Comparative Study of the Starting Stage of Adaptive Differential Evolution on the Induction of Oblique Decision Trees

Miguel Ángel Morales-Hernández[1], Rafael Rivera-López[2(✉)], Efrén Mezura-Montes[3], Juana Canul-Reich[4], and Marco Antonio Cruz-Chávez[5]

[1] Laboratorio Nacional de Informática Avanzada, Xalapa, Mexico
[2] DSC, TecNM, Instituto Tecnológico de Veracruz, Veracruz, Mexico
rafael.rl@veracruz.tecnm.mx
[3] IIIA, Universidad Veracruzana, Xalapa, Mexico
emezura@uv.mx
[4] DACyTI, Universidad Juárez Autónoma de Tabasco, Cunduacán, Mexico
juana.canul@ujat.mx
[5] CIICAP, Universidad Autónoma del Estado de Morelos, Cuernavaca, Mexico
mcruz@uaem.mx

Abstract. This study describes the application of four adaptive differential evolution algorithms to generate oblique decision trees. A population of decision trees encoded as real-valued vectors evolves through a global search strategy. Three schemes to create the initial population of the algorithms are applied to reduce the number of redundant nodes (whose test condition does not divide the set of instances). The results obtained in the experimental study aim to establish that the four algorithms have similar statistical behavior. However, using the dipole-based start strategy, the JSO method creates trees with better accuracy. Furthermore, the Success-History based Adaptive Differential Evolution with linear population reduction (LSHADE) algorithm stands out for inducing more compact trees than those created by the other variants in the three initializations evaluated.

Keywords: Oblique Decision Trees · Differential Evolution · Initialization stage

1 Introduction

The growing interest in using machine learning (ML) techniques to solve prediction problems and support decision-making in almost any area of human activity

Mexico's National Council of Humanities, Science, and Technology (CONAHCYT) awarded a scholarship to the first author (CVU 1100085) for graduate studies at the Laboratorio Nacional de Informática Avanzada (LANIA).

H. Calvo et al. (Eds.): MICAI 2023 Workshops, LNAI 14502, pp. 439–452, 2024.
https://doi.org/10.1007/978-3-031-51940-6_34

is undeniable. Using artificial neural networks (ANNs) has allowed novel application development. However, ANNs are black-box models that require more means to explain how they make predictions. Therefore, their use has begun to be regulated to prevent wrong decisions supported using these methods in areas such as medicine and economics [29]. This has driven the use of post hoc methods to generate explanations of ANNs predictions [31]. Other ML strategies, such as decision trees (DTs), are white-box models with high interpretability and transparency levels. Among the DT types are those that use oblique hyperplanes in their internal nodes, producing more compact DTs than those that use a single attribute to make their partitions. However, traditional methods to induce DTs present some drawbacks, such as overfitting, selection bias toward multi-valued attributes, and instability to small changes in the training set [16]. For this reason, other inducing techniques have been proposed that, instead of applying recursive partitioning techniques, perform a search in the space of possible DTs. Evolutionary algorithms (EAs), such as genetic algorithms and genetic programming, have been widely applied to find near-optimal DTs that are more precise than those created by traditional techniques [16].

Differential Evolution (DE) is an EA that has successfully solved numerical optimization problems, and its standard versions have also been applied to induce DTs [6,9,18,24,25]. To the best of our knowledge, adaptive DE versions have not been applied to induce oblique DTs. On the other hand, it has been observed that an initial random population produces redundant internal nodes that affect the size and precision of the induced model. In this study, we analyzed the effect of using four adaptive DE versions to induce oblique DTs (JADE, SHADE, LSHADE, and JSO) and the effect on model precision and size when two additional strategies are used to create the initial DE population: dipoles and centroids. The experimental results establish that the four algorithms exhibit similar statistical behavior. However, using the dipole-based start strategy, the JSO method creates trees with better accuracy. Furthermore, the LSHADE algorithm induces more compact trees in the three initializations evaluated.

The rest of this paper is organized into four additional sections. Section 2 introduces the oblique DT characteristics and the adaptive DE approaches are described in Sect. 3. In Sect. 4, elements for the comparative study are detailed. Section 5 presents the experimental details and results. Finally, in Sect. 6, the conclusions of this study are presented, and some directions for future work are defined.

2 Oblique Decision Trees

DTs are classification models that split datasets according to diverse criteria, such as the distance between instances or the reduction of classification error. These models create a hierarchical structure using test conditions (internal nodes) and class labels (leaf nodes), allowing visualization of attribute importance in decisions and how they are used to classify an instance. DTs are the most popular interpretable algorithm for classification and regression [13]. Depending on the number of attributes evaluated in each internal node, two decision tree

types are induced: univariate (axis parallel DTs) and multivariate (oblique and non-linear DTs). In particular, oblique DTs (ODTs) use test conditions representing hyperplanes having an oblique orientation relative to the axes of the instance space. ODTs are generally smaller and more accurate than univariate DTs, but they are generally more difficult to interpret [4]. ID3 [22], C4.5 [23] and CART [2] are the most popular methods for inducing univariate DTs, and CART and OC1 [19] are well-known methods for creating oblique DTs. Figure 1 shows an example of an ODT. On the right of this figure, the instance space is split using two oblique hyperplanes.

