# A Global Search Approach for Inducing Oblique Decision Trees Using Differential Evolution

Rafael Rivera-Lopez[1] and Juana Canul-Reich[2(✉)]

[1] Departamento de Sistemas y Computación, Instituto Tecnológico de Veracruz, Veracruz, Mexico
rrivera@itver.edu.mx
[2] División Académica de Informática y Sistemas, Universidad Juárez Autónoma de Tabasco, Cunduacán, Mexico
juana.canul@ujat.mx

**Abstract.** This paper describes the application of a Differential Evolution based approach for inducing oblique decision trees in a global search strategy. By using both the number of attributes and the number of class labels in a dataset, this approach determines the size of the real-valued vector utilized for encoding the set of hyperplanes used as test conditions in the internal nodes of an oblique decision tree. Also a scheme of three steps to map the linear representation of candidate solutions into feasible oblique decision trees is described. Experimental results obtained show that this approach induces more accurate classifiers than those produced by other proposed induction methods.

**Keywords:** Machine learning · Classification · Evolutionary algorithms

## 1 Introduction

Evolutionary algorithms (EAs) are population-based search methods that have been successfully applied for providing near-optimal solutions for many computationally complex problems in almost all areas of science and technology. The effectiveness of EAs is due to two factors: (1) they combine a clever exploration of the search space to identify promising areas, and (2) they perform an efficient exploitation of these areas aiming to improve the known solution or solutions. EAs are inspired by evolutionary theories that synthesize the Darwinian evolution through natural selection with the Mendelian genetic inheritance. In particular, Differential Evolution (DE) algorithm is an EA designed for solving optimization problems with variables in continuous domains that, instead of implementing traditional crossover and mutation operators, it applies a linear combination of several randomly selected candidate solutions to produce a new solution [28]. DE has been applied for solving optimization problems arising in several domains of science and engineering including economics, medicine, biotechnology, manufacturing and production, big data and data mining, etc., [25]. In data mining, DE has been utilized for constructing models of classification [19], clustering [7], and rule generation [8] with the aim of identifying


M. Mouhoub and P. Langlais (Eds.): Canadian AI 2017, LNAI 10233, pp. 27–38, 2017.
DOI: 10.1007/978-3-319-57351-9_3

hidden relationships among known instances. DE has been used in conjunction with neural networks [19], support vector machines [20], bayesian classifiers [13], instance based classifiers [12] and decision trees [30] for the induction of classifiers.

In this paper, a differential evolution-based approach named DE-ODT for inducing oblique decision trees in a global search strategy is described. The representation scheme of candidate solutions used by DE-ODT allows to apply DE operators without any modification, and the procedure for mapping a real-valued chromosome into a feasible decision tree ensures to carry out an efficient search in the solution space. DE-ODT is compared with three approaches for inducing oblique decision trees: the Oblique Classifier 1 (OC1) [24], the Perceptron Decision Tree (PDT) method [23], and the EFTI algorithm [32], and with the J48 method [34]. Experimental results obtained in this work show that DE-ODT induces more accurate classifiers than those found by the other methods. In order to describe the implementation of DE-ODT method, this paper is organized as follows: Sect. 2 describes the elements of differential evolution algorithm. The use of evolutionary algorithms for inducing oblique decision trees is presented in Sect. 3, and in Sect. 4 details of DE-ODT method with emphasis in both the determination of the size of candidate solutions and the induction of feasible oblique decision trees are given. Section 5 describes experimental results, and finally, Sect. 6 gives conclusions and future work.

## 2 Differential Evolution Algorithm

DE is a population-based metaheuristic that evolves a set of candidate solutions by applying evolutionary operators in order to find near-optimal solutions to optimization problems. Each candidate solution is encoded using a real-valued vector $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,m})^T$ of $m$ variables.

In this paper, the standard DE algorithm [28], named DE/rand/1/bin in accordance to the nomenclature adopted for referencing DE variants, is used as a procedure for oblique decision tree induction that implements a global search strategy. DE/rand/1/bin uses the following evolutionary operators:

1. **Mutation:** Three randomly selected candidate solutions ($\mathbf{x}_a$, $\mathbf{x}_b$ and $\mathbf{x}_c$) are linearly combined, using Eq. (1), to yield a mutated solution $\mathbf{x}_{mut}$.

