Efficiently Finding the Optimum Number of Clusters in a Dataset with a New Hybrid Cellular Evolutionary Algorithm

Javier Arellano-Verdejo, Adolfo Guzmán-Arenas, Salvador Godoy-Calderon, and Ricardo Barrón Fernández

Centro de Investigación en Computación, Instituto Politécnico Nacional, México D.F., Mexico

jarellanob10@sagitario.cic.ipn.mx, a.guzman@acm.org, sgodoyc@cic.ipn.mx, rbarron@cic.ipn.mx

Abstract. A challenge in hybrid evolutionary algorithms is to employ efficient strategies to cover all the search space, applying local search only in actually promising search areas; on the other hand, clustering algorithms, a fundamental base for data mining procedures and learning techniques, suffer from the lack of efficient methods for determining the optimal number of clusters to be found in an arbitrary dataset. Some existing methods use evolutionary algorithms with cluster validation index as the objective function. In this article, a new cellular evolutionary algorithm based on a hybrid model of global and local heuristic search is proposed for the same task, and extensive experimentation is done with different datasets and indexes.

Keywords. Clustering, cellular genetic algorithm, micro-evolutionary algorithms, particle swarm optimization, optimal number of clusters.

Búsqueda eficiente del óptimo número de grupos en un conjunto de datos con un nuevo algoritmo evolutivo celular híbrido

Resumen. Un reto actual en el área de algoritmos evolutivos híbridos es el empleo eficiente de estrategias para cubrir la totalidad del espacio de búsqueda usando búsqueda local sólo en las regiones prometedoras. Por otra parte, los algoritmos de agrupamiento, fundamentales para procesos de minería de datos y técnicas de aprendizaje, carecen de métodos eficientes para determinar el número óptimo de grupos a formar a partir de un conjunto de datos. Algunos de los métodos existentes hacen uso de algunos algoritmos evolutivos, así como una función para validación de agrupamientos como su función objetivo. En este artículo se propone un nuevo algoritmo evolutivo celular, para abordar dicha tarea. El algoritmo propuesto está basado en un modelo híbrido de búsqueda, tanto global como local y tras presentarlo se prueba con una estensa experimentación sobre diferentes conjuntos de datos y diferentes funciones objetivo.

Palabras clave. Agrupamiento, algoritmo gentico celular, microalgoritmos evolutvos, optimizacin por cmulo de partculas, nmero ptimo de clases.

1 Introduction

Modern search methods for optimization consider hybrid evolutionary algorithms (HEA) which combine evolutionary algorithm (EA) and local optimizers. The hybridism comes from the balancing of global and local search procedures, and the main focus of such models is real-world problems. In recent years, the integration of different learning and adaptation techniques in order to overcome individual limitations and achieve synergetic effects through hybridization or fusion of these techniques has contributed to a large number of new hybrid evolutionary algorithms [16].

On the other hand, a clustering algorithm is a procedure that groups data patterns according to their similarity [19] and [44]. Clustering is a very useful and popular task among artificial intelligence activities; consequently, numerous algorithms and procedures for solving clustering problems have been reported: centroid based methods [17], graph theory based clustering [35], fuzzy [30], probability accumulation based clustering [41], hierarchical [8], and kernel methods for clustering [37].

Once a dataset has been clustered, there is a certain warranty that elements from the same cluster are more similar to each other than to those in other clusters, therefore, they share some common properties [44]. By using inductive learning procedures like those in [23], knowledge about the cluster structure of a multidimensional dataset can be used to construct a synthetic and characteristic definition of each cluster. Such characteristic definition is generally used as the core learning information for a wide variety of intelligent agents, concept generalization mechanisms, and all sorts of knowledge discovery procedures [26] and [27]. That is why clustering algorithms are most valuable tools for data mining processes.

During the last few decades cluster analysis has played a major role in a wide variety of research fields, including microbiology, genetics, psychology, marketing, economics, machine learning [3], computer vision [13], and many others [7]. However, a major family of clustering algorithms (aka. centroid based or partitional ones), heavily depend on the number of clusters to form, and although several rules of thumb can be used to estimate this number, there is no definite rule for that.

The problem of finding the optimum number *k* of clusters for an arbitrary dataset has distinguished itself as one of the most intriguing and challenging problems in the Pattern Recognition field [10]. More than often, the choice of this number determines the behavior of several learning applications such as [18], [46], and [38] and has profound implications, as well as interpretations, regarding the nature and properties of the dataset being clustered [42]. Once this number *k* has been estimated, there is a wide variety of algorithms like [45] and [6] that take it as a parameter and effectively decide a way to cluster all available data into exactly that number of clusters. However, there is still no universal criterion or algorithm for estimating k.

1.1 Clustering Problem

The clustering problem can be formally defined as follows. Let $\Omega = \{o_1, \cdots, o_n\}$ be a set of r dimensional patterns. A clustering of Ω is a partitioning of Ω into k clusters $\{C_1, \cdots, C_k\}$ satisfying the following conditions:

— The union of all clusters must be equal to the original universe of discourse: $\bigcup_{j=1}^{k} C_j = \Omega$ for all $i, j \in [1,k]$.

— Each pattern can belong to one and only one cluster: $C_i \bigcap C_j = \emptyset$, for all $i, j \in [1,k]$.

— There can be no empty clusters: $C_j \neq \emptyset$, for all $i, j \in [1,k]$.

However, the problem of estimating k is formally defined in [10] as an optimization problem. Given the fact that a dataset can be partitioned in several different ways while preserving the aforementioned properties, an objective (target) function for the optimization process must be defined. The problem then turns out to be one of finding a partition C^* of optimal or near-optimal adequacy, as compared to all other feasible solutions.

The well-known intrinsic relationship between clustering and optimization has attracted the attention of researchers in the field of meta-heuristic optimization to the problem of estimating k for any given dataset. Numerous research papers have been published where all kinds of evolutionary algorithms are used for tackling the problem of estimating k [2], [10], [36], and [42] and the problem of finding an appropriate clustering of data into exactly k clusters [1] [21], [28], [33], and [25]. In this paper we try to provide a new best-performing algorithm, to the best of our knowledge, for the same task.

When *k* is known a priori, an adequate clustering of a dataset can be found by determining the position of all cluster representatives that minimizes the global intra-cluster variance. On the other hand, when estimating *k*, each possibility can only be tested with the aid of a global structure-related measure, such as a cluster validation index. However, since a dataset can be clustered in different ways with the same number of clusters, all meta-heuristic methods that seek to estimate *k* at the same time also find the best clustering of data in that particular number of clusters. These methods, characterized by the use of an internal cluster validation index as the fitness function, model the clustering problem as a combinatorial optimization [20].

In this article, we present a novel evolutionary algorithm which implements a hybrid model of heuristic search. We experimentally test the efficiency of the proposed algorithm for finding the optimal number of clusters in a dataset, and compare its performance with that of the most widely used evolutionary techniques for the same problem and under the same conditions. Exhaustive experimentation with synthetic datasets, as well as the use of non-parametric statistical tests, convincingly shows that the proposed algorithm is the most efficient evolutionary algorithm used for that problem so far.