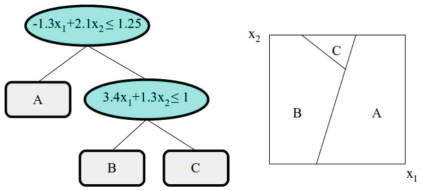

Fig. 1. Example of an oblique decision tree.

3 Adaptive Differential Evolution Approaches

DE is an EA evolving a population of real-valued vectors $\mathbf{x}_i = \left(x_{i,1}, x_{i,2}, \cdots, x_{i,n}\right)^T$ of n variables, to find a near-optimal solution to an optimization problem [21]. Instead of implementing traditional crossover and mutation operators, DE applies a linear combination of several randomly selected individuals to produce a new individual. Three randomly selected candidate solutions ($\mathbf{x}_a$, $\mathbf{x}_b$, and $\mathbf{x}_c$) are linearly combined to yield a mutated solution $\mathbf{x}_{mut}$, as follows:

$$\mathbf{x}_{mut} = \mathbf{x}_a + F\left(\mathbf{x}_b - \mathbf{x}_c\right) \tag{1}$$

where F is a scale factor for controlling the differential variation.

The mutated solution is utilized to perturb another candidate solution $\mathbf{x}_{cur}$ using the binomial crossover operator defined as follows:

$$x_{new,j} = \begin{cases} x_{mut,j} & \text{if } r \leq Cr \vee j = k \\ x_{cur,j} & \text{otherwise} \end{cases} ; j \in \{1, \ldots, n\} \tag{2}$$

where $x_{new,j}$, $x_{mut,j}$ and $x_{cur,j}$ are the values in the j-th position of $\mathbf{x}_{new}$, $\mathbf{x}_{mut}$ and $\mathbf{x}_{cur}$, respectively, $r \in [0, 1)$ and $k \in \{1, \ldots, n\}$ are uniformly distributed random numbers, and Cr is the crossover rate.

Finally, $\mathbf{x}_{new}$ is selected as a member of the new population if it has a better fitness value than that of $\mathbf{x}_{cur}$.

DE starts with a population of randomly generated candidate solutions whose values are uniformly distributed in the range $[x_{min}, x_{max}]$ as follows:

$$x_{i,j} = x_{min} + r\left(x_{max} - x_{min}\right); i \in \{1, \ldots, NP\} \wedge j \in \{1, \ldots, n\} \tag{3}$$

where NP is the population size.

DE is characterized by using fewer parameters than other EAs and by its spontaneous self-adaptability, diversity control, and continuous improvement [7]. Although DE requires fewer parameters than other EAs, its performance is sensitive to the values selected for Cr, F, and NP [32]. In the literature, several approaches exist to improve DE performance using techniques to adjust the values of its parameters or combine the advantages of different algorithm variants. Methods that adjust the algorithm parameters can be considered global approaches when the parameters are updated at the end of each generation, and all population members use their values [5,17]. On the other hand, the most successful approaches are those in which each individual uses a different value of the control parameters [27,30]. Finally, there are other methods where different mutation or recombination strategies are combined within the evolutionary process [11,26]. In this study, four adaptive DE versions are used:

1) JADE: This DE variant introduces a successful mutation strategy and an adaptive parameter method using Gaussian and Cauchy distributions [30]. The *current-to-pbest* mutation, shown in Eq. (4), improves the balance between search space exploration and exploitation by allowing the selection of an individual ($\mathbf{x}_{pbest}$) from a subset of the p best individuals in the population to create a mutant vector ($\mathbf{x}_{mut}$). An optional external archive is also used to diversify the donor vectors. $\mathbf{x}_a$ is chosen randomly from the current population, and $\mathbf{x}_b$ are selected from this external archive that recorded solutions previously discarded during the evolutionary process.

$$\mathbf{x}_{mut} = \mathbf{x}_{cur} + F_i(\mathbf{x}_{pbest} - \mathbf{x}_{cur}) + F_i(\mathbf{x}_a - \mathbf{x}_b) \tag{4}$$

JADE uses F and Cr parameter values adjusted to each i-th individual in the population. F_i is selected from a Cauchy distribution, $F_i = randc_i(\mu_F, 0.1)$, and Cr_i is generated using a normal distribution, $Cr_i = randn_i(\mu_{Cr}, 0.1)$. μ_{Cr} is updated at the end of each generation using the arithmetic mean of the set of all successful crossover probabilities. μ_F is similarly computed, but the Lehmer mean of all successful scale factors is used.

2) SHADE: The Success-History based Adaptive DE (SHADE) is an enhanced JADE version employing a historical record of the pair of μ_{Cr} and μ_F values [27]. These values are randomly selected to create each new individual instead of using the same values in each generation.