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

   where $F$ is a scale factor for controlling the differential variation.
2. **Crossover:** The mutated solution is utilized to perturb another candidate solution $\mathbf{x}_{cur}$ using the binomial crossover operator defined as follows:

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

   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 $k \in \{1, \ldots, m\}$ are uniformly distributed random numbers, and $Cr$ is the crossover rate.

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

DE/rand/1/bin, described in Algorithm 1, starts with a population of randomly generated candidate solutions whose values are uniformly distributed in the range $[x_{min}, x_{max}]$ as follows:

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

where $NP$ is the population size. New populations of candidate solutions are iteratively created until a stop condition is reached and then the best solution of the last population is returned. It can be observed that DE requires few control parameters ($Cr$, $F$, and $NP$) in comparison to other EAs.

**Algorithm 1.** Standard DE algorithm introduced in [28].

$k \leftarrow 0$
$\mathbf{X}_k \leftarrow \emptyset$
**for** $i$ in $\{1, \ldots, NP\}$ **do**
    $\mathbf{x}_i \leftarrow$ Randomly generated candidate solution using (3)
    $\mathbf{X}_k \leftarrow \mathbf{X}_k \cup \{\mathbf{x}_i\}$
**end for**
**while** stop condition is not reached **do**
    $k \leftarrow k + 1$
    $\mathbf{X}_k \leftarrow \emptyset$
    **for** $cur$ in $\{1, \ldots, NP\}$ **do**
        $\{\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c\} \leftarrow$ Randomly selected candidate solutions of $\mathbf{X}_{k-1}$
        $\mathbf{x}_{mut} \leftarrow$ Mutated candidate solution using (1)
        $\mathbf{x}_{new} \leftarrow$ Perturbed candidate solution of $\mathbf{x}_{cur}$ using (2)
        $\mathbf{x}_{sel} \leftarrow \begin{cases} \mathbf{x}_{new} & \text{if } fitness(\mathbf{x}_{new}) \text{ is better than } fitness(\mathbf{x}_{cur}) \\ \mathbf{x}_{cur} & \text{otherwise} \end{cases}$
        $\mathbf{X}_k \leftarrow \mathbf{X}_k \cup \{\mathbf{x}_{sel}\}$
    **end for**
**end while**
**return** The best candidate solution in $\mathbf{X}_k$

## 3 EAs for Inducing Oblique Decision Trees

A decision tree (DT) is a hierarchical structure composed of a set of nodes containing both test conditions (internal nodes) and class labels (leaf nodes) that are joined by arcs representing the possible result values of each test condition. A DT is a classification model induced through a set of training instances which is used for predicting the class membership of new unclassified instances. Each training instance is encoded as a vector $\mathbf{v} = (v_1, v_2, \ldots, v_d, c)^T$ of $d$ variables (attributes or features) and a label $c$ that determines the class membership of

the instance. The simplicity and the high level of interpretability of a DT along with its predictive power has made it one of the most widely used classifiers.

The number of attributes used in the test conditions of a DT determines its type (univariate or multivariate). Since efficient induction methods such as CART [5] and C4.5 [27] generate univariate DTs (also called axis-parallel DTs) it is the most known type of DTs. On the other hand, oblique DTs and nonlinear DTs are multivariate DTs in which a linear combination and a nonlinear composition of attributes in test conditions is utilized, respectively. In particular, oblique DTs use a set of not axis-parallel hyperplanes for splitting the instance space in order to predict the class membership of unclassified instances. A hyperplane is defined as follows:

$$\sum_{i=1}^{d} x_i v_i + x_{d+1} > 0 \tag{4}$$

where $v_i$ is the value of attribute $i$, $x_i$ is a real-valued coefficient used in the hyperplane and $x_{d+1}$ represents the independent term. Oblique DTs are generally smaller and more accurate than univariate DTs but they are generally more difficult to interpret [6].

