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

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, microevolutionary algorithms, particle swarm optimization, optimal number of clusters.

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 \(\Omega = \{o_1, \ldots, o_n\}\) be a set of \(r\) dimensional patterns. A clustering of \(\Omega\) is a partitioning of \(\Omega\) into \(k\) clusters \(\{C_1, \ldots, C_k\}\) 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: \(C_i \cap C_j = \emptyset\), 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, Data 5, Data 6, and Data 4) 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
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).

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 neighborhood: the $C_n$ compact form and the $L_n$ 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).

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

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

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

$$Dist(C_i) = \left( \frac{1}{|C_i|} \sum_{o \in C_i} \|o - \mu_i\|^2 \right)^{\frac{1}{2}},$$

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 $j$:

$$XB = \frac{\sum_{j=1}^{k} \sum_{i=1}^{n} \mu_{ij} \|o_i - u_j\|^2}{n \min_{p \neq j} \left\{ \|u_j - u_p\|^2 \right\}^{\frac{1}{2}}}.$$

where $\mu_{ij}$ 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 $p = 2$:
\[ I = \left( \frac{1}{k} 
abla E_1 \cdot \nabla E_k \cdot D_k \right)^p, \tag{4} \]

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 \( \mu \)PSOLS (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 \( \mu \)PSO, with local search capabilities (LS), is used to determine the optimal clustering of objects for the individual’s value of \( k \).

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 \( \mu \)PSO 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 \( \mu \)PSO 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.
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).

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/sangham/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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cGA grid size</td>
<td>5 × 5</td>
</tr>
<tr>
<td>cGA generations</td>
<td>50</td>
</tr>
<tr>
<td>cGA recombination probability</td>
<td>0.8</td>
</tr>
<tr>
<td>cGA mutation probability</td>
<td>0.01</td>
</tr>
<tr>
<td>cGA neighborhood tour</td>
<td>lineal</td>
</tr>
<tr>
<td>cGA neighborhood type</td>
<td>compact, lineal</td>
</tr>
<tr>
<td>cGA neighborhood size</td>
<td>1, 2, 5</td>
</tr>
<tr>
<td>cGA parent selection</td>
<td>center, binary tournament</td>
</tr>
<tr>
<td>PSO inertia</td>
<td>[0.1 ... 0.9]</td>
</tr>
<tr>
<td>PSO cognitive weight</td>
<td>0.8</td>
</tr>
<tr>
<td>PSO Social weight</td>
<td>1.6</td>
</tr>
<tr>
<td>PSO population size</td>
<td>5</td>
</tr>
<tr>
<td>PSO generations</td>
<td>30</td>
</tr>
<tr>
<td>PSO velocity</td>
<td>[-0.1 ... 0.1]</td>
</tr>
<tr>
<td>PSO survivors</td>
<td>2</td>
</tr>
<tr>
<td>PSO reset generations</td>
<td>5</td>
</tr>
</tbody>
</table>

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 cGA Center+BT and cGABT+BT. The two cGAs differ from each other in the selection method for the first parent: in cGA Center+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

<table>
<thead>
<tr>
<th>Data Set</th>
<th>No. Clusters</th>
<th>I-Index</th>
<th>Xie-Beni</th>
<th>DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>8</td>
</tr>
</tbody>
</table>

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 cGA Center+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
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)](image)

![Fig. 13. Size of neighborhood (dataset 9,2)](image)

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+BTL5, undoubtedly caused by the selection pressure induced by its type of neighborhood. The best performance among all five tested algorithms was achieved by cGACenter+BTL9 and cGACenter+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 \( \leq 0.016667 \).
Table 4. Statistical results of all instances