3) LSHADE: A population size linear reduction strategy is applied in this SHADE variant [28]. The population decreases linearly in each generation until its size equals a minimum value.

4) JSO: This LSHADE improvement replaces the F_i parameter in the second term of Eq. (4) for a weighted F_i value updated as a function of the number of objective function evaluations [3].

Recursive-partitioning and global-search strategies to induce ODTs have been implemented using DE-based approaches. In the first case, OC1-DE [25], the Adapted JADE with Multivariate DT (AJADE-MDT) method [12], and the Parallel-Coordinates (PA-DE) algorithm [6] evolve a population of real-valued

individuals to find near-optimal hyperplanes. In the other case, two methods implement a global search strategy to find a near-optimal ODT: (1) The Perceptron DT (PDT) method [9,18], where the hyperplane coefficients of one DT are encoded with a real-valued individual. Hyperplane-independent terms and the class label of leaf nodes are stored in two additional vectors. In each DE iteration, the mutation parameters are randomly altered. A group of new DTs randomly created replaces the worst individuals in the population. (2) The DE algorithm to build ODTs (DE-ODT) [24], where the size of the real-valued vector is computed as a factor of the number of internal nodes of an ODT estimated using the number of dataset attributes.

4 Comparative Study Details

In this study, we use the mapping scheme introduced by the DE-ODT method [24]. DE-ODT evolves a population of ODTs encoded in fixed-length real-valued vectors. Figure 2 shows the scheme for converting a DE individual into an ODT. The steps of this mapping scheme are described in the following paragraphs.

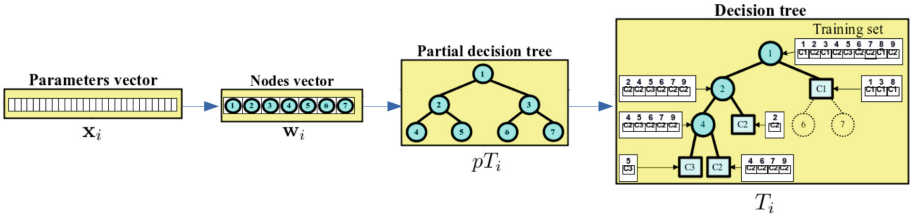

Fig. 2. Mapping scheme used on the DE-ODT method.

1) ODTs linear representation: Each candidate solution encodes only the internal nodes of a complete binary ODT stored in a fixed-length real-valued vector ($\mathbf{x}_i$). This vector represents the set of hyperplanes used as the ODT test conditions. The vector size (n) is determined using both the number of features (d) and the number of class labels (s) of the training set, as follows:

$$n = n_e(d+1) \tag{5}$$

where $n_e = 2^{max(H_i,H_l)-1} - 1$, $H_i = \lceil log_2(d+1) \rceil$, and $H_l = \lceil log_2(s) \rceil$

2) Hyperplanes construction: Vector $\mathbf{x}_i$ is used to build the vector $\mathbf{w}_i$ encoding the sequence of candidate internal nodes of a partial ODT. Since the values of $\mathbf{x}_i$ represent the hyperplane coefficients contained in these nodes, the following criterion applies: Values $\{x_{i,1}, \ldots, x_{i,d+1}\}$ are assigned to the hyperplane h_1, the values $\{x_{i,d+2}, \ldots, x_{i,2d+2}\}$ are assigned to the hyperplane h_2, and so on. These hyperplanes are assigned to the elements of $\mathbf{w}_i$: h_1 is assigned to $w_{i,1}$, h_2 is assigned to $w_{i,2}$, an so on.

3) Partial ODT construction: A straightforward procedure is applied to construct the partial DT (pT_i) from $\mathbf{w}_i$: First, the element in the initial location of $\mathbf{w}_i$ is used as the root node of pT_i. Next, the remaining elements of $\mathbf{w}_i$ are inserted in pT_i as successor nodes of those previously added so that each new level of the tree is completed before placing new nodes at the next level, in a similar way to the breadth-first search strategy. Since a hyperplane divides the instances into two subsets, each internal node has assigned two successor nodes.

4) Decision tree completion: In the final stage, several leaf nodes are added in pT_i by evaluating the training set to build the final ODT T_i. One instance set is assigned to one internal node (starting with the root node), and by evaluating each instance with the hyperplane associated with the internal node, two instance subsets are created and assigned to the successor nodes. This assignment is repeated for each node in pT_i. Two cases should be considered: (1) If an instance set is located at the end of a branch of pT_i, two leaf nodes are created and designated as its successor nodes. (2) If the instance set contains elements for the same class, the internal node is labeled as a leaf node, and its successor nodes are removed if they exist. Furthermore, an internal node is labeled as a leaf when it contains an empty instance subset.

As one example of this inducing approach, Fig. 3 shows two DT induced from the well-known Iris dataset: The left DT is created using J48 method from Weka library [8], and the right DT is and ODT induced using LSHADE-based approach. The ODT is more compact and more accurate than the left DT.