2 State of the Art

In this section, the state of the art on the main techniques used during the development of this research work will be briefly described. In Section 2.1, most representative research works regarding the determination of the optimal number of clusters (value of k) in a dataset are described. Section 2.2 describes the latest advancements in cellular genetic algorithms as well as their different possible configurations. Finally, in Section 2.3 micro-evolutionary algorithms are analyzed, and their advantages in convergence speed and reduced population size are discussed.

2.1 Finding Optimal Value for *K*

In [2] the authors proposed a genetic algorithm based procedure called Genetic Clustering for Unknown *K* (GCUK-Clustering). It was developed for finding the optimal number of clusters in a dataset. The authors only reported experiments with the Davis-Boulding index as the fitness function and with both synthetic (Data_3_2, Data_5_2, Data_6_2 and Data_4_3) and real datasets (IrisData and Breast Cancer sets from the Machine Learning Repository). The GUCK-Clustering algorithm successfully found the correct number of clusters in all reported experiments.

Six years later, the authors of [10] developed a differential evolution algorithm called A Classical Differential Evolution for clustering (ACDE) for the same purpose. This algorithm encodes the cluster representatives in each chromosome, but it also adds, for each representative, the probability of that particular representative to be active in that individual. The ACDE algorithm also uses the Davis-Boulding index as its fitness function, and experiments only with the IrisData, Breast-Cancer and Wine datasets from the same UCI repository were reported. Again, the correct number of clusters was found in all reported experiments.

Apparently, as a response to the previous paper, [36] presented another differential evolution algorithm called ADEFC (A new Differential Evolution based on Fuzzy Clustering) for finding the appropriate number of clusters in a fuzzy dataset. Almost with the same genotype coding explained before, ADEFC encodes each cluster representative with real numbers, but with an extra array of bits for activating/deactivating each representative. The authors used the Xie-Beni index as the fitness function, and although in all their reported experiments the optimal number of clusters was found correctly, almost for the first time in this kind of research, they also used the Wilcoxon non-parametric ranking test to support their experimental results.

Lastly, the authors of [42] devised a differential evolution based algorithm called Automatic Clustering with Differential Evolution; it used the cluster number Oscillation method (ACDE-O) which includes an oscillation mechanism. This algorithm, while having the same structure of a canonical differential evolution, also has the ability to locally adjust the estimated number of clusters based on activation of the information stored in the individual's genotype. The authors used the I-index (designed by Maulik and Bandyopadhyay) as the fitness function. IrisData and Breast Cancer datasets were used in reported experimentation; however, no statistical test was used for evaluation purposes.

2.2 Cellular Evolutionary Algorithms (cEAs)

The first cGA was designed by Robertson in 1987 [216-22] for a classification system. Robertson's idea was to construct a parallel genetic algorithm where each individual was connected with others in its neighborhood. His experimental results were quite encouraging, since the cGA got better results than a classic GA.

In 1991, Davidor [58-27] experimented with cGA and bi-dimensional grids as a means for structuring the population into 8-individual neighborhoods. He used the proportional selection method to select ancestors for recombination and to reinsert individuals into the neighborhood with a probability based on their fitness value. Davidor concluded that cGA with the aforementioned characteristics had faster convergence speeds than the traditional cGA.

In 1994, Gordon et al. [110-28] studied the behavior of cGAs with different types of neighborhoods. They used cGA on a wide variety of optimization problems, both continuous and discreet, and concluded that neighborhoods with bigger neighborhood radius work better with simple problems. However, for more complex problems, cGAs with smaller neighborhood radius work better.

In 2002, Alba et al. [19-28] performed a comparative research of different individual update criteria in both synchronous and asynchronous modes. Experimental results obtained by them suggest that asynchronous cGAs show a higher selection pressure than synchronous cGAs, so their convergence speed is also faster and they find solutions faster in less complex problems. On the other hand, synchronous cGAs show a better performance when facing complex problems.

Also in 2002, Ekund [84-28] made an extensive empirical research in order to determine the best selection criterion for cGAs as well as the optimal shape and size for neighborhoods. He concluded that shape and size of the neighborhoods directly depend on the size of the population. He also showed that any selection criterion can lead to good results as long as an elitism criterion is also used to preserve the best solutions found.

2.3 Micro-Evolutionary Algorithms (μEAs)

In 1989, [24] presented the first implementation of a μEA in the form of a micro-Genetic Algorithm (μGA) with a very small population and with elitism. He tested his μGA against a conventional Genetic Algorithm (GA) and found that the μGA was able to get better results when dealing with stationary functions as well as in certain control engineering problems.

That same year Goldberg suggested in [15], basing on some theoretical results, a μGA that starts with a reduced population and applies recombination operators such as crossover and mutation until the population reaches a state of nominal convergences. This algorithm randomly reinitializes the population but keeps the best fitted individuals from the last generation. All the above processes are repeated until a certain termination condition is met. Following his experimental results, Goldberg found that a population with only three individuals is enough to ensure the algorithm's convergence, independently from the length of the chromosomes.

As described in the state of the art, μEAs can be classified into two classes: (1) those that are modified versions of original EAs and (2) those specifically designed to work with small populations [40]. The most representative algorithms of the first class are μGA [24], μPSO [4], and μDE [34]. On the other hand, one of the few known algorithms in the second class is Elitist Evolution (EEv) designed in 2009 [39].

Concerning more recent studies, in 2010 Coello et al. proposed a μPSO for multi-objective problems (μMOPSO) [5]. In this algorithm, the swarm's leader selection is made with the Pareto efficiency parameter, a density-based neighborhood estimator, a mutation operation, and a re-initialization process. The experimental results show that for all tested objective functions, their algorithm finds better solution and with a faster convergence time than the NSGA-II algorithm [12], a representative multi-objective algorithm.

2.4 Hybrid Evolutionary Algorithms (HEA)

Even though evolutionary computation has clearly shown its advantages for tackling difficult practical optimization problems, there are many reported cases in which the performance yielded by direct evolutionary algorithms turns out to be marginal at best. On several other occasions the practical complexity in selecting parameter values and appropriate representations highlight the limitations of the direct approach. All these facts clearly illustrate the need for hybrid approaches to optimize the performance of the direct evolutionary approach. Recently, hybridization of evolutionary algorithms has become popular due to their capabilities in handling several real world problems involving complexity, noisy environment, imprecision, uncertainty, and vagueness.

Several hybridization approaches have been used to improve the general efficiency of evolutionary algorithms. The most widely used technique is the hybridization of one evolutionary algorithm with another one having different exploration/exploitation capabilities. Examples of such approach include neural network assisted EA, swarm assisted EA, and ant colony assisted EA. Still another popular approach is the hybridization of some EA and other heuristics such as local search, taboo search, simulated annealing, hill climbing, etc.