<table>
<thead>
<tr>
<th>Dataset 4.2</th>
<th>Std. Measure</th>
<th>CGACenter-BTLS</th>
<th>CGABT-BTC9</th>
<th>CGABT-BTLS</th>
<th>CGABT-BTL9</th>
<th>CGABT-BTL13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>0.3065</td>
<td>0.3744</td>
<td>0.3491</td>
<td>0.3775</td>
<td>0.3806</td>
<td></td>
</tr>
<tr>
<td>Max</td>
<td>0.4782</td>
<td>0.5236</td>
<td>0.5167</td>
<td>0.5294</td>
<td>0.5280</td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>0.4003</td>
<td>0.4779</td>
<td>0.4643</td>
<td>0.4666</td>
<td>0.4623</td>
<td></td>
</tr>
<tr>
<td>Median</td>
<td>0.4190</td>
<td>0.5035</td>
<td>0.4952</td>
<td>0.5155</td>
<td>0.5124</td>
<td></td>
</tr>
<tr>
<td>Std</td>
<td>0.0703</td>
<td>0.0612</td>
<td>0.0693</td>
<td>0.0638</td>
<td>0.0622</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset 5.2</th>
<th>Std. Measure</th>
<th>CGACenter-BTLS</th>
<th>CGABT-BTC9</th>
<th>CGABT-BTLS</th>
<th>CGABT-BTL9</th>
<th>CGABT-BTL13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
<td>16.8240</td>
<td>18.8230</td>
<td>18.5080</td>
<td>18.9530</td>
<td>18.9970</td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>14.2902</td>
<td>17.2230</td>
<td>16.5132</td>
<td>17.3764</td>
<td>17.3008</td>
<td></td>
</tr>
<tr>
<td>Median</td>
<td>14.6990</td>
<td>18.1450</td>
<td>17.4400</td>
<td>18.3630</td>
<td>18.5630</td>
<td></td>
</tr>
<tr>
<td>Std</td>
<td>2.3241</td>
<td>2.2519</td>
<td>2.3647</td>
<td>2.2277</td>
<td>2.4392</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset 6.2</th>
<th>Std. Measure</th>
<th>CGACenter-BTLS</th>
<th>CGABT-BTC9</th>
<th>CGABT-BTLS</th>
<th>CGABT-BTL9</th>
<th>CGABT-BTL13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>304.2400</td>
<td>395.4500</td>
<td>375.5600</td>
<td>400.0800</td>
<td>393.1600</td>
<td></td>
</tr>
<tr>
<td>Max</td>
<td>554.6600</td>
<td>613.9900</td>
<td>592.1400</td>
<td>604.8200</td>
<td>604.3800</td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>440.6040</td>
<td>520.9820</td>
<td>484.7240</td>
<td>508.0380</td>
<td>507.2960</td>
<td></td>
</tr>
<tr>
<td>Median</td>
<td>449.2300</td>
<td>593.4200</td>
<td>582.0900</td>
<td>604.3500</td>
<td>612.2000</td>
<td></td>
</tr>
<tr>
<td>Std</td>
<td>101.0624</td>
<td>90.3410</td>
<td>98.7087</td>
<td>95.7730</td>
<td>97.0172</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset 9.2</th>
<th>Std. Measure</th>
<th>CGACenter-BTLS</th>
<th>CGABT-BTC9</th>
<th>CGABT-BTLS</th>
<th>CGABT-BTL9</th>
<th>CGABT-BTL13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>3.7629</td>
<td>4.2465</td>
<td>4.2168</td>
<td>4.2525</td>
<td>4.3572</td>
<td></td>
</tr>
<tr>
<td>Max</td>
<td>4.8965</td>
<td>4.9932</td>
<td>4.9823</td>
<td>5.0117</td>
<td>5.0026</td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>4.8627</td>
<td>4.9292</td>
<td>4.7415</td>
<td>4.8518</td>
<td>4.8267</td>
<td></td>
</tr>
<tr>
<td>Median</td>
<td>4.9589</td>
<td>4.9452</td>
<td>4.9381</td>
<td>4.9501</td>
<td>4.9705</td>
<td></td>
</tr>
<tr>
<td>Std</td>
<td>0.4548</td>
<td>0.3143</td>
<td>0.3147</td>
<td>0.3197</td>
<td>0.2745</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset 10.2</th>
<th>Std. Measure</th>
<th>CGACenter-BTLS</th>
<th>CGABT-BTC9</th>
<th>CGABT-BTLS</th>
<th>CGABT-BTL9</th>
<th>CGABT-BTL13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>234.2100</td>
<td>257.9200</td>
<td>254.1300</td>
<td>258.9300</td>
<td>255.7700</td>
<td></td>
</tr>
<tr>
<td>Max</td>
<td>289.1000</td>
<td>299.1100</td>
<td>296.5500</td>
<td>299.5700</td>
<td>299.7000</td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>266.2940</td>
<td>287.8440</td>
<td>283.6300</td>
<td>288.8460</td>
<td>287.7800</td>
<td></td>
</tr>
<tr>
<td>Median</td>
<td>270.5600</td>
<td>295.5000</td>
<td>291.7100</td>
<td>297.8500</td>
<td>297.7400</td>
<td></td>
</tr>
<tr>
<td>Std</td>
<td>21.8989</td>
<td>27.2526</td>
<td>17.7088</td>
<td>17.3374</td>
<td>18.7631</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>i</th>
<th>Algorithm</th>
<th>$z = (R_0 - R_i)/SE$</th>
<th>$p$ Holm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CGABT-BTLS</td>
<td>0.042967</td>
<td>0.965728</td>
</tr>
<tr>
<td>2</td>
<td>CGABT-BTC9</td>
<td>0.386702</td>
<td>0.698977</td>
</tr>
<tr>
<td>3</td>
<td>CGABT-BTL5</td>
<td>1.203073</td>
<td>0.228948</td>
</tr>
<tr>
<td>4</td>
<td>CGACenter-BTLS</td>
<td>3.095616</td>
<td>0.001977</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGABT-BTLS</td>
<td>8.6</td>
</tr>
<tr>
<td>CGABT-BTL13</td>
<td>8.8</td>
</tr>
<tr>
<td>CGABT-BTC9</td>
<td>10.4</td>
</tr>
<tr>
<td>CGABT-BTL5</td>
<td>14.2</td>
</tr>
<tr>
<td>CGACenter-BTLS</td>
<td>23</td>
</tr>
</tbody>
</table>

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 analysis is needed to determine the effectiveness of the algorithm in real-world applications.
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


Javier Arellano-Verdejo received his Master’s degree in Science from Tecnológico de Estudios Superiores de Ecatepec, Mexico, in 2008. He is currently pursuing the Ph.D. degree in Computer Science at the Computing Research Center, National Polytechnic Institute, Mexico City. His current research interests include parallel bio-inspired 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.