EAs have been previously applied for DT induction (DTI) and there exist several surveys that describe their implementation [2,11]. Some approaches implement a recursive partitioning strategy in which an EA is used for finding a near-optimal test condition for each tree internal node [6], however, the approach most commonly used is to perform a global search in the solution space with the aim of finding near-optimal DTs [3,17,18,23,31–33]. A genetic algorithm (GA) is an EA that generally employs a linear representation of candidate solutions and its implementation for DTI is associated to the problem of mapping an oblique DT from a linear structure [17]. However, with the application of special genetic operators, GA can use a tree representation for DTI [3,18]. An special GA that evolves a unique candidate solution is used in EFTI method [32] for inducing a complete oblique DT. Furthermore, DE has been utilized for finding the parameter settings of a DTI method [30] and for constructing both univariate DTs [31] and oblique DTs [23]. Finally, since genetic programming (GP) represents its candidate solutions as trees, standard GP [22] and GP variants such as strongly-typed GP [4] and grammar-based GP [1] have been applied for oblique DTI.

## 4 DE-ODT Method for Inducing Oblique DTs

DE-ODT is proposed in this paper as a method for oblique DTI in a global search strategy that evolves a population of candidate solutions encoded as fixed-length real-valued vectors. A similar approach known as PDT method is described in [23] but, although DE-ODT and PDT share the same objective and both implement a global search strategy for DTI with DE, substantial differences in the representation scheme used in DE-ODT allows for induction of more accurate

and compact oblique DTs than PDT and other similar approaches. In the next paragraphs the main elements of DE-ODT method are described.

### 4.1 Global Search Strategy to Generate Near-Optimal Oblique DTs

The great majority of algorithms for DTI apply a recursive partitioning strategy that implements some splitting criterion in order to separate the training instances. Several studies point out that this strategy has three fundamental problems: overfitting [14], selection bias towards multi-valued attributes [15] and instability to small changes in the training set [29]. On the other hand, algorithms that implement a global search strategy can ensure a more efficient exploration of the solution space although it is known that building optimal DTs is NP-Hard [16].

DE-ODT implements a global search strategy with the aim of constructing more accurate oblique DTs, and also for overcoming the inherent problems of the recursive partitioning strategy. Since oblique DTs use hyperplanes with real-valued coefficients as test conditions, the search for near-optimal oblique DTs can be considered a continuous optimization problem, and DE has proven to be a very competitive approach for solving this type of problems. Although other metaheuristic-based approaches have been previously used for classifications tasks, DE-ODT is introduced in this work as a simple and straightforward method in which DE is applied for finding near-optimal solutions, and where each real-valued chromosome encodes only a feasible oblique DT.

### 4.2 Linear Representation of Oblique DTs

Two schemes for encoding candidate solutions in EAs can be used: tree or linear representation. When tree representation is adopted, special crossover and mutation operators must be implemented in order to ensure the construction of only feasible candidate solutions [3,18]. An advantage of this representation is that EAs can evolve DTs with different sizes but it is known that crossover has a destructive effect on the offsprings [18]. On the other hand, if a linear representation is utilized then a scheme for mapping the sequence of values into a DT must be applied [17]. The main advantage of linear representation is that it is applied for encoding candidate solutions in several EAs such as GA, DE and evolutionary strategies and they can be implemented for DTI without any modification. Nevertheless, since these EAs use a fixed-length representation, it is necessary to define a priori this length and this can limit the performance of the induced DTs.

In DE-ODT method each candidate solution encodes only the internal nodes of a complete binary oblique DT stored in breadth-first order in a fixed-length real-valued vector (Fig. 1). This vector encodes the set of hyperplanes used as test conditions of the oblique DT. Vector size is determined using both the number of attributes and the number of class labels.

