HYBRID ADAPTIVE DIFFERENTIAL EVOLUTION IN PARTITIONAL CLUSTERING

Josef Tvrdík and Ivan Krivy

University of Ostrava
Department of Computer Science
30. dubna 22, 701 03 Ostrava
Czech Republic
josef.tvrdik@osu.cz, ivan.krivy@osu.cz

Abstract: The problem of optimal partitioning by minimizing pooled-within-variance of groups is addressed. Three state-of-the-art adaptive differential evolution algorithms are compared on four real-world data sets. A novel hybrid differential evolution algorithm, including k-means algorithm for local search is proposed. The experimental comparison is done with either the plain adaptive differential evolution variants or the hybrid algorithms. Experimental results showed that hybrid algorithms are substantially better performing when compared with plain differential evolution variants. Among hybrid variants, the competitive differential evolution appeared to be the most efficient.

Keywords: Optimal partitioning; trace-within minimizing criterion; adaptive differential evolution; k-means algorithm; hybrid search.

1 Introduction

Cluster analysis is an important exploratory technique used for partitioning objects data sets into relatively homogeneous groups of similar objects. Each group, called a cluster, contains objects that are similar among themselves and dissimilar to objects of other groups. There is a number of data partitioning algorithms [8], which can be subdivided into hierarchical and partitional ones. In hierarchical algorithms, the output is a tree showing a sequence of clustering with each cluster being a partition of the data set. On the other hand, partitional clustering algorithms try to decompose the data set directly into a set of disjoint clusters using available optimizing criteria [20, 25]. Typically, the criteria include minimizing some measure of dissimilarity among objects within each cluster and, at the same time, maximizing the dissimilarity of different clusters.

The partitioning problem is described as follows. Let \( O \) be a set of \( n \) objects, each of which is characterized by \( p \) real-valued attributes. Furthermore, let \( Z \) be a data matrix of size \( n \times p \). Therefore, the matrix can be considered as composed of \( n \) data vectors \( z_i \), where each element \( z_{ij} \) represents the \( j \)th real-valued attribute of the \( i \)th object. Given the matrix \( Z \), the aim of the partitional clustering algorithm is to find such a partition \( G = \{C_1, C_2, \ldots, C_g\} \), \( C_k \neq \emptyset \) for all \( k \), \( C_k \cap C_l = \emptyset \) for all \( k \neq l \), \( \cup_{k=1}^{g} C_k = O \), \( k, l = 1, 2, \ldots, g \), that the objects belonging to the same cluster are as similar to each other as possible, while the objects belonging to different clusters are as dissimilar as possible. Thus, it is necessary to find a criterion for determining the degree of optimality of a given partition.

2 Criteria of Optimal Partitioning

Several criteria convenient for comparing the degree of optimality over all possible partitions are presented in [10]. In this paper, we use optimal classification by minimizing trace within criterion (hereafter \( TRW \)) proposed by Friedman and Rubin [6]:

\[
TRW = \text{trace}(W),
\]

(1)

where \( W \) is the pooled-within groups scatter matrix \((W)\) defined as

\[
W = \sum_{k=1}^{g} W_k,
\]

(2)

\( W_k \) being the variance matrix of attributes for the objects belonging to cluster \( C_k \),

\[
W_k = \sum_{j=1}^{n_k} (z_j^{(k)} - \bar{z}^{(k)})(z_j^{(k)} - \bar{z}^{(k)})^T,
\]

(3)
between 0 and 1. The mutation vector \( v \) not coinciding with the current \( x \) means (centroids) for cluster \( C_k \), and \( n_k \) the number of objects in cluster \( C_k \).

We search an optimal partition by minimizing the trace of matrix \( W \) over all feasible partitions. This criterion assumes a low correlation among observations and tends to create spherical clusters.

## 3 Differential Evolution Algorithm

The differential evolution (DE) introduced by Storn and Price [19] was proposed for dealing with continuous optimization problems. The objective function to be minimized is \( f(x) \), \( x = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d \), and the feasible domain \( D \) is defined by specifying boundary constraints, which are lower \((a_j)\) and upper \((b_j)\) limits of each component \( j \), \( D = \prod_{j=1}^{d} [a_j, b_j] \), \( a_j < b_j \), \( j = 1, 2, \ldots, d \). The global minimum point \( x^* \) satisfying condition \( f(x^*) \leq f(x) \) for \( \forall x \in D \) is the solution of the problem.