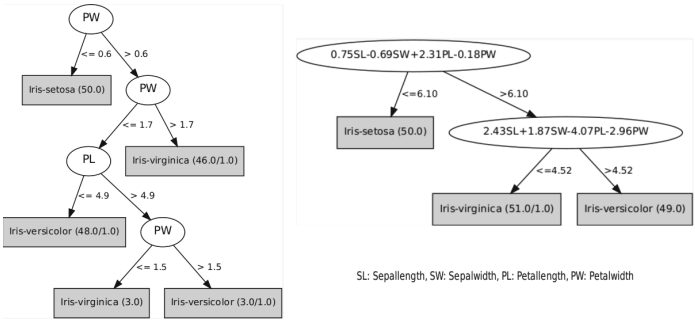

Fig. 3. DTs for Iris dataset using J48 (Left) and LSHADE (Right).

On the other hand, three initialization strategies are analyzed in this work:

1) Random initialization: This is the classic initialization strategy previously described in Eq. (3). This approach generates many hyperplanes that do not divide the instances into two subsets, producing redundant nodes impacting the induced model's size and precision.

2) Dipoles: A dipole is a pair of training instances. A mixed dipole occurs when these instances have different classes [1]. Dipoles were first used to

induce ODTs through a recursive partitioning strategy [15]. A hyperplane h_i must split a mixed dipole to divide the instances into two nonempty subsets. Each individual of the initial population is built by creating hyperplanes splitting randomly selected mixed dipoles from the training set. h_1 uses a mixed dipole chosen from the training set, h_2 uses a mixed dipole from the first subset created by h_1, h_3 from the second subset, and so on until the number of internal nodes n_e is completed. In particular, a random hyperplane is created if an internal node contains an empty instance subset.

3) Centroids: This initialization strategy is proposed in this work. It is similar to the previous one, but the hyperplanes are created using the centroid of the instance set instead of a random mixed dipole. The centroid is a dummy instance where each value is the middle between the instance set's minimum and maximum values.

5 Experiments and Results

In this section, the experimental study conducted to analyze and compare the performance of the algorithms is presented. First, the datasets used in this study and the validation technique applied in the comparative analysis are described. Next, the experimental results and statistical tests are outlined. Finally, a discussion about the results is provided. Thirteen datasets (shown in Table 1), from the UCI machine learning repository are used in the experimental study [14]. These datasets have only numerical attributes since ODT internal nodes are a linear combination of their values.

Table 1. Datasets description.

Dataset	Instances	Features	Classes	Dataset	Instances	Features	Classes
Glass	214	9	7	Breast-tissue-6	106	9	6
Australian	690	14	2	Ionosphere	351	34	2
Iris	150	4	3	Balance-scale	625	4	3
Ecoli	336	7	8	Heart-statlog	270	13	2
Wine	178	13	3	Liver-disorders	345	6	2
Diabetes	768	8	2	Parkinsons	195	22	2
Seeds	210	6	3				

The methods used in this study are implemented in the Java languaje, using the JMetal [20] and Weka [8] libraries. The implemented algorithms run 30 times for each dataset and initialization scheme, using the ten-fold cross-validation sampling strategy to estimate the precision and size of induced trees. Subsequently, the Friedman test is applied to statistical analysis of the results obtained. Friedman test is selected since it has been demonstrated that the conditions to apply parametric tests are not satisfied to EA-based machine learning

methods [10]. In the subsequent tables of this section, the best result for each dataset is highlighted with bold numbers, and the numbers in parentheses refer to the ranking reached by each method for each dataset. The last row in these tables indicates the average ranking of each method.

Tables 2, 3, 4, 5, 6, and 7 show the average accuracy and the average size (number of leaf nodes) of the trees obtained for each dataset and each DE variant, respectively.

Table 2. Average accuracy of DE variants using random initialization.

Datasets	DE	JADE	SHADE	LSHADE	JSO
Iris	96.31 (3)	96.04 (5)	96.53 (2)	**96.76** (1)	96.20 (4)
Glass	60.75 (3)	**61.12** (1)	60.34 (5)	60.37 (4)	60.78 (2)
Parkinsons	78.02 (2)	77.71 (4)	**78.12** (1)	77.95 (3)	77.28 (5)
Diabetes	67.82 (4)	**68.16** (1)	67.59 (5)	68.14 (2)	68.04 (3)
Australian	75.49 (3)	75.39 (4)	75.28 (5)	75.52 (2)	**75.60** (1)
Ionosphere	89.32 (3)	89.03 (5)	89.23 (4)	89.56 (2)	**89.65** (1)
Balance-scale	**90.59** (1)	90.35 (4.5)	90.44 (2)	90.35 (4.5)	90.39 (3)
Ecoli	**80.31** (1)	79.75 (5)	79.98 (4)	80.00 (2.5)	80.00 (2.5)
Heart-statlog	**74.04** (1)	73.40 (3)	73.35 (4)	73.91 (2)	73.27 (5)
Liver-disorders	**68.19** (1)	67.46 (5)	68.08 (4)	68.09 (3)	68.14 (2)
Wine	86.72 (2)	86.10 (5)	**86.91** (1)	86.39 (4)	86.70 (3)
Breast-tissue-6	55.47 (4)	**56.07** (1)	55.38 (5)	55.79 (2.5)	55.79 (2.5)
Seeds	90.75 (5)	90.90 (4)	91.10 (2)	**91.38** (1)	91.06 (3)
Average ranking	**2.538**	3.654	3.385	2.577	2.846

Table 3. Average accuracy of DE variants using dipole-based initialization.