Noteworthy is the case of [32], where the authors propose a new hybrid EA for clustering tasks. The proposed HEA combines an ACO with a Simulated Annealing (SA) algorithm for ACO-SA. Although it is very similar in structure to the hybridization technique used during this research, the marked difference between the objectives of both studies as well as their distinct experimental evaluation processes make them virtually impossible to compare. On the one hand, in [32] the HEA is used for clustering tasks and the desired total number of clusters is received as a user parameter. On the other hand, in [32] the evaluation process is based on a Minimal Squared Error (MSE). The primary objective of our research is to estimate the total number of clusters to be found in a dataset and not to directly cluster such data.

3 Theoretical Background

A description of the theoretical framework for the methods and techniques used during this research is briefly presented in this section. Section 3.1 describes the interconnection topology and the neighborhoods of the proposed cGA. Section 3.2 shows the PSO used for local search including its pseudocode. Finally, Section 3.3 describes the cluster validation indexes used as the fitness function for the proposed cGA.

3.1 Cellular Genetic Algorithm

Cellular Genetic Algorithms (cGA), like many other evolutionary algorithms, search for the solution to a problem by evolving a population of candidate solutions. However, cGAs model the evolutionary process in a more individual-oriented perspective rather than the population-oriented perspective typical for many other models like genetic algorithms, evolutionary strategies, and differential evolution. In a cGA, a grid provides structure for the population (see Fig. 1).

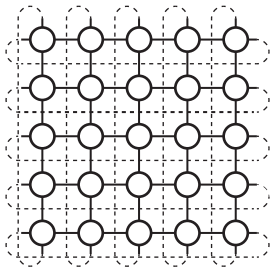

Fig. 1. cGA population structure

The grid, along with the selection of a neighborhood model, sets the interaction and communication processes to take place during the population evolution. Unlike what happens in population-oriented meta-heuristics, in a cGA each individual can only recombine with those that are part of its local neighborhood. Such behavior significantly modifies the properties of the evolutionary meta-heuristic search. On the one hand, the relatively small size of an individual's neighborhood causes its diversity to decrease quite rapidly during the evolutionary process. On the other hand, since all neighborhoods overlap, an individual has as much chance of being replaced as the number of neighborhoods of which it is part. The size of the overlap sets the relative speed at which the best solutions are propagated through the whole population or takeover (see Fig. 2).

The exploration/exploitation capabilities of a cGA can be regulated by varying the size and topology of the neighborhoods. There are two classical ways to define a

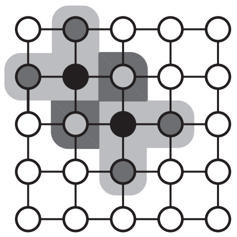

Fig. 2. Neighborhoods overlapping

neighborhood: the *Cn* compact form and the *Ln* linear form (see Fig. 3), where *n* indicates the size of the neighborhood in both cases. The behavior of a cGA notoriously depends on its individual replacement policy as well as on other parameters such as the size and geometry of neighborhoods, see the pseudocode of the canonical cGA algorithm 1. Here, in lines 1 and 2, the initial population is created, initialized and evaluated. Afterwards, for each individual in line 5, the neighborhood of the current individual is calculated. Line 6 selects both parents to be recombined (line 7) and mutated (line 8). In line 10, the current individual is replaced with the best-fitted mutated descendant. The whole process is run while the stop-condition is not met (line 3).

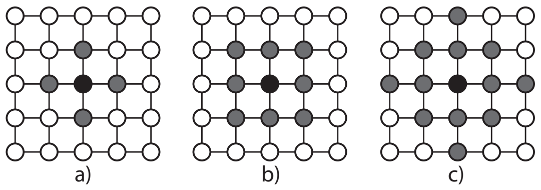

Fig. 3. Typically used neighborhoods: a) L5, B) C9, c) C13

3.2 MicroPSO with Local Search

A Particle Swarm Optimization algorithm (PSO) is a population meta-heuristic inspired by the social behavior observed in flocks of birds or shoals of fish. This kind of algorithm follows a biological model known as social metaphor [22], which states that when individuals take part in a society, their opinions emerge partly from their own experience and partly from their society's set of beliefs (the search space).

A μPSO is a PSO with a very small population (typically between 3 and 5 individuals). Unlike classical swarm algorithms, a μPSO operates along two loops (see the pseudocode 2 for μPSO algorithm): the inner

Algorithm 1 Pseudocode of the cGA

```
 1: p ← generateInitialPopulation()
 2: evaluate($population$)
 3: while not terminationConditionMeet do
 4:   for each individual in p do
 5:     neighboors ← getNeighborhood(p, individual)
 6:     parents ← selection(neighboors)
 7:     offspring ← recombination(parents)
 8:     offspring ← mutation(offspring)
 9:     evaluate(offspring)
10:     $p_{temp}$ ← replacement(individual,offspring)
11:   end for
12:   p ← $p_{temp}$
13: end while
14: return bestSolutionFound
```

loop (lines 7 to 16) works exactly as in a canonical PSO until it finds a best solution; the outer loop (line 4) re-initializes the swarm with a preset frequency (line 18). The re-initialization of the swarm yields new random individuals and preserves the best fitted individual from the previous swarm, thus promoting the exploration capabilities of the algorithm.

Algorithm 2 Pseudocode of the μPSO-LS

```
 1: swarm ← initSwarm()
 2: bestParticlePosition ← getBestParticlePosition()
 3: globalBest ← getGlobalBest()
 4: while not terminationConditionMeet do
 5:   while not nominalConvergenceMeet do
 6:     for each particle in swarm do
 7:       $v$ ← calculateVelocity(particle)
 8:       $x$ ← calculatePosition(particle,v)
 9:       $x$ ← localSearch($x$)
10:       if fitness($x$) $\geq$ fitness(bestParticlePosition) then
11:         bestParticlePosition ← $x$
12:         if fitness($x$) $\geq$ fitness(globalBest) then
13:           globalBest ← $x$
14:         end if
15:       end if
16:     end for
17:   end while
18:   swarm ← restartSwarm(swarm, globalBest)
19: end while
20: return bestSolutionFound
```

3.3 Cluster Validation Indexes

Different clustering algorithms can deliver very different results on the same dataset [29]. The aim of the cluster validity is to find the partitioning that best fits the underlying data. Many validation indexes exist; however, only three of them will be used for this research: the Davis-Bouldin (DB) index, the Xie-Beni (XB) index, and the I-index (*I*).

In the DB index [11], Equation 1, a similarity measure between the clusters C_i and C_j is defined based on a measure of dispersion of a cluster C_i and a comparison with the separation between both clusters. DB is the average similarity between each cluster C_i and its most similar one:

$$DB = \frac{1}{k}\sum_{i=1}^{k} \max_{\substack{j=1..k \\ j \neq i}} \left\{ \frac{Disp(C_i)+Disp(C_j)}{Sep\left(C_i,C_j\right)} \right\}, \quad (1)$$

where $Disp(C_i)$ is the dispersion of class C_i calculated as

$$Dist \quad (C_i) = \sqrt{\frac{1}{|C_i|} \sum_{\substack{O_i,O_j \in C_i \\ i \neq j}} \left\| o_i, o_j \right\|^2} \quad (2)$$

and $Sep(C_i,C_j)$ is the separation between class i and class j, measured with any appropriate linkage function (single, complete, average, etc.).

