evolution," *IEEE Access*, vol. 7, pp. 140 438–140 450, 2019.

[35] L. Xiao, Q. Ma, and H. Wang, "Improved pso ert image reconstruction algorithm for human lung based on prior knowledge and clustering," in *2018 Chinese Control And Decision Conference (CCDC)*. IEEE, 2018, pp. 5840–5844.

[36] L. D. S. Pacifico and T. B. Ludermir, "Hybrid k-means and improved group search optimization methods for data clustering," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[37] R. F. Abdel-Kader, "Genetically improved pso algorithm for efficient data clustering," in *Machine Learning and Computing (ICMLC), 2010 Second International Conference on*. IEEE, 2010, pp. 71–75.

[38] L. D. S. Pacifico and T. B. Ludermir, "A group search optimization method for data clustering," in *Intelligent Systems (BRACIS), 2014 Brazilian Conference on*. IEEE, 2014, pp. 342–347.

[39] R. Storn and K. Price, "Differential evolution–a simple , efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[40] 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.

[41] 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.

[42] 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.

[43] ——, "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.

[44] 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.

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