Datasets	DE	JADE	SHADE	LSHADE	JSO
Iris	96.07 (5)	96.22 (4)	96.27 (3)	**96.40** (1)	96.38 (2)
Glass	63.80 (5)	64.38 (2.5)	64.38 (2.5)	63.96 (4)	**64.53** (1)
Parkinsons	**83.30** (1)	83.16 (2)	83.09 (3)	82.87 (5)	82.89 (4)
Diabetes	74.01 (3)	74.00 (4)	73.78 (5)	**74.21** (1)	74.02 (2)
Australian	84.52 (2)	84.26 (5)	84.43 (3)	84.37 (4)	**84.58** (1)
Ionosphere	88.74 (5)	88.76 (4)	**89.27** (1)	89.03 (2)	88.98 (3)
Balance-scale	90.22 (5)	90.34 (3)	90.41 (2)	90.31 (4)	**90.51** (1)
Ecoli	82.20 (4)	82.10 (5)	**82.38** (1)	82.25 (2.5)	82.25 (2.5)
Heart-statlog	78.75 (2)	77.95 (5)	77.96 (4)	**78.83** (1)	78.70 (3)
Liver-disorders	69.18 (2)	69.06 (3.5)	69.04 (5)	69.06 (3.5)	**69.33** (1)
Wine	81.16 (3)	80.39 (5)	81.24 (2)	**81.91** (1)	80.43 (4)
Breast-tissue-6	**54.43** (1)	52.74 (5)	53.30 (3)	53.14 (4)	53.71 (2)
Seeds	88.44 (4)	**89.22** (1)	88.38 (5)	88.51 (3)	88.56 (2)
Average ranking	3.231	3.769	3.038	2.769	**2.192**

Table 4. Average accuracy of DE variants using centroid-based initialization.

Datasets	DE	JADE	SHADE	LSHADE	JSO
Iris	**96.18** (1)	96.00 (4)	95.93 (5)	96.16 (2.5)	96.16 (2.5)
Glass	65.61 (2)	64.81 (5)	65.37 (4)	**65.62** (1)	65.42 (3)
Parkinsons	**82.44** (1)	82.19 (2)	82.15 (3)	91.97 (4)	81.88 (5)
Diabetes	**72.02** (1)	71.88 (3)	71.92 (2)	71.85 (4)	71.77 (5)
Australian	78.00 (5)	78.45 (3)	78.71 (2)	**78.81** (1)	78.38 (4)
Ionosphere	89.67 (5)	89.77 (4)	89.89 (3)	**90.12** (1)	90.11 (2)
Balance-scale	45.76 (3)	45.76 (3)	45.76 (3)	45.76 (3)	45.76 (3)
Ecoli	82.88 (2)	82.62 (4)	82.67 (3)	**82.92** (1)	82.45 (5)
Heart-statlog	77.35 (2)	76.46 (5)	**77.68** (1)	76.93 (3)	76.60 (4)
Liver-disorders	69.60 (2)	69.15 (4)	69.29 (3)	**69.81** (1)	68.62 (5)
Wine	81.24 (4)	**81.67** (1)	81.50 (3)	81.57 (2)	81.07 (5)
Breast-tissue-6	54.43 (3)	53.14 (4)	52.92 (5)	54.75 (2)	**54.94** (1)
Seeds	88.19 (4)	88.14 (5)	88.32 (2)	88.29 (3)	**88.71** (1)
Average ranking	2.692	3.615	3.000	**2.192**	3.500

Table 5. Average size of DE variants using random initialization.

Datasets	DE	JADE	SHADE	LSHADE	JSO
Iris	7.21 (4.5)	**7.16** (1)	7.19 (3)	7.21 (4.5)	7.17 (2)
Glass	11.93 (3)	11.91 (2)	**11.87** (1)	12.04 (4)	12.07 (5)
Parkinsons	12.64 (4)	12.48 (2)	**12.39** (1)	12.59 (3)	12.73 (5)
Diabetes	21.44 (2)	21.5 (4)	21.63 (5)	**20.99** (1)	21.47 (3)
Australian	20.08 (5)	**19.27** (1)	19.62 (4)	19.48 (2)	19.51 (3)
Ionosphere	22.18 (4)	21.94 (3)	22.37 (5)	**21.46** (1)	21.8 (2)
Balance-scale	12.53 (5)	12.32 (2)	12.49 (4)	**12.21** (1)	12.41 (3)
Ecoli	14.18 (4)	14.27 (5)	**14.03** (1)	14.16 (3)	14.12 (2)
Heart-statlog	**8.41** (1)	8.71 (5)	8.60 (3)	8.48 (2)	8.64 (4)
Liver-disorders	12.56 (5)	12.29 (3)	12.22 (2)	**12.19** (1)	12.31 (4)
Wine	6.84 (5)	6.82 (4)	6.78 (3)	**6.56** (1)	6.61 (2)
Breast-tissue-6	**17.47** (1)	17.53 (3)	17.5 (2)	17.74 (5)	17.67 (4)
Seeds	7.94 (3)	**7.73** (1)	8.06 (5)	7.90 (2)	7.95 (4)
Average ranking	3.577	2.769	3.000	**2.346**	3.308

Table 6. Average size of DE variants using dipole-based initialization..