DE algorithm has become one of the most frequently evolutionary algorithms used for solving the continuous global optimization problems in recent years [14]. The results of research in DE are summarized by Neri and Tirronen [12] and by Das and Suganthan [4]. Like other evolutionary algorithms, DE works with a population of individuals \((NP\) points in the feasible domain \( D)\), that are considered as candidates of solution. The population is developing iteratively during the process by using evolutionary operators of selection, mutation, and crossover. The basic scheme of DE is written in a pseudo-code in Algorithm 1.

![Figure 1: Differential evolution](image)

The trial vector \( y \) is generated by crossover of two parent vectors, the current (target) vector \( x \), and a mutant vector \( v \). The mutant vector \( v \) is obtained by a mutation. During last years more than ten kinds of mutation have been proposed and tested. Here we mention four kinds of mutation used in algorithms compared in this study. Suppose that \( r_1, r_2, r_3, r_4 \), and \( r_5 \) are five mutually distinct points taken randomly from population \( P \), not coinciding with the current \( x_i \), \( F > 0 \) control parameter, and \( rand \) uniformly distributed random number between 0 and 1. The mutation vector \( v \) can be generated as follows:

- **rand/1/**
  \[
  v = r_1 + F (r_2 - r_3),
  \]

- **rand/2/**
  \[
  v = r_1 + F (r_2 - r_3) + F (r_4 - r_5),
  \]

- **randr/1/**
  \[
  v = r_1 + F (r_2 - r_3),
  \]

where the point \( r_1 \) is not chosen randomly like in rand/1/, but tournament best among \( r_1, r_2, \) and \( r_3 \), \( r_1 = \arg \min_{i \in \{1,2,3\}} f(r_i) \), as proposed in [9],

- **current-to-rand/1/**
  \[
  y = x_i + rand (r_1 - x_i) + F (r_2 - r_3).
  \]

Notice, that the current-to-rand/1/ mutation generates a trial point \( y \) directly, because (7) includes so called arithmetic crossover.
The crossover operator constructs the trial vector \( y \) from current individual \( x_i \) and the mutant vector \( v \). Two types of crossover were proposed in \([19]\). Binomial crossover replaces the elements of vector \( x_i \), using the following rule

\[
y_j = \begin{cases} 
v_j & \text{if } U_j \leq CR \\
x_{ij} & \text{if } U_j > CR \end{cases} \quad \text{or } j = l,
\]

where \( l \) is a randomly chosen integer from \( \{1, 2, \ldots, d\} \), and \( U_1, U_2, \ldots, U_d \) are independent random variables uniformly distributed in \([0, 1]\). \( CR \in [0, 1] \) is a control parameter influencing the number of elements to be exchanged by crossover. Eq. (8) ensures that at least one element of \( x_i \) is changed, even if \( CR = 0 \). The variant of DE using mutation (4) and binomial crossover, in abbreviation DE/rand/1/bin, is the most frequently used DE strategy in applications.

For exponential crossover (DE/\:-\:/exp), the starting position of crossover is chosen randomly from \( 1, \ldots, d \), and \( L \) consecutive elements (counted in circular manner) are taken from the mutant vector \( v \). Probability of replacing the \( k \)th element in the sequence \( 1, 2, \ldots, L \), \( L \leq d \), decreases exponentially with increasing \( k \). \( L \) adjacent elements are changed in exponential variant, in binomial one the changed coordinates are dispersed randomly over the dimensions \( 1, 2, \ldots, d \). While in binomial crossover the relation between the probability of mutation and the \( CR \) is linear, in the exponential crossover this relation is nonlinear and the deviation from linearity enlarges with increasing dimension of problem. Probability of mutation (\( p_m \)) controls the number of exchanged elements in crossover, \( p_m \times d \) is the mean value of mutant elements used in producing offsprings. Zaharie \([26, 27]\) derived the relation between \( p_m \) and \( CR \) for exponential crossover. Her result can be rewritten in the form of polynomial equation

\[
CR^d - d p_m CR + d p_m - 1 = 0.
\]

The value of \( CR \) for given value of \( p_m \in (1/d, 1) \) can be evaluated as the root of the equation (9).