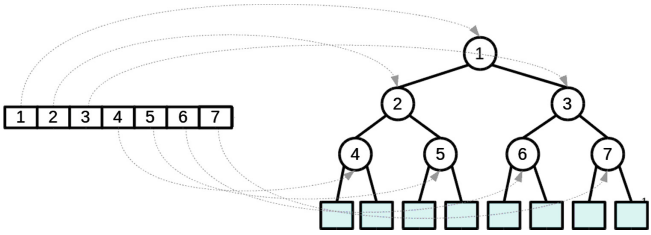



**Fig. 1.** Linear encoding scheme for the internal nodes of a complete binary tree.

### 4.3 Estimated Number of Tree Internal Nodes

Since each internal node of an oblique DT has a hyperplane as test condition, the size of real-valued vector used for encoding each candidate solution is fixed as $n_e(d+1)$, where $n_e$ is the estimated number of internal nodes of a complete binary DT and $d$ is the number of attributes of the training set. Considering that: (1) an oblique DTs is more compact than an univariate DTs when they are induced with the same training set, and (2) the DT size is related to the structure of the training set, DE-ODT determines the value of $n_e$ using both the number of attributes ($d$) and the number of class labels ($k$) in the training set. If, for one complete binary DT, $h$ is the depth, $n_i$ is the number of internal nodes and $n_l$ is the number of leaf nodes, respectively, then $d$ and $k$ can be used as lower bounds for $n_i$ and $n_l$ ($n_i = 2^{h-1} - 1 \geq d$ and $n_l = 2^{h-1} \geq k$), respectively. Using these relations, two estimated depths ($h_i = \lceil \log_2 (d+1) + 1 \rceil$ and $h_l = \lceil \log_2 (k) + 1 \rceil$) are calculated and $n_e$ is obtained as follows:

$$n_e = 2^{\max(h_i, h_l) - 1} - 1 \tag{5}$$

### 4.4 Induction of Feasible Oblique DTs

Since the training set must be utilized for inducing one DT, DE-ODT uses it for mapping each candidate solution into a feasible oblique DT, including its leaf nodes. DE applies the training accuracy of a DT as fitness value within the evolutionary process. The induction of a feasible oblique DT (Fig. 2) in DE-ODT has three steps:

1. **Construction:** First, an empty complete binary DT with $n_e$ nodes (7 nodes in example of Fig. 2) is created. Then, the coefficient values of a hyperplane are assigned to each node in this DT by applying the next criterion: Values on positions $\{1, \ldots, d+1\}$ of this vector are used in the hyperplane of the first node, values on positions $\{d+2, \ldots, 2d+2\}$ are used in the hyperplane of the second node, and so on.
2. **Assignment:** One instance set is assigned to a node (the complete training set for the root node of the tree) and it is labeled as an internal node. For evaluating each instance in this set using the hyperplane associated to the

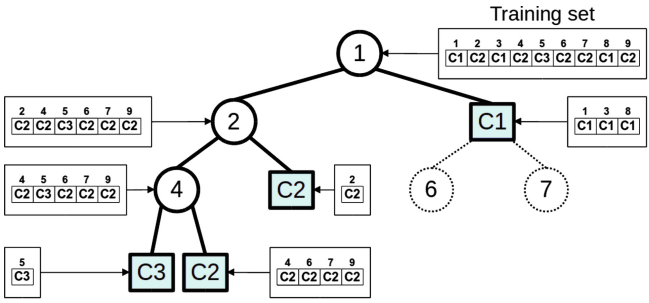



**Fig. 2.** Construction and assignment of a feasible oblique DT.

internal node, two instance subsets are created and they are assigned to the successor nodes of this node. This assignment is repeated for each node of the DT. If the internal node is located at the end of a branch of the DT (node 4 in Fig. 2, for example), then two leaf nodes are created and are designated children of the ending node of the branch. The subsets created are assigned to these leaf nodes. On the other hand, if all instances in the set assigned to the internal node have the same class label, it is labeled as a leaf node (nodes 3 and 5 in Fig. 2) and its successor nodes are removed, if they exist (nodes 6 and 7 in Fig. 2).

3. **Pruning:** Finally, when the assignment of training instances is completed, a pruning procedure is applied for removing tree branches in order to improve the accuracy of the DT induced.

These steps allow to induce feasible oblique DTs with different number of nodes, although the candidate solutions are encoded using fixed-length real-valued vectors.

## 5 Experiments

In order to evaluate the performance of DE-ODT method and for comparison to other approaches, two experiments were conducted using several datasets with numerical attributes chosen from UCI repository [21]. DE-ODT is implemented in Java language using the JMetal library [10]. The parameters used in the

**Table 1.** Parameters used in experiments with DE-ODT.

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Scale factor | 1 | Num. of generations | 50 |
| Crossover rate | 0.9 | Population size | $20\sqrt{d}$ (proposed in [6]) |
| Fitness function | DT accuracy | Pruning method | Reduced error pruning [26] |

experiments are described in Table 1 and the datasets are described in the first four columns of Tables 2 and 3.