Datasets	DE	JADE	SHADE	LSHADE	JSO
Iris	**6.56** (1)	6.68 (3)	6.70 (4)	6.62 (2)	6.77 (5)
Glass	23.93 (2)	23.96 (3)	23.98 (4)	**23.90** (1)	24.16 (5)
Parkinsons	30.61 (4)	30.90 (5)	30.44 (2)	**30.16** (1)	30.59 (3)
Diabetes	20.73 (5)	**20.22** (1)	21.43 (4)	20.48 (2)	20.53 (3)
Australian	**12.40** (1)	12.79 (5)	12.74 (4)	12.5 (2)	12.55 (3)
Ionosphere	24.74 (5)	**24.05** (1)	24.10 (2)	24.33 (4)	24.16 (3)
Balance-scale	12.76 (2.5)	12.78 (4)	12.92 (5)	**12.61** (1)	12.76 (2.5)
Ecoli	15.46 (2)	**15.45** (1)	15.48 (3)	15.62 (5)	15.57 (4)
Heart-statlog	5.41 (3)	5.56 (4)	5.58 (5)	**5.22** (1.5)	**5.22** (1.5)
Liver-disorders	11.44 (2)	11.49 (3)	**11.22** (1)	11.50 (4)	11.64 (5)
Wine	**8.93** (1)	9.85 (5)	9.73 (4)	9.14 (2)	9.61 (3)
Breast-tissue-6	22.06 (5)	**20.93** (1)	20.95 (2)	21.53 (4)	21.22 (3)
Seeds	11.52 (5)	11.32 (4)	11.19 (2)	**11.16** (1)	11.28 (3)
Average ranking	2.884	3.077	3.308	**2.346**	3.385

Table 7. Average size of DE variants using centroid-based initialization.

Datasets	DE	JADE	SHADE	LSHADE	JSO
Iris	7.00 (5)	6.97 (4)	6.89 (2)	**6.87** (1)	6.91 (3)
Glass	24.17 (3)	24.13 (2)	24.20 (4)	**24.05** (1)	24.21 (5)
Parkinsons	22.28 (4)	**21.87** (1)	22.07 (2)	22.34 (5)	22.26 (3)
Diabetes	15.76 (2)	15.81 (3)	15.99 (5)	15.84 (4)	**15.59** (1)
Australian	10.09 (4)	10.19 (5)	**9.95** (1)	10.04 (2)	10.08 (3)
Ionosphere	19.09 (5)	19.04 (4)	18.89 (2)	18.93 (3)	**18.79** (1)
Balance-scale	1.11 (3)	1.11 (3)	1.11 (3)	1.11 (3)	1.11 (3)
Ecoli	16.35 (4.5)	16.35 (4.5)	16.33 (3)	**16.19** (1)	16.31 (2)
Heart-statlog	**5.79** (1)	6.16 (5)	5.97 (2)	6.01 (3)	6.07 (4)
Liver-disorders	11.92 (2)	12.22 (5)	12.13 (3.5)	**11.91** (1)	12.13 (3.5)
Wine	9.93 (5)	9.54 (2)	9.65 (3)	**9.44** (1)	9.77 (4)
Breast-tissue-6	22.10 (3)	**21.77** (1)	21.84 (2)	22.3 (5)	22.16 (4)
Seeds	11.04 (4)	11.18 (5)	**10.84** (1)	11.02 (3)	10.96 (2)
Average ranking	3.500	3.423	2.577	**2.538**	2.962

From Tables 2, 3, and 4, it is observed that the methods getting better accuracy are: the standard DE using random initialization, JSO with dipole-based initialization, and LSHADE with centroid-based start strategy. On the other hand, Tables 5, 6, and 7 show that the LSHADE method produces more compact ODTs using the three start strategies.

Tables 8 and 9 show the best results obtained by the DE versions for accuracy and tree size, respectively.

Table 8. Accuracy for the best methods for each initialization strategy.