In the Xie-Beni index [43], Equation 3, the fuzzy deviation of x_i from cluster j is defined as the distance between x_i and the center of the cluster weighted by its fuzzy membership to that same cluster j:

$$XB = \frac{\sum_{j=1}^{k}\sum_{i=1}^{n} \mu_{ij} \left\| o_i, u_j \right\|^2}{n \left(\min\limits_{\substack{p=1..k \\ p \neq j}} \left\{ \left\| u_j, u_p \right\|^2 \right\} \right)}, \quad (3)$$

where μ_{ij} is the membership of object i in class j and u_j is the selected class representative object (class centroid) of class j.

Finally, in the I-index [31], Equation 4, the first factor normalizes each index value by the overall number of clusters k, the second term sets the overall error sum of squares of the complete dataset in relation to the intra-cluster error of a given clustering. Lastly, the third factor incorporates the maximum observed difference between two of the k clusters. The index computation includes a variable parameter $p \in \mathbb{R}$ to control the contrast between the different cluster configurations. The authors recommend a value of $p = 2$:

$$I = \left(\frac{1}{k} \cdot \frac{E_1}{E_k} \cdot D_k \right)^p, \quad (4)$$

where E_k is the weighted variance of class j, E_1 is the global non-weighted variance of all patterns, and D_k is the maximum distance between classes. Variable p is the only scale factor which authors recommend to be 2.

4 Hybrid cGA

The algorithm herein proposed is a Hybrid Cellular Genetic Algorithm for finding the optimal number of clusters in a dataset. The proposed algorithm incorporates a local search technique taken from the Micro-Particle Swarm Optimization model which increases the efficiency with respect to the works discussed in the state of the art. In addition, our proposal is very simple to understand and implement (Occam's razor); we test it on a superset of problems taken from the mentioned literature and assess the statistical validation of our results. This, we guess, represents a self-contained model and hopefully, a useful contribution for future research in a wide variety of domains using clustering.

This section presents the proposed cellular genetic algorithm for finding the optimal number of clusters in any given dataset including the computational representation for individuals (4.1), the fitness function (4.2), as well as datasets, configuration, statistical and analysis results (Sections 5, 5.1, and 5.5).

The proposed algorithm is a hybrid evolutionary algorithm made up of two heuristics: a cGA and a μPSOLS (see algorithms 1 and 2 in Section 3.1 and 3.2).

The upper part of Fig. 4 shows a cGA, with its population structured by a toroidal grid, and where each individual represents one possible solution (one value for k). The lower part of the same figure shows that a μPSO, with local search capabilities (LS), is used to determine the optimal clustering of objects for the individual's value of *k*.

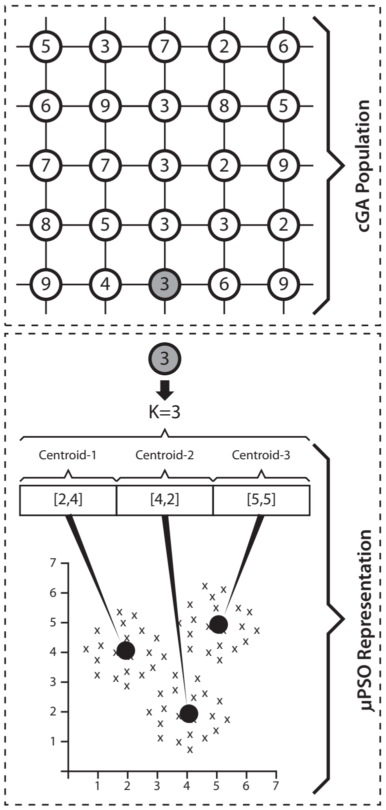

Fig. 4. Architecture of the proposed CGA

4.1 Computational Representation

As stated in the previous paragraph, the cGA works over a population structured with a completely connected grid (toroid). Each individual has a different value within the search range for *k*. The μPSO uses an array of length *k* to represent each individual. Each position in such array, codes the centroid's position for each of the *k* clusters represented by a cGA individual.

4.2 Fitness Function

Fig. 5 shows a graphical representation of the proposed fitness evaluation mechanism. The μPSO takes a k value as input (from the cGA) and returns an optimal partition of the objects with exactly that number of clusters. The partition is then assessed by using a cluster validation index (see Section 2) which in turn yields the fitness value required by the cGA.

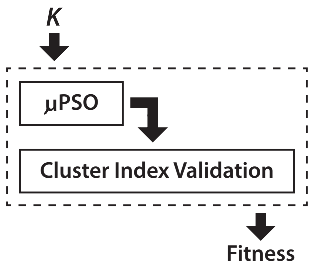

Fig. 5. Individual evaluation mechanism for the CGA

In order to enhance the performance of the μPSO, a local search was included within its evolutionary process. The left part of Fig. 6 shows the μPSO representation of one particular centroid for a cluster. After performing the local search procedure, the position of the centroid is adjusted to better fit the cluster. Such adjustment induces an increase in the algorithm convergence speed, as it can be seen in the experimental results section.

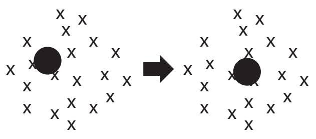

Fig. 6. μPSO local search

4.3 The cGA/μPSOLS Algorithm

As it can be seen in the pseudocode 1, the algorithm generates the initial population with a uniform random distribution (line 1 of the pseudocode). Using the method explained in Section 4.2, the whole generated population is evaluated (line 2). The specific tour to be performed on the population is determined in line 4 of the algorithm. For this research, a lineal tour was performed, from the upper-left corner to the opposite lower-right corner of the grid. For each individual in the tour a neighborhood is specified (line 6), from which the ancestors for recombination are selected (line 7). The resulting offspring are generated, mutated, and then evaluated in order to set the next generation (lines 8, 9 and 10). A buffer will store the best fitted individuals, and the whole process will run while the stop condition is not met (line 3).

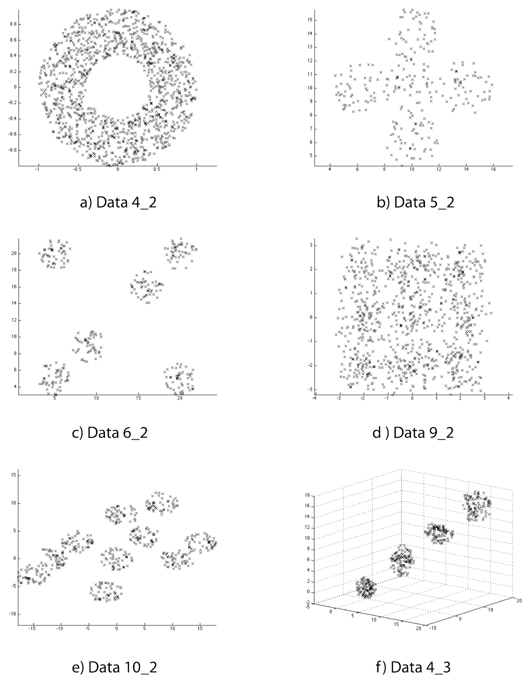

Fig. 7. Synthetic datasets used for experimentation

Table 1. Dataset characteristics

Dataset	Instances	Attributes	Classes
4_2	1288	2	4
5_2	250	2	5
6_2	300	2	6
9_2	900	2	9
10_2	500	2	10

5 Experiments and Results

To show the performance of the proposed algorithm, five synthetic datasets were selected to test the algorithm. Each dataset was tested with three different fitness functions and five different evolutionary algorithms. The synthetic datasets (see Fig. 7) were taken from the repository on the personal web page of S. Bandyopadhyay, which is available at http://www.isical.ac.in/sanghami/data.html. Those five datasets were designed to show different configurations for cluster geometry, compactness and separation. Table 1 shows the characteristics of each dataset.