In the first experiment, DE-ODT is compared to PDT and J48 methods using the sampling procedure described in [23]: Each dataset is randomly divided into two sets, 85% of instances are used for training and the rest are used for testing. 30 independent runs of each dataset are conducted. For each run, an oblique DT is induced and its test accuracy is calculated. Both average accuracy and average DT size across these 30 runs are obtained. This experimental scheme is replicated using three induction methods: DE-ODT, PDT and J48. Table 2 shows the experimental results[1]: Column 5 shows the average accuracy reported by [23]. Columns 6 and 7 show the average accuracy and the average DT size produced by DE-ODT method and the results obtained by J48 are shown in columns 8 and 9. In this table can be observed that accuracies obtained by DE-ODT method are better than: (1) the accuracies produced by J48 for all datasets, and (2) the accuracies reported by PDT for 7 of 8 datasets. DE-ODT produces more compact oblique-DTs than those produced for J48. In the case of DT size for PDT method, these results were not reported in [23]. Figure 3(a) shows a comparative plot of the average accuracies obtained.

**Table 2.** Results obtained for PDT, DE-ODT and J48.

| (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) |
|---|---|---|---|---|---|---|---|---|
| | | | | PDT | DE-ODT | | J48 | |
| Dataset | Inst. | Attr. | Classes | Acc. | Acc. | Size | Acc. | Size |
| Breast tissue | 106 | 9 | 6 | 39.92 | **64.44** | **5.97** | 33.70 | 7.6 |
| Vertebral column | 310 | 6 | 2 | 84.11 | **93.04** | **2.90** | 79.29 | 5.0 |
| Ecoli | 336 | 7 | 8 | 76.73 | **87.33** | **5.83** | 80.47 | 9.1 |
| Glass | 214 | 9 | 7 | 58.08 | **71.04** | **7.03** | 64.42 | 11.5 |
| Hill valley | 1212 | 100 | 2 | **99.45** | 81.35 | 3.83 | 50.49 | **1.0** |
| Iris | 150 | 4 | 3 | 94.35 | **99.97** | **3.37** | 93.77 | 3.5 |
| Libras movement | 360 | 90 | 15 | 31.85 | **56.34** | **26.1** | 55.80 | 26.2 |
| Sonar | 208 | 60 | 2 | 77.29 | **94.73** | **4.03** | 71.13 | 5.4 |

In the second experiment, DE-ODT is compared to OC1 and EFTI methods using 5 independents runs of 10-fold stratified cross-validation. EFTI is a novel approach that reports better results with several datasets than other EA-based algorithms. Results are shown in Table 3: Columns 5–8 show the average accuracy and the average DT size reported in [32] for both OC1 and EFTI methods. Columns 9 and 10 show the average accuracy and the average DT size produced by DE-ODT method. In Table 3 can be observed that the accuracies obtained for DE-ODT are better than those obtained by both EFTI and OC1 methods

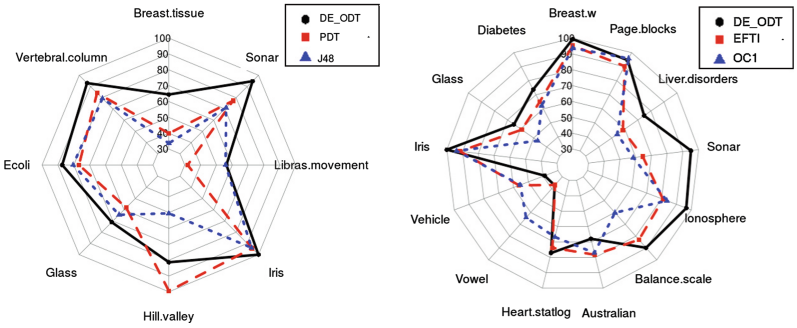
[1] Highest values for each dataset are in bold.

in 10 of 13 datasets, although the DT size are slightly larger than the DT size reported for EFTI method. Figure 3(b) shows a comparative plot of the average accuracies obtained in this experiment.