Datasets	Classes	Random DE	Dipoles JSO	Centroids LSHADE
Parkinsons	2	78.02 (3)	**82.89** (1)	81.97 (2)
Diabetes	2	67.82 (3)	**74.02** (1)	71.85 (2)
Australian	2	75.49 (3)	**84.58** (1)	78.81 (2)
Ionosphere	2	89.32 (2)	88.98 (3)	**90.12** (1)
Heart-statlog	2	74.04 (3)	**78.70** (1)	76.93 (2)
Liver-disorders	2	68.19 (3)	69.33 (2)	**69.81** (1)
Iris	3	96.31 (2)	**96.38** (1)	96.16 (3)
Balance-scale	3	**90.59** (1)	90.51 (2)	45.76 (3)
Wine	3	**86.72** (1)	80.43 (3)	81.57 (2)
Seeds	3	**90.75** (1)	88.56 (2)	88.29 (3)
Breast-tissue-6	6	**55.47** (1)	53.71 (3)	54.75 (2)
Glass	7	60.75 (3)	64.53 (2)	**65.62** (1)
Ecoli	8	80.31 (3)	82.25 (2)	**82.92** (1)
Average ranking		2.231	**1.846**	1.923

Table 9. Tree size for the best methods for each initialization strategy.

Datasets	Classes	Random LSHADE	Dipoles LSHADE	Centroids LSHADE
Parkinsons	2	**12.59** (1)	30.16 (3)	22.34 (2)
Diabetes	2	20.99 (3)	20.48 (2)	**15.84** (1)
Australian	2	19.48 (3)	12.50 (2)	**10.04** (1)
Ionosphere	2	21.46 (2)	24.33 (3)	**18.93** (1)
Heart-starlog	2	8.48 (3)	**5.22** (1)	6.01 (2)
Liver-disorders	2	12.19 (3)	**11.50** (1)	11.91 (2)
Iris	3	7.21 (3)	**6.62** (1)	6.87 (2)
Wine	3	**6.56** (1)	9.14 (2)	9.44 (3)
Seeds	3	**7.90** (1)	11.16 (3)	11.02 (2)
Balance-scale	3	12.21 (2)	12.61 (3)	**1.11** (1)
Breast-tissue-6	6	**17.74** (1)	21.53 (2)	22.30 (3)
Glass	7	**12.04** (1)	23.90 (2)	24.05 (3)
Ecoli	8	**14.16** (1)	15.62 (2)	16.19 (3)
Average ranking		**1.923**	2.077	2.000

From the results on in Table 8, the Friedman test shows a p-value of 0.5836, pointing out that the three methods behave similarly, but it is observed that, on average, the JSO algorithm gets ODTs with better accuracy than the other approaches. Also, Table 8 shows that JSO with dipoles finds more accurate ODTs from datasets with two or three classes (Parkinsons, diabetes, Australian, Heart-statlog, Iris), and LSHADE with centroids finds accurate ODTs from those with two (Ionosphere, Liver-disorders) or with more than three classes (Glass, Ecoli). On the other hand, Standard DE with random initialization induces better ODTs for imbalanced datasets: Balance-scale has three classes with 288, 288, and 49 instances, respectively, and the Wine dataset has three classes with 59, 71, and 48 instances, respectively. Breast-tissue-6 has six classes with 22, 21, 14, 15, 16, and 18 instances, and only Seeds is a balanced dataset with 70 instances per class.

The statistic value computed by the Friedman test from results in Table 9 shows a p-value of 0.926, indicating that the three initialization strategies built ODTs of similar sizes. The centroid initialization scheme produces the smallest ODTs for datasets with two classes, and the random initialization creates more tiny ODTs for multiclass datasets (Wine, Breast-tissue-t, Glass and Ecoli).

6 Conclusions and Future Work

The experimental results indicate that it is important to continue studying the application of adaptive variants of DE to solve non-numerical optimization problems because it is necessary to analyze in detail the search space's characteristics and the strategies for mapping between DE individuals and their tree-like representation. LSHADE and JSO obtained slightly better results than the other approaches, which is to be expected since they are improved versions of SHADE and JADE methods. In future work, we will integrate a tree-pruning strategy and a more effective method to remove redundant nodes into the decision tree induction process.

References

1. Bobrowski, L.: Piecewise-linear classifiers, formal neurons and separability of the learning sets. In: Proceedings of 13th International Conference on Pattern Recognition, vol. 4, pp. 224–228 (1996)
2. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Chapman and Hall (1984)
3. Brest, J., Maučec, M.S., Bošković, B.: Single objective real-parameter optimization: algorithm jSO. In: CEC 2017, pp. 1311–1318 (2017)
4. Cantú-Paz, E., Kamath, C.: Using evolutionary algorithms to induce oblique decision trees. In: GECCO 2000, pp. 1053–1060 (2000)
5. Draa, A., Bouzoubia, S., Boukhalfa, I.: A sinusoidal differential evolution algorithm for numerical optimisation. Appl. Soft Comput. **27**, 99–126 (2015)