Comparing with other evolutionary algorithms (EAs), the differential evolution has a very few control parameters. Except the size of population \( NP \) common for all EAs it is the choice of mutation and crossover strategy, and pair of parameters \( F \) and \( CR \), controlling the mutation and crossover, respectively. However, the efficiency of differential evolution is very sensitive mainly to the control parameter setting of \( F \) and \( CR \) values. Based on wide experimental results, there are some recommendations for the setting of these parameters, see \([5, 7, 14, 18, 19]\), but such recommendations are not valid over all the problem to be solved. The suitable control parameters values for a specific problem may be found by trial-and-error tuning, but it requires a lot of time. That is why many self-adaptive DE variants have been proposed. Nowadays, state-of-the-art DE variants are \( jDE \) [2], \( JADE \) [28], \( SaDE \) [16], and \( EPSDE \) [11] that are used in comparison with a newly proposed DE variant.

4 DE Variants for Experimental Comparison

From the state-of-the-art DE variants, the \( jDE \) was chosen for the experimental comparison in optimal partitioning. This evolutionary self-adaptation of \( F \) and \( CR \) in DE/rand/1/bin was proposed by Brest et al. [2]. The values of \( F \) and \( CR \) are initialized randomly for each point in population and survive with the individuals in population, but they can be randomly mutated in each generation with given probabilities \( \tau_1 \) and \( \tau_2 \). If mutation happens, new values of \( CR \in [0, 1] \) uniformly distributed, and \( F \) also distributed uniformly in \([F_l, F_u] \) are used in generating a trial vector and stored in the new population. Input parameters are usually set to \( F_l = 0.1, F_u = 0.9, \tau_1 = 0.1, \) and \( \tau_2 = 0.1 \) as applied in [2].

The second algorithm used in the experiments is DE with composite trial vector generation strategies and control parameters (CoDE), presented this year by Wang et al. [24]. CoDE algorithm was also compared with adaptive DE variants [2, 11, 16, 28]. In extensive benchmark tests, CoDE outperformed all of these algorithms except \( EPSDE \). Comparing with \( EPSDE \), it was competitive. The CoDE combines three well-studied trial vector strategies with three control parameter settings in a random way to generate trial vectors. The strategies are rand/1/bin, rand/2/bin, and current-to-rand/1/ and all the three strategies are performed when generating a new trial vector (line 4 in Algorithmus 1). It results in having three mutant vectors. They take part in a crossover and the offspring vector with the least function value is used as a trial vector. The values of control parameters \( F \) and \( CR \) are chosen randomly from the parameter pool containing \([F = 1.0, CR = 0.1], [F = 1.0, CR = 0.9], \) and \( [F = 0.8, CR = 0.2] \). This CoDE is used as a very efficient DE variant for optimal partitioning in this study.