**Table 3.** Results obtained by OC1, EFTI and DE-ODT.

| (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) | (10) |
|---|---|---|---|---|---|---|---|---|---|
| | | | | OC1 | | EFTI | | DE-ODT | |
| Dataset | Inst. | Attr. | Classes | Acc. | Size | Acc. | Size | Acc. | Size |
| Breast-w | 683 | 9 | 2 | 95.53 | 3.68 | 96.59 | **2.02** | **99.56** | 3.48 |
| Diabetes | 768 | 8 | 2 | 73.03 | 6.54 | 74.94 | **2.35** | **81.25** | 3.70 |
| Glass | 214 | 9 | 7 | 62.04 | 13.12 | 70.82 | 7.31 | **75.05** | **7.22** |
| Iris | 150 | 4 | 3 | 95.60 | 3.54 | 94.13 | **3.00** | **100.00** | 3.20 |
| Vehicle | 846 | 18 | 4 | 68.16 | 33.54 | **68.75** | **5.44** | 55.51 | 6.70 |
| Vowel | 990 | 10 | 11 | **74.55** | 51.68 | 54.90 | 17.86 | 55.11 | **9.96** |
| Heart-statlog | 270 | 13 | 2 | 76.30 | 4.70 | 81.28 | **2.12** | **83.70** | 3.38 |
| Australian | 690 | 14 | 2 | 83.63 | 6.10 | **84.51** | **2.28** | 77.10 | 4.08 |
| Balance-scale | 625 | 4 | 3 | 71.58 | 3.08 | 87.85 | **2.40** | **92.70** | 3.06 |
| Ionosphere | 351 | 34 | 2 | 88.26 | 6.18 | 86.39 | **2.49** | **97.61** | 6.26 |
| Sonar | 208 | 60 | 2 | 70.39 | 6.76 | 74.64 | **2.38** | **96.54** | 5.38 |
| Liver-disorders | 345 | 6 | 2 | 67.23 | 5.38 | 70.36 | **2.33** | **81.97** | 2.92 |
| Page-blocks | 5473 | 10 | 5 | 97.05 | 23.78 | 93.16 | **2.04** | **97.17** | 5.12 |

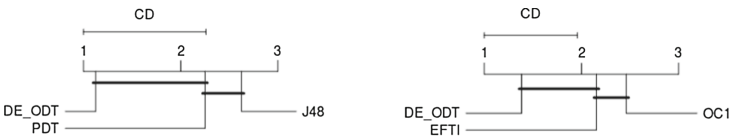
In order to evaluate the performance of DE-ODT a statistical test of the results obtained was realized. Friedman test is applied for detecting the existence of significant differences between the performance of two or more methods and the Nemenyi post-hoc test is utilized for checking these differences. Nemenyi test uses the average ranks of each classifier and checks for each pair of classifiers




(a) First experiment (b) Second experiment

**Fig. 3.** Average accuracies obtained in experiments.

whether the difference between their ranks is greater than the critical difference $\left(CD = q_\alpha \sqrt{k(k+1)/(6N)}\right)$ defined in [9], where $k$ is the number of methods, $N$ is the number of datasets, and $q_\alpha$ is a critical value associated of the significance level $\alpha$. For the first experiment, Friedman statistic for 3 methods using 8 datasets is 9.75 and the p-value obtained is 0.007635 for 2° of freedom (dof) of chi-square distribution. This p-value indicates the existence of statistical differences between these methods and then Nemenyi test post-hoc is conducted. Figure 4(a) shows the CDs obtained for Nemenyi test and it shown as well that DE-ODT has better performance than PDT and J48. For the second experiment, Friedman statistic for 3 methods using 13 datasets is 8.0, the p-value is 0.01832 with 2 dof and it also indicates statistical differences between these methods. Figure 4(b) shows the CDs obtained for Nemenyi test and in it can be observed that DE-ODT has better performance than EFTI and OC1 methods.