Table 2. Configuration parameters for experiments

Parameter	Value
cGA grid size	5×5
cGA generations	50
cGA recombination probability	0.8
cGA mutation probability	0.01
cGA neighborhood tour	lineal
cGA neighborhood type	compact, lineal
cGA neighborhood size	1,2,5
cGA parent selection	center, binary tournament
PSO inertia	[0.1 … 0.9]
PSO cognitive weight	0.8
PSO Social weight	1.6
PSO popualtion size	5
PSO generations	30
PSO velocity	[-0.1 … 0.1]
PSO survivors	2
PSO reset generations	5

5.1 Configuration Parameters

All experiments were performed on a 16 PC cluster with 64 bits Ubuntu Linux v12.10 operating system, an Intel(R) Core(TM)2 Quad processor running at 2.66 GHz and with 4 GB of RAM. For each one of the algorithms, one hundred independent runs were performed. Three cluster validation indexes and 5 datasets were used amounting to a total of 7500 experiments; all algorithms were programmed with Matlab 2012b for Unix.

The parameterization of the studied algorithms is described in Table 2. Specifically, we study two cGAs: the cGACenter+BT and cGABT+BT. The two cGAs differ from each other in the selection method for the first parent: in cGACenter+BT the first parent is the current individual itself, while in the case of cGABT+BT the first parent is selected by binary tournament, exactly as the other parent. The algorithms were tested using different neighborhoods and sizes: *C9* represents a compact topology with nine neighbors while L5 represents a linear topology with five neighbors.

As a first important result, we show that the proposed algorithm, like its predecessors, is also capable of achieving 100% accuracy in all reported experiments. Table 3 presents the summary of the results obtained by running the proposed algorithm with each of the indexes used as fitness functions, to find the optimal number of groups in each dataset. As it can be seen, the best results were obtained using the I-index.

Table 3. Number of clusters found with the proposed algorithm for each experiment

Data Set	No. Clusters	I-Index	Xie-Beni	DB
4_2	4	**4**	4	4
5_2	5	**5**	5	5
6_2	6	**6**	4	4
9_2	9	**9**	9	4
10_2	10	**10**	10	8

5.2 Selection of Type of Parents

As discussed before, two parental selection methods were used by the cGA we experimented with: Center (C) and binary tournament (BT). Those algorithms labeled as cGACenter+BT select the first parent from the center of a neighborhood, and the second parent is selected using BT on the rest of the neighborhood. The inclusion of the central individual in each neighborhood ensures that all individuals will be part of the recombination process. On the other hand, those algorithms labeled as cGABT+BT select both parents using the BT method. As it can be seen in the experimental results provided (see Figures 8 and 9), the selection method has the overall effect of increasing or decreasing the selection pressure of the cGA. This change on the selection pressure sets different priorities to the exploration/exploitation capabilities of the algorithm causing it to converge faster or slower. The best experimental results were always achieved when both parents were selected with the BT method. This means that by increasing the selection pressure (aka. the exploration capability), the cGA is able to better traverse its search space (see statistical results table).

5.3 Type of Neighborhood

Once the selection method is decided for a particular problem, the type of neighborhood that best suits the problem must be specified. Two of the most common types of neighborhoods were tested: compact neighborhood and linear neighborhood (see Fig. 3). As Figures 10 and 11 show, the fastest convergence times for datasets 6_2 and 9_2 were achieved with linear neighborhoods. The same experiments were performed with all available datasets and the results were always the same (see statistical results table), which compels us to

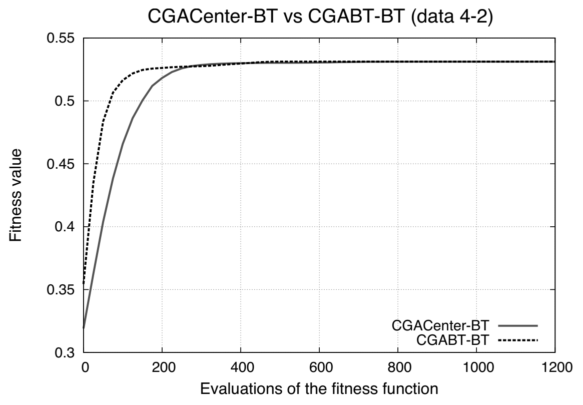

Fig. 8. Type of parents selection (dataset 4_2)

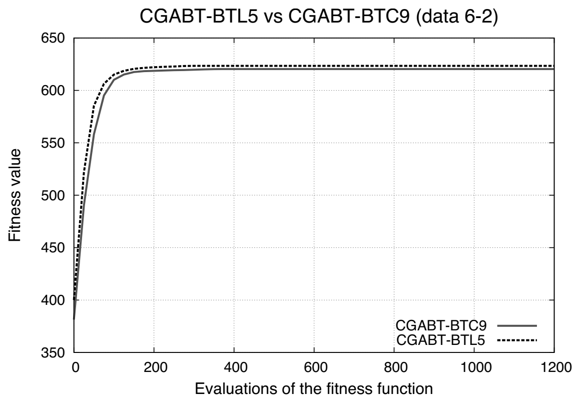

Fig. 10. Type of neighborhood (dataset 6_2)

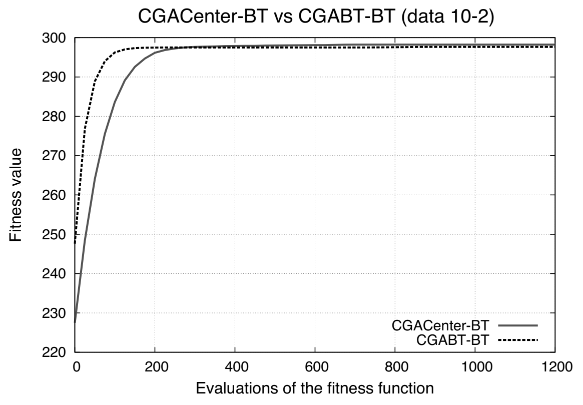

Fig. 9. Type of parents selection (dataset 10_2)

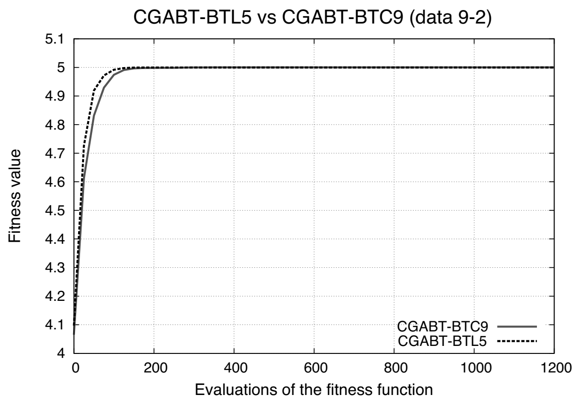

Fig. 11. Type of neighborhood (dataset 9_2)

draw the conclusion that linear neighborhoods are the ones to induce the best performance.