Adaptive DE with competition of different DE strategies (\( CompDE \)) was presented in [21]. Any of \( H \) strategies in the pool can be chosen for the generation of a new trial point \( y \). A strategy is selected randomly with probability \( q_h, h = 1, 2, \ldots, H \). At the start the values of probability are set uniformly, \( q_h = 1/H \), and they are modified according to the success rate in the preceding steps of the search process. The \( h \)th setting is
considered successful if it generates such a trial vector \( y \) satisfying \( f(y) \leq f(x_i) \). Probability \( q_h \) is evaluated as the relative frequency according to
\[
q_h = \frac{n_h + n_0}{\sum_{j=1}^{H}(n_j + n_0)},
\]
where \( n_h \) is the current count of the \( h \)th setting successes, and \( n_0 > 0 \) is an input parameter. The setting of \( n_0 > 1 \) prevents from a dramatic change in \( q_h \) by one random successful use of the \( h \)th strategy. To avoid degeneration of the search process, the current values of \( q_h \) are reset to their starting values if any probability \( q_h \) decreases below some given limit \( \delta > 0 \). The input parameters controlling competition are recommended to set up to \( n_0 = 2 \) and \( \delta = 1/(5 \times H) \). For optimal partitioning we use a variant of CompDE that appeared well-performing and robust in several benchmark tests \([22]\). In this CompDE variant, denoted \( b6e6 \) hereafter, 12 strategies are in competition \((H = 12)\), six of them use a binomial crossover, the others an exponential crossover. The randrl/1/ mutation \((6)\) is applied in all the strategies, two different values of control parameter \( F \) are used, \( F = 0.5 \) and \( F = 0.8 \). Binomial crossover uses three different values of \( CR \), \( CR \in \{0, 0.5, 1\} \). Values of \( CR \) for exponential crossover are evaluated as roots of the equation \((9)\), three values of probability \( p_m \) are set up equidistantly in the interval \((1/d, 1)\). These 12 very different competing strategies (small and medium value of \( F \), six very different crossover operators) give a good chance for balancing exploration and exploitation in the different stages of the search process.

5 Encoding in Clustering Problems

When using population-based heuristics, such as various modifications of differential evolution, it is desirable to solve the problem how to encode a feasible partition of objects. Data matrix \( Z \) is size of \( n \times p \) with real-valued elements and it should be partitioned into \( g \) clusters. Each center of the cluster of a partition could be considered to be just one vector of length \( p \) so that each partition could be represented by \( g \)-tuple of such vectors. Therefore, any partition of \( g \) clusters can be encoded using a floating point array of length \( g \times p \). This encoding is also used in \([3, 13]\). Each object is classified into a cluster with the least Euclidean distance to the center of the cluster. If happens that a cluster or even more clusters are empty within the search process and makes the current classification unfeasible, such classification can be repaired for example by assignment of a center of the cluster. If happens that a cluster or even more clusters are empty within the search process and makes the current classification unfeasible, such classification can be repaired for example by assignment of a center of the cluster. If happens that a cluster or even more clusters are empty within the search process and makes the current classification unfeasible, such classification can be repaired for example by assignment of a center of the cluster.

Another question arises when we consider how to encode an object-to-cluster association. In this paper, a direct encoding of the object-to-cluster association proposed by Raghavan and Birchand \([17]\) was used. The encoding scheme is based on the idea to represent any feasible partition by a vector (of length \( n \)) whose \( i \)th component gives the number of the corresponding cluster. However, this scheme has a large disadvantage because it is ambiguous. For instance, the vectors \((2,2,3,1,1)\) and \((3,3,1,2,2)\) represent the same partitions. Such unambiguity could make the search by EAs inefficient because the mutation and crossover are performed on individuals with similar objective function values but different in their representation (different order of cluster centers). The application of evolutionary operators to such population might make the search rather chaotic. Rearrangement of the rank of cluster centers making the object-to-cluster association as similar as possible for all individuals in the current population can avoid such a chaotic search and can increase the efficiency of the algorithm. To find the proper rearrangement of one individual representation making the object-to-cluster association equivalent to object-to-cluster association of another individual is easy when the both individuals produces the same classifications like in a small example shown above. However, the rearrangement is difficult when the classifications differ. To find the most similar object-to-cluster association is a hard problem. Exhaustive search has complexity \( O(g!) \) and it is not applicable for higher values of \( g \). Due to the fact, a new heuristic search of most similar object-to-cluster association was developed. It is based on the comparing of the \( g \) most promising rearrangements and the selection of the best of them.

Such rearrangement of all individuals in current population to the individual with the least objective function value is applied after completing each generation. DE variants where this recoding is used are denoted by suffix “G” at the end of their labels.