(a) First experiment (b) Second experiment

**Fig. 4.** Comparison of classifiers using Nemenyi post-hoc test.

## 6 Conclusions

In this paper, a DE-based method implementing a global search strategy for finding a near-optimal oblique DTs is introduced. This search strategy ensures a more efficient exploration of solution space in order to reach more compact and accurate DTs. DE-ODT uses a fixed-length linear representation of oblique DTs that permits to apply DE operators without any modification. By using the training set in the mapping scheme implemented in this work, the induction of feasible oblique DTs is guaranteed. DE-ODT was evaluated using two sampling procedures with several UCI datasets and statistical tests suggest that DE-ODT achieves a better performance than other induction methods. In general, since DE-ODT uses DE for constructing oblique DTs, it induces more accurate oblique DTs than those produced by other proposed induction methods. Based on our results, future work will be oriented to evaluate other DE variants for inducing oblique DTs and to investigate the effect of using several parameter configurations on the DE-ODT performance, also more experiments will be conducted for analyzing the DE-ODT execution time, as well as to compare the DE-ODT performance with those obtained by other classification methods such as random forest and support vector machines.

**Acknowledgments.** This work has been supported by the Mexican Government (CONACyT FOMIX-DICC project No. TAB-2014-C01-245876 and the PROMEP-SEP project No. DSA/103.5/15/6409).

## References


1. Agapitos, A., O'Neill, M., Brabazon, A., Theodoridis, T.: Maximum margin decision surfaces for increased generalisation in evolutionary decision tree learning. In: Silva, S., Foster, J.A., Nicolau, M., Machado, P., Giacobini, M. (eds.) EuroGP 2011. LNCS, vol. 6621, pp. 61–72. Springer, Heidelberg (2011). doi:10.1007/978-3-642-20407-4_6
2. Barros, R.C., Basgalupp, M.P., Carvalho, A., Freitas, A.A.: A survey of evolutionary algorithms for decision-tree induction. IEEE Trans. Syst. Man Cybern.-Part C: Appl. Rev. **42**(3), 291–312 (2012). doi:10.1109/TSMCC.2011.2157494
3. Basgalupp, M.P., Barros, R.C., de Carvalho, A.C., Freitas, A.A.: Evolving decision trees with beam search-based initialization and lexicographic multi-objective evaluation. Inf. Sci. **258**, 160–181 (2014). doi:10.1016/j.ins.2013.07.025
4. Bot, M.C.J., Langdon, W.B.: Improving induction of linear classification trees with genetic programming. In: Whitley, L.D., Goldberg, D.E., Cantú-Paz, E., Spector, L., Parmee, I.C., Beyer, H.G. (eds.) GECCO-2000, pp. 403–410. Morgan Kaufmann, Burlington (2000)
5. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Taylor & Francis, Abingdon (1984)
6. Cantú-Paz, E., Kamath, C.: Inducing oblique decision trees with evolutionary algorithms. IEEE Trans. Evol. Comput. **7**(1), 54–68 (2003). doi:10.1109/TEVC.2002.806857
7. Das, S., Abraham, A., Konar, A.: Automatic clustering using an improved differential evolution algorithm. IEEE Trans. Syst. Man Cybern.-Part A: Syst. Hum. **38**(1), 218–237 (2008). doi:10.1109/tsmca.2007.909595
8. De Falco, I.: Differential evolution for automatic rule extraction from medical databases. Appl. Soft Comput. **13**(2), 1265–1283 (2013). doi:10.1016/j.asoc.2012.10.022
9. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**(Dec), 1–30 (2006)
10. Durillo, J.J., Nebro, A.J.: jMetal: a Java framework for multiobjective optimization. Adv. Eng. Softw. **42**(10), 760–771 (2011). doi:10.1016/j.advengsoft.2011.05.014
11. Espejo, P.G., Ventura, S., Herrera, F.: A survey on the application of genetic programming to classification. IEEE Trans. Syst. Man Cybern.-Part C: Appl. Rev. **40**(2), 121–144 (2010). doi:10.1109/TSMCC.2009.2033566
12. García, S., Derrac, J., Triguero, I., Carmona, C.J., Herrera, F.: Evolutionary-based selection of generalized instances for imbalanced classification. Knowl.-Based Syst. **25**(1), 3–12 (2012). doi:10.1016/j.knosys.2011.01.012
13. Geetha, K., Baboo, S.S.: An empirical model for thyroid disease classification using evolutionary multivariate Bayesian prediction method. Glob. J. Comput. Sci. Technol. **16**(1), 1–9 (2016)
14. Hawkins, D.M.: The problem of overfitting. ChemInform **35**(19) (2004). doi:10.1002/chin.200419274
15. Hothorn, T., Hornik, K., Zeileis, A.: Unbiased recursive partitioning: a conditional inference framework. J. Comput. Graph. Stat. **15**(3), 651–674 (2006). doi:10.1198/106186006x133933
16. Hyafil, L., Rivest, R.L.: Constructing optimal binary decision trees is NP-complete. Inf. Process. Lett. **5**(1), 15–17 (1976). doi:10.1016/0020-0190(76)90095-8
17. Kennedy, H.C., Chinniah, C., Bradbeer, P., Morss, L.: The contruction and evaluation of decision trees: a comparison of evolutionary and concept learning methods. In: Corne, D., Shapiro, J.L. (eds.) AISB EC 1997. LNCS, vol. 1305, pp. 147–161. Springer, Heidelberg (1997). doi:10.1007/BFb0027172