It can be empirically observed that, when compact neighborhoods are used for the studied problem, the selection pressure promotes exploitation over exploration, which in turn reduces the convergence speed of the algorithm. Since the probability of selecting a parent that forms part of two overlapping neighborhoods increases, the probability of selecting an exploration-promoting individual on that same search space naturally decreases.

5.4 Size of Neighborhood

Until this point, based on the experiments performed, a selection method and a neighborhood type have been picked up, that bring the proposed cGA to its best possible performance. To bluntly conclude this research, the length (radius) of the best possible neighborhood was also searched for. Since the structuring grid for all experiments has a size of 5x5 individuals, lengths of 5, 9, and 13 individuals were tested. Figures 12 and 13 show that the bigger difference resulted when comparing the performance of neighborhoods with 5 and 9 individuals as well as with 5 and 13 individuals. This means that neighborhoods of size 5 show the worse performance, including the lowest fitness of all tested algorithms. Also, the performance of algorithms with 9 and 13-sized neighborhoods are nearly the same, so the conclusion is drawn as follows.

Since the structuring grid is toroidal, an overlapping is always present between individuals from different neighborhoods, and the neighborhood of size 13 always includes the central individual from a neighborhood of size 9. Consequently, both cases promote a relaxation on pressure selection of the algorithm, and a decrease in the convergence speed.

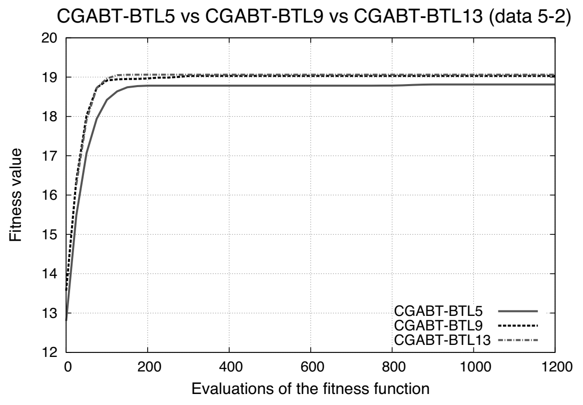

Fig. 12. Size of neighborhood (dataset 5_2)

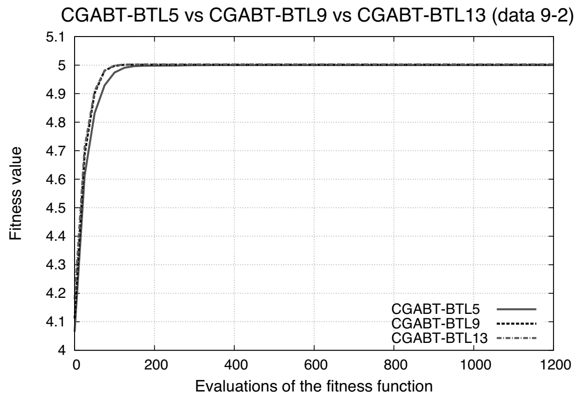

Fig. 13. Size of neighborhood (dataset 9_2)

5.5 Statistical Results

In order to be able to compare the experimental results from all tested algorithm variants and their parameters, two statistical analyses were performed. The first one involves all classical statistical indicators (maximum, mean, median, standard deviation, etc.), which allow an initial comparison. The second analysis involves non-parametric indexes based on hypothesis tests which detect and alert if there is a significant difference among performances of the tested algorithms as well as help to identify the best performing test.

Table 4 shows that the worst performance was that from the cGACenter+BTL1 algorithm, since it had the lowest fitness value in all experiments. The algorithm with the lower standard deviation was cGACenter+BTC5, undoubtedly caused by the selection pressure induced by its type of neighborhood. The best performance among all five tested algorithms was achieved by cGABT+BTL9 and cGABT+BTL13. This can also be explained by the type and size of the neighborhood those algorithms used. Therefore, from the data showed in Table 4 it can be concluded that for the currently studied problem, the best performing algorithms are those that select parents by the BT method and with neighborhoods of size between 9 and 13 individuals.

5.6 Statistical Analysis of Experimental Results

Non-parametric statistical tests have emerged as an effective, affordable, and robust way for evaluating new proposals of meta-heuristic and evolutionary algorithms [14]. The validation and comparison of new algorithms often requires the definition of a comprehensive experimental framework, including a range of problems and state of the art algorithms. A critical part of these comparisons lies in the statistical validation of the results, contrasting the differences between methods.

To analyze the obtained experimental results, two non-parametric statistical tests have been used. Particularly, the Aligned Friedman Test was applied, and also the Multi-compare Holm test was used as a post hoc procedure to find out which algorithms have worse performance than the proposed algorithm.

The Friedman Test [9] assigns a r_{ij} ranking to the results obtained by each algorithm i over each data set j. Each ranking is a real number $1 \leq r_{ij} \leq k$, where k is the total number of algorithms to compare. Rankings are assigned in an ascending way, so 1 is the best possible rank, and it gets worse as the assigned rank grows.

The null hypothesis, established for the performed post hoc study, assumes that all three compared algorithms have a statistically similar behavior. The Holm test, with $\alpha = 0.05$, aims at succeeding in 95% or more of the analyzed results, having a Gaussian (normal) distribution as its point of reference.

Holm's procedure rejects those hypotheses that have an unadjusted p-value ≤ 0.016667.