6 Hybrid DE with k-means Algorithm

In order to improve the search of minimum value of the clustering criterion, a modified hybrid DE algorithm was proposed. After finding a trial vector \( y \) satisfying the condition \( f(y) \leq f(x_i) \), see line 5 in Algorithm 1, the k-means algorithm with \( y \) as input is used to get the locally best solution. This solution is than used as a trial vector in line 6 of Algorithm 1. Advantages of k-means algorithm are fast convergence to a local minimum and low complexity \( O(n) \). In DE variants, where this hybrid local search is used, the prefix “H” appears at the beginning of their labels hereafter.
7 Data used in Benchmark Tests

All DE algorithms were tested on real-word data provided for benchmark tests by the first author of paper [13]. It involved four well-known data sets, namely: iris plants data set, Wisconsin breast cancer data set, glass identification data set, and vowel data set. The data was used to compare our experimental results with those given in [19], because the data used in their experiments differs slightly from the sets available in the Machine learning repository [1].

Iris plants data set (hereafter iris) contains three different classes (Iris setosa, Iris versicolour, and Iris Virginica) of 50 objects each, where each object has four numeric attributes (sepal length, sepal width, petal length, and petal width). No missing attribute values occur.

Wisconsin breast cancer data set (bcw) contains only two classes: malignant (458 objects) and benign (241 objects), each object being characterized by nine numeric attributes (clump thickness, cell size uniformity, cell shape uniformity, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses). In the original data set, there are 16 objects containing a single missing attribute value. When the objects with missing values were omitted, the database contained overall 683 objects, out of which 444 objects belonging to malignant class and 239 objects to benign class.

Glass identification data set (glass) includes data of six classes (types of glass): building window float processed (70 objects), building windows non float processed (76 objects), vehicle windows float processed (17 objects), containers (13 objects), table ware (9 objects), and hand lamps (29 objects), each object having nine numeric attributes (refractive index and weight contents of Na, Mg, Al, Si, K, Ca, Ba, and Fe oxides). No missing attribute values are present.

Vowel data set (vowel) consists of overall 870 objects (Indian Telugu sounds) each of which has three numeric attributes (first, second, and third vowel frequencies). There are six strongly overlapping classes: ò (72 objects), a (89 objects), i (172 objects), u (151 objects), e (207 objects), and o (180 objects).

8 Experiments and Control Parameter Settings

These four real-world data sets (bcw, glass, iris, vowel) were used in the numerical experiments. The search space $D$ was chosen to give the $Z$ matrix domain $[z_{\text{min}}, z_{\text{max}}]$, where $z_{\text{min}}$ and $z_{\text{max}}$ are vectors of minimum and maximum values of each variable in data set $Z$. Population size was set up to $NP = 2 \times d$, $d = g \times p$, in all experiments. Individuals in initial population were generated as $g$ randomly taken rows of matrix $Z$ like in [13].

The stopping condition of the search was in the form

$$\frac{f_{\text{max}} - f_{\text{min}}}{f_{\text{min}}} < \varepsilon \quad \text{OR} \quad nfe \geq 5000 \times d$$

(11)

where $f_{\text{max}}, f_{\text{min}}$ are maximum and minimum TRW values in the current generation, respectively, and $nfe$ is the number of objective-function evaluations. In the adaptive plain DE variants without hybrid local search the stopping value is set up to $\varepsilon = 1 \times 10^{-3}$, for hybrid variants to $\varepsilon = 1 \times 10^{-2}$. The control parameters of k-means algorithm are the relative difference of objective function values in two subsequent steps and allowed maximum number of iterations. These parameters were set up to $1 \times 10^{-5}$ and 100, respectively. For each algorithm and each test problem, 30 independent runs were performed.

Time cost of a run was measured by the number of objective-function evaluations denoted $nfe$ in results. For easier comparison, the accuracy of minimum function value found in a run is evaluated by the number of digits identical with the known correct solution. The number of duplicated digits $\lambda$ is calculated using log relative error [15] as

$$\lambda = \begin{cases} 
0 & \text{if } \frac{|f_{\text{min}} - f^*_{\text{min}}|}{f_{\text{min}}} \geq 1 \\
8 & \text{if } \frac{|f_{\text{min}} - f^*_{\text{min}}|}{f_{\text{min}}} < 1 \times 10^{-8} \\
-\log_{10} \left( \frac{|f_{\text{min}} - f^*_{\text{min}}|}{f_{\text{min}}} \right) & \text{otherwise,} 
\end{cases}$$

(12)

where $f^*_{\text{min}} \neq 0$ denotes the correct value and $f_{\text{min}}$ denotes the value found by the search. The $f^*_{\text{min}}$ for TRW criterion are given in Table 1. Except vowel data set, the $f^*_{\text{min}}$ values are the same as the best values of TRW reported in [13]. For vowel data set, the value of the TRW found in [23] is less than the least value reported in [13].