6. Estivill-Castro, V., Gilmore, E., Hexel, R.: Constructing interpretable decision trees using parallel coordinates. In: Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J.M. (eds.) ICAISC 2020. LNCS (LNAI), vol. 12416, pp. 152–164. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-61534-5_14
7. Feoktistov, V.: Differential Evolution: In Search of Solutions. Springer, New York (2007). https://doi.org/10.1007/978-0-387-36896-2
8. Frank, E., Hall, M., Witten, I.: The WEKA Workbench. Online Appendix (2016). https://www.cs.waikato.ac.nz/ml/weka/Witten_et_al_2016_appendix.pdf
9. Freitas, A.R.R., Silva, R.C.P., Guimarães, F.G.: Differential evolution and perceptron decision trees for fault detection in power transformers. In: Snášel, V., Abraham, A., Corchado, E. (eds.) SOCO 2012. AISC, vol. 188, pp. 143–152. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-32922-7_15
10. García, S., Fernández, A., Luengo, J., Herrera, F.: A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. Soft. Comput. **13**, 959–977 (2009). https://doi.org/10.1007/s00500-008-0392-y
11. Ghosh, A., Das, S., Panigrahi, B.K., Das, A.K.: A noise resilient differential evolution with improved parameter and strategy control. In: CEC 2017, pp. 2590–2597 (2017)
12. Jariyavajee, C., Polvichai, J., Sirinaovakul, B.: Searching for splitting criteria in multivariate decision tree using adapted JADE optimization algorithm. In: SSCI 2019, pp. 2534–2540 (2019)
13. Kamath, U., Liu, J.: Explainable Artificial Intelligence: An Introduction to Interpretable Machine Learning. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-83356-5
14. Kelly, M., Longjohn, R., Nottingham, K.: The UCI Machine Learning Repository (2023). https://archive.ics.uci.edu
15. Kręt owski, M.: An evolutionary algorithm for oblique decision tree induction. In: Rutkowski, L., Siekmann, J.H., Tadeusiewicz, R., Zadeh, L.A. (eds.) ICAISC 2004. LNCS (LNAI), vol. 3070, pp. 432–437. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24844-6_63
16. Kretowski, M.: Evolutionary Decision Trees in Large-Scale Data Mining. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21851-5
17. Liu, J., Lampinen, J.: A fuzzy adaptive differential evolution algorithm. Soft. Comput. **9**, 448–462 (2005). https://doi.org/10.1007/s00500-004-0363-x
18. Lopes, R.A., Freitas, A.R.R., Silva, R.C.P., Guimarães, F.G.: Differential evolution and perceptron decision trees for classification tasks. In: Yin, H., Costa, J.A.F., Barreto, G. (eds.) IDEAL 2012. LNCS, vol. 7435, pp. 550–557. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32639-4_67
19. Murthy, S.K., Kasif, S., Salzberg, S., Beigel, R.: OC1: a randomized algorithm for building oblique decision trees. In: AAAI 1993, vol. 93, pp. 322–327 (1993)
20. Nebro, A.J., Durillo, J.J., Vergne, M.: Redesigning the jMetal multi-objective optimization framework. In: GECCO 2015, pp. 1093–1100 (2015)
21. Price, K., Storn, R.M., Lampinen, J.A.: Differential Evolution: A Practical Approach to Global Optimization. Springer, Heidelberg (2006). https://doi.org/10.1007/3-540-31306-0
22. Quinlan, J.R.: Induction of decision trees. Mach. Learn. **1**(1), 81–106 (1986). https://doi.org/10.1007/BF00116251
23. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)

24. Rivera-Lopez, R., Canul-Reich, J.: A global search approach for inducing oblique decision trees using differential evolution. In: Mouhoub, M., Langlais, P. (eds.) AI 2017. LNCS (LNAI), vol. 10233, pp. 27–38. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57351-9_3
25. Rivera-Lopez, R., Canul-Reich, J., Gámez, J.A., Puerta, J.M.: OC1-DE: a differential evolution based approach for inducing oblique decision trees. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2017. LNCS (LNAI), vol. 10245, pp. 427–438. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59063-9_38
26. Sallam, K.M., Elsayed, S.M., Sarker, R.A., Essam, D.L.: Improved united multi-operator algorithm for solving optimization problems. In: CEC 2018, pp. 1–8 (2018)
27. Tanabe, R., Fukunaga, A.: Success-history based parameter adaptation for differential evolution. In: CEC 2013, pp. 71–78 (2013)
28. Tanabe, R., Fukunaga, A.S.: Improving the search performance of SHADE using linear population size reduction. In: CEC 2014, pp. 1658–1665 (2014)
29. Yeung, K., Lodge, M.: Algorithmic Regulation. Oxford University Press, Oxford (2019)
30. Zhang, J., Sanderson, A.C.: JADE: self-adaptive differential evolution with fast and reliable convergence performance. In: CEC 2007, pp. 2251–2258 (2007)
31. Zhang, Y., Tiňo, P., Leonardis, A., Tang, K.: A survey on neural network interpretability. IEEE Trans. Emerg. Top. Comput. Intell. **5**(5), 726–742 (2021)
32. Zielinski, K., Laur, R.: Stopping criteria for differential evolution in constrained single-objective optimization. In: Chakraborty, U.K. (ed.) Advances in Differential Evolution. SCI, vol. 143, pp. 111–138. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68830-3_4