Table 4. Statistical results of all instances

Dataset	Std. Measure	CGACenter-BTL5	CGABT-BTC9	CGABT-BTL5	CGABT-BTL9	CGABT-BTL13
dataset 4_2	Min	**0.3065**	0.3744	0.3491	0.3775	0.3806
	Max	0.4782	0.5236	0.5167	**0.5294**	0.5280
	Mean	0.4032	0.4779	0.4643	**0.4846**	0.4823
	Median	0.4190	0.5035	0.4952	**0.5155**	0.5124
	Std	0.0703	**0.0612**	0.0693	0.0638	0.0622
dataset 5_2	Min	**11.0240**	13.3710	12.6400	13.7200	13.2270
	Max	16.8240	18.8230	18.5080	18.9530	**18.9970**
	Mean	14.2902	17.2230	16.5132	**17.3764**	17.3008
	Median	14.6990	18.1450	17.4400	**18.5630**	18.5230
	Std	2.3241	2.2519	2.3847	**2.2277**	2.4392
dataset 6_2	Min	**304.3400**	395.4500	375.5600	400.0800	393.1600
	Max	554.6600	613.9900	611.8800	**619.8500**	619.0700
	Mean	440.6040	550.9820	534.7240	**558.0380**	557.2960
	Median	449.2300	593.4200	582.0900	604.3500	**612.2000**
	Std	101.0624	**90.9416**	98.7067	92.7730	97.0172
dataset 9_2	Min	**3.7629**	4.2465	4.2168	4.2525	4.3572
	Max	4.8965	4.9932	4.9823	5.0017	**5.0026**
	Mean	4.4627	4.7928	4.7415	4.8018	**4.8267**
	Median	4.5980	4.9452	4.8690	4.9681	**4.9705**
	Std	0.4548	0.3143	0.3147	0.3197	**0.2745**
dataset 10_2	Min	**234.2100**	257.9200	254.1300	258.9300	255.7700
	Max	289.1000	299.1100	296.5500	299.5700	**299.7000**
	Mean	266.2940	287.8440	283.6300	**288.8460**	287.7800
	Median	270.5600	295.5000	291.7100	**297.8500**	297.7400
	Std	21.8989	**17.2526**	17.7088	17.3374	18.7831

Table 5. Post Hoc comparison for $\alpha = 0.05$

i	algorithm	$z = (R_0 - R_i)/SE$	p	Holm
1	CGABT-BTL13	0.042967	0.965728	0.050000
2	CGABT-BTC9	0.386702	0.698977	**0.025000**
3	CGABT-BTL5	1.203073	0.228948	**0.016667**
4	CGACenter-BTL5	3.093616	0.001977	**0.012500**

Table 6. Average rankings of the algorithms (Aligned Friedman Test)

Algorithm	Ranking
CGABT-BTL9	**8.6**
CGABT-BTL13	8.8
CGABT-BTC9	10.4
CGABT-BTL5	14.2
CGACenter-BTL5	23

6 Conclusions

A new Hybrid Cellular Genetic Algorithm (HcGA) was proposed for finding the optimal number of clusters in a dataset. Several selection methods, as well as types and lengths of neighborhoods were tested for the proposed algorithm. The best experimental results were achieved when using BT as the parent selection method, and a linear neighborhood of size 9. The choice of the selection method has a direct effect over the algorithm's selection pressure, which in turn causes the convergence speed to significantly improve when both parents are selected with the BT method. Also, the effect induced by the choice of different neighborhoods was also researched, and the best performance was achieved by linear neighborhoods where the probability of selecting the same two parents for two overlapped neighborhoods decreases. Finally, statistical analysis show that selecting between 9 and 13 individuals to form neighborhoods also leads to the best possible performance. The non-parametric statistical analysis confirms that cGABT+BTL9 behaves in a very similar way to cGABT+BTL13, while cGABT+BTC5 performs better in 97.5% of experiments, and even in 99% of experiments when compared with cGABT+BTL5 and cGACenter+BTL5.

7 Future Work

All experiments were performed with synthetic datasets, so testing the proposed algorithm with real datasets is first thing on the list. Also, an exhaustive comparative study with synchronous and asynchronous cGAs using diverse routes will be sought.

Finally, a parallel implementation of all tested algorithms (over GPUs or multi-core CPUs) will be useful for statistically assessing their viability.

Acknowledgements

The authors would like to express their gratitude to SIP-IPN, CONACyT and ICyT-DF for their economic support of this research, particularly, through grants SIP-20130932 and ICyT-PICCO-10-113.

References

1. **Bandyopadhyay, S. & Maulik, U.** (**2002**). An evolutionary technique based on k-means algorithm for optimal clustering in rn. *Information Sciences*, 146(1), 221–237.
2. **Bandyopadhyay, S. & Maulik, U.** (**2002**). Genetic clustering for automatic evolution of clusters and application to image classification. *Pattern Recognition*, 35(6), 1197–1208.
3. **Bellis, M. A., Jarman, I., Downing, J., Perkins, C., Beynon, C., Hughes, K., & Lisboa, P.** (**2012**). Using clustering techniques to identify localities with multiple health and social needs. *Health & place*, 18(2), 138–143.
4. **Cabrera, J. C. F. & Coello, C. A. C.** (**2007**). Handling constraints in particle swarm optimization using a small population size. In *MICAI 2007: Advances in Artificial Intelligence*. Springer, 41–51.
5. **Cabrera, J. C. F. & Coello, C. A. C.** (**2010**). Micro-mopso: a multi-objective particle swarm optimizer that uses a very small population size. In *Multi-Objective Swarm Intelligent Systems*. Springer, 83–104.
6. **Cao, J., Wu, Z., Wu, J., & Liu, W.** (**2012**). Towards information-theoretic k-means clustering for image indexing. *Signal Processing*, 39(2), 1–12.
7. **Chang, L., Duarte, M. M., Sucar, L., & Morales, E. F.** (**2012**). A bayesian approach for object classification based on clusters of sift local features. *Expert Systems With Applications*, 39(2), 1679–1686.
8. **Correa-Morris, J., Espinosa-Isidron, D. L., & Alvarez-Nadiozhin, D. R.** (**2010**). An incremental nested partition method for data clustering. *Pattern Recognition*, 43(7), 2439–2455.
9. **Cortina-Borja, M.** (**2012**). Handbook of parametric and nonparametric statistical procedures. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 175(3), 829–829.
10. **Das, S., Abraham, A., & Konar, A.** (**2008**). Automatic clustering using an improved differential evolution algorithm. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 38(1), 218–237.
11. **Davies David L. Bouldin, D. W.** (**1979**). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2, 224–227.
12. **Deb, K., Agrawal, S., Pratap, A., & Meyarivan, T.** (**2000**). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. *Lecture notes in computer science*, 1917, 849–858.
13. **Franek, L., Abdala, D., Vega-Pons, S., & Jiang, X.** (**2011**). Image segmentation fusion using general ensemble clustering methods. *Computer Vision–ACCV 2010*, 373–384.
14. **Garcia, S., Molina, D., Lozano, M., & Herrera, F.** (**2009**). A study on the use of non-parametric tests for analyzing the evolutionary algorithms? behaviour: a case study on the cec 2005 special session on real parameter optimization. *Journal of Heuristics*, 15(6), 617–644.
15. **Goldberg, D. E.** (**1989**). Sizing populations for serial and parallel genetic algorithms. *Proceedings of the 3rd International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers Inc., pp. 70–79.
16. **Grosan, C., Abraham, A., & Ishibuchi, H.** (**2007**). *Hybrid evolutionary algorithms*. Springer Publishing Company, Incorporated.
17. **Hartigan, J. A. & Wong, M. A.** (**1979**). Algorithm as 136: A k-means clustering algorithm. *Applied statistics*, 100–108.
18. **Hong, Y., Kwong, S., Chang, Y., & Ren, Q.** (**2008**). Unsupervised feature selection using clustering ensembles and population based incremental learning algorithm. *Pattern Recognition*, 41(9), 2742–2756.
19. **Jain, A. K., Murty, M. N., & Flynn, P. J.** (**1999**). Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3), 264–323.
20. **Jarboui, B., Cheikh, M., Siarry, P., & Rebai, A.** (**2007**). Combinatorial particle swarm optimization (cpso) for partitional clustering problem. *Applied Mathematics and Computation*, 192(2), 337–345.
21. **Kanade, P. M. & Hall, L. O.** (**2003**). Fuzzy ants as a clustering concept. *22nd International Conference of the North American Fuzzy Information Processing Society (NAFIPS 2003)*, IEEE, pp. 227–232.