18. Krętowski, M., Grześ, M.: Evolutionary learning of linear trees with embedded feature selection. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Żurada, J.M. (eds.) ICAISC 2006. LNCS (LNAI), vol. 4029, pp. 400–409. Springer, Heidelberg (2006). doi:10.1007/11785231_43
19. Leema, N., Nehemiah, H.K., Kannan, A.: Neural network classifier optimization using differential evolution with global information and back propagation algorithm for clinical datasets. Appl. Soft Comput. **49**, 834–844 (2016). doi:10.1016/j.asoc.2016.08.001
20. Li, J., Ding, L., Li, B.: Differential evolution-based parameters optimisation and feature selection for support vector machine. Int. J. Comput. Sci. Eng. **13**(4), 355–363 (2016)
21. Lichman, M.: UCI Machine Learning Repository (2013). University of California, Irvine. http://archive.ics.uci.edu/ml
22. Liu, K.H., Xu, C.G.: A genetic programming-based approach to the classification of multiclass microarray datasets. Bioinformatics **25**(3), 331–337 (2009). doi:10.1093/bioinformatics/btn644
23. 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). doi:10.1007/978-3-642-32639-4_67
24. Murthy, S.K., Kasif, S., Salzberg, S., Beigel, R.: OC1: a randomized algorithm for building oblique decision trees. In: Proceedings of AAAI 1993, vol. 93, pp. 322–327 (1993)
25. Plagianakos, V.P., Tasoulis, D.K., Vrahatis, M.N.: A review of major application areas of differential evolution. In: Chakraborty, U.K. (ed.) Advances in Differential Evolution. SCI, vol. 143, pp. 197–238. Springer, Heidelberg (2008). doi:10.1007/978-3-540-68830-38
26. Quinlan, J.R.: Simplifying decision trees. Int. J. Hum.-Comput. Stud. **27**(3), 221–234 (1987). doi:10.1006/ijhc.1987.0321
27. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, Burlington (1993)
28. Storn, R., Price, K.: Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**(4), 341–359 (1997). doi:10.1023/A:1008202821328
29. Strobl, C., Malley, J., Tutz, G.: An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests. Psychol. Methods **14**(4), 323–348 (2009). doi:10.1037/a0016973
30. Tušar, T.: Optimizing accuracy and size of decision trees. In: ERK-2007, pp. 81–84 (2007)
31. Veenhuis, C.B.: Tree based differential evolution. In: Vanneschi, L., Gustafson, S., Moraglio, A., Falco, I., Ebner, M. (eds.) EuroGP 2009. LNCS, vol. 5481, pp. 208–219. Springer, Heidelberg (2009). doi:10.1007/978-3-642-01181-8_18
32. Vukobratović, B., Struharik, R.: Evolving full oblique decision trees. In: CINTI 2015, pp. 95–100. IEEE (2015). doi:10.1109/CINTI.2015.7382901
33. Wang, P., Tang, K., Weise, T., Tsang, E.P.K., Yao, X.: Multiobjective genetic programming for maximizing ROC performance. Neurocomputing **125**, 102–118 (2014). doi:10.1016/j.neucom.2012.06.054
34. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, Burlington (2005)