9 Results for non-hybrid adaptive DE

The result obtained from experiments with adaptive non-hybrid variants od DE are summarized in Table 2. The reliability of the search is expressed by $R$, which is the percentage of runs that found the solution with $\lambda > 4$.
Table 1: The minimum values of TRW criterion \( (f_{\text{min}}^*) \) found before

<table>
<thead>
<tr>
<th>data</th>
<th>bcw</th>
<th>iris</th>
<th>glass</th>
<th>vowel</th>
</tr>
</thead>
<tbody>
<tr>
<td>( g )</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>( p )</td>
<td>9</td>
<td>4</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>( n )</td>
<td>683</td>
<td>150</td>
<td>214</td>
<td>871</td>
</tr>
</tbody>
</table>

\( f_{\text{min}}^* \) for \( TRW \) 19323.17 7885.14 336.06 30 686 238

what means that at least four digits in the minimum \( TRW \) value found by an algorithm are the same as in the right solution \( TRW^* \). Reordering of centers making the classification of individuals in the current population as similar as possible (the variants with suffix “G”) brought no benefit for the convergence of algorithms. When comparing the corresponding pairs of the variants, average values of \( nfe \) are even higher for all the problems stopping before \( nfe \) limit. When comparing different variant of adaptive DE, \( jDE \) was the best performing algorithm, while \( CoDE \) was the worst one. \( CoDE \) algorithm stopped due to maximum \( nfe \) limit in difficult problems (glass and vowel) in all runs and its solution quality measured by either average \( \lambda \) or reliability \( R \) is also the worst among the variants in comparison.

10 Results for hybrid adaptive DE

Basic numerical characteristics for the hybrid DE variants are shown in Table 3. The meaning of symbols in the Table is the same as described in section 9. In comparison with non-hybrid DE, it is apparent that the search of the optimal classification by hybrid algorithms is much faster (almost ten times in average) and the reliability of the search for the difficult problems is higher (cf. Table 2 and 3). The difference in the convergence of hybrid DE variants is clearly evident in Figure 2. In the case of hybrid DE, the rearrangement of centers making the classification of all the individuals in the current population as similar as possible (the variants with suffix “G”) brought benefit for the convergence of algorithms, mainly in difficult problems glass and vowel. For easy problems, the benefit from rearrangement is negligible despite the statistical significance of one-way ANOVA tests. From practical point of view, the significant difference in performance of the algorithms for difficult problems is important. When comparing different variant of adaptive hybrid DE, the best performing algorithm is \( Hb6e6G \). From Tukey-Kramer multiple-comparison tests, the \( Hb6e6G \) is significantly better than all the remaining five variants in both the glass and the vowel problems.
11 Conclusion

Adaptive DE was applied to solve the problem of optimal partitional clustering. A hybridization of DE algorithm was proposed. In the hybrid variants of DE, the simple k-means algorithm was used for local search. Non-hybrid and hybrid variants of three different adaptive DE algorithms were used in the solution of four real-world benchmark problems. It was found that hybrid DE variants perform much better than non-hybrid ones. When comparing the hybrid DE variants, the best performing algorithm was $Hb6e6G$, which is a competitive DE with 12 competing strategies and rearrangement of cluster centers to make individuals of the current population as similar as possible. The convergence of this algorithm was significantly faster compared with the others and even the reliability of the search was equal to or better than other variants. We can conclude that hybrid $Hb6e6G$ variant of competitive DE is a very promising algorithm for optimal partitioning and it is recommendable for real-world problems. Our future work will be focused on testing this algorithm using other criteria of optimal partitioning and more benchmark data sets.

Acknowledgement: This work was supported by the research projects MSM 6198898701 of Czech Ministry of Education and SGS 22/2011 of University of Ostrava.

References