22. Kennedy, J. & Eberhart, R. (**1995**). Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks*, volume 4, IEEE, pp. 1942–1948.

23. Kodratoff, Y. & Michalski, R. S. (**1990**). *Machine learning: an artificial intelligence approach*, volume 3. Morgan Kaufmann Publishers.

24. Krishnakumar, K. (**1989**). Micro-genetic algorithms for stationary and non-stationary function optimization. *Advances in Intelligent Robotics Systems Conference*, International Society for Optics and Photonics, pp. 289–296.

25. Kwedlo, W. (**2011**). A clustering method combining differential evolution with the k-means algorithm. *Pattern Recognition Letters*, 32(12), 1613–1621.

26. Lau, R. Y., Li, Y., Song, D., & Kwok, R. C. W. (**2008**). Knowledge discovery for adaptive negotiation agents in e-marketplaces. *Decision Support Systems*, 45(2), 310–323.

27. Lopez-Ortega, O. & Rosales, M.-A. (**2011**). An agent-oriented decision support system combining fuzzy clustering and the ahp. *Expert Systems with Applications*, 38(7), 8275–8284.

28. Lu, Y., Lu, S., Fotouhi, F., Deng, Y., & Brown, S. J. (**2004**). Fgka: a fast genetic k-means clustering algorithm. *Proceedings of the 2004 ACM symposium on Applied computing*, ACM, pp. 622–623.

29. Martínez-Álvarez, F., Troncoso, A., Riquelme, J., & Aguilar-Ruiz, J. (**2011**). Energy time series forecasting based on pattern sequence similarity. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, pp. 1230–1243 vol.23 No. 8.

30. Martinez-Trinidad, J. F. & Guzman-Arenas, A. (**2001**). The logical combinatorial approach to pattern recognition, an overview through selected works. *Pattern Recognition*, 34(4), 741–751.

31. Maulik, U. & Bandyopadhyay, S. (**2002**). Performance evaluation of some clustering algorithms and validity indices. *IEEE T. Pattern*, 24(12), 1650–1654.

32. Niknam, T., Firouzi, B. B., & Nayeripour, M. (**2008**). An efficient hybrid evolutionary algorithm for cluster analysis. *World Applied Sciences Journal*, 4(2), 300–307.

33. Omran, M., Engelbrecht, A. P., & Salman, A. (**2005**). Particle swarm optimization method for image clustering. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(03), 297–321.

34. Parsopoulos, K. E. (**2009**). Cooperative micro-differential evolution for high-dimensional problems. *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, ACM, pp. 531–538.

35. Rousseeuw, P. J. & Kaufman, L. (**1990**). Finding groups in data: An introduction to cluster analysis. *John, John Wiley & Sons*.

36. Saha, I., Maulik, U., & Bandyopadhyay, S. (**2009**). A new differential evolution based fuzzy clustering for automatic cluster evolution. *IEEE International Advance Computing Conference (IACC 2009)*, 706–711.

37. Vega-Pons, S., Ruiz-Shulcloper, J., & Guerra-Gandon, A. (**2011**). Weighted association based methods for the combination of heterogeneous partitions. *Pattern Recognition Letters*, 32(16), 2163–2170.

38. Villa, A., Chanussot, J., Benediktsson, J. A., Jutten, C., & Dambreville, R. (**2012**). Unsupervised methods for the classification of hyperspectral images with low spatial resolution. *Pattern Recognition*.

39. Viveros-Jiménez, F., Mezura-Montes, E., & Gelbukh, A. (**2009**). Elitistic evolution: a novel micro-population approach for global optimization problems. *Eighth Mexican International Conference on Artificial Intelligence (MICAI 2009)*, IEEE, pp. 15–20.

40. Viveros Jiménez, F., Mezura Montes, E., & Gelbukh, A. (**2012**). Empirical analysis of a micro-evolutionary algorithm for numerical optimization. *Int. J. Phys. Sci*, 7, 1235–1258.

41. Wang, X., Yang, C., & Zhou, J. (**2009**). Clustering aggregation by probability accumulation. *Pattern Recognition*, 42(5), 668–675.

42. Wei-Ping Lee, S.-W. C. (**2010**). Automatic clustering with differential evolution using a cluster number oscillation method. *Intelligent Systems and Applications*, 218–237.

43. Xie, X. L. & Beni, G. (**1991**). A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and machine Intelligence*, 13(4).

44. Xu, R., Wunsch, D., et al. (**2005**). Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3), 645–678.

45. Yan, H., Chen, K., Liu, L., & Yi, Z. (**2010**). Scale: a scalable framework for efficiently clustering transactional data. *Data mining and knowledge Discovery*, 20(1), 1–27.

46. Yang, Y., Liao, Y., Meng, G., & Lee, J. (**2011**). A hybrid feature selection scheme for unsupervised learning and its application in bearing fault diagnosis. *Expert Systems With Applications*, 38(9), 1311–1320.

Javier Arellano-Verdejo recieved his Master's degree in Science from Tecnológico de Estudios Superiores de Ecatepec, Mexico, in 2008. He is currently pursing the Ph.D. degree in Computer Science at the Computing Research Center, National Polytechnic Institute, Mexico City. His current research interests include parallel bioinspired heuristics, estimation of distribution algorithms and pattern recognition.

Adolfo Guzmán-Arenas is a Computer Science professor at the Computing Research Center, National Polytechnic Institute (CIC-IPN), Mexico City, of which he was Founding Director. He holds a B.Sc. in Electronics from ESIME-IPN, and a Ph.D. from MIT. He is an ACM Fellow, IEEE Life Fellow Member, member of MIT Educational Council, member of the Engineering Academy and the National Academy of Sciences (Mexico). From the President of Mexico, he received the National Prize in Science and Technology (1996) and the Jaime Torres Bodet National Award for Excellence (2006). He works in semantic information processing and AI techniques, often with distributed information systems. More at http://alum.mit.edu/www/aguzman.

Salvador Godoy-Calderon holds a Ph.D. in Computer Science from CIC-IPN and a Master's degree in C.S. from CINVESTAV-IPN. He is a full time researcher and Head of the Artificial Intelligence Laboratory at CIC-IPN with research interests including pattern recognition, evolutionary computation, strategic and tactical knowledge systems, and formal logics.

Ricardo Barrón Fernández holds a Ph.D. in Computer Science from CIC-IPN and a Master's degree in C.E. from CIC-IPN. He is a full time researcher at the Artificial Intelligence Laboratory of CIC-IPN with research interests including pattern recognition, GPU computing, evolutionary computation, parallel systems.

Article received on 07/10/2013, accepted on 15/12/2013.