IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 23, 2017, accepted December 20, 2017, date of publication January 1, 2018,
date of current version February 28, 2018.

Digital Object Identifier 10.1109/ACCESS.2017.2788700

Construction of Near-Optimal Axis-Parallel Decision Trees Using a Differential-Evolution-Based Approach

RAFAEL RIVERA-LOPEZ[1] AND JUANA CANUL-REICH[2]
[1]Departamento de Sistemas y Computación, Instituto Tecnológico de Veracruz, Veracruz 91800, México
[2]División Académica de Informática y Sistemas, Universidad Juárez Autónoma de Tabasco, Cunduacán 86690, México

Corresponding author: Juana Canul-Reich (juana.canul@ujat.mx)

This work was supported in part by the Mexican Government through CONACyT FOMIX-DICC under Project TAB-2014-C01-245876 and in part by PROMEP-SEP under Project DSA/103.5/15/6409.

ABSTRACT In this paper, a differential-evolution-based approach implementing a global search strategy to find a near-optimal axis-parallel decision tree is introduced. In this paper, the internal nodes of a decision tree are encoded in a real-valued chromosome, and a population of them evolves using the training accuracy of each one as its fitness value. The height of a complete binary decision tree whose number of internal nodes is not less than the number of attributes in the training set is used to compute the chromosome size, and a procedure to map a feasible axis-parallel decision tree from one chromosome is applied, which uses both the smallest-position-value rule and the training instances. The best decision tree in the final population is refined replacing some leaf nodes with sub-trees to improve its accuracy. The differential evolution algorithm has been successfully applied in conjunction with several supervised learning methods to solve numerous classification problems, due to it exhibiting a good tradeoff between its exploitation and exploration skills, and to the best of our knowledge, it has not been utilized to build axis-parallel decision trees. To obtain reliable estimates of the predictive performance of this approach and to compare its results with those achieved by other methods, a repeated stratified ten-fold cross-validation procedure is applied in the experimental study. A statistical analysis of these results suggests that our approach is better as a decision tree induction method as compared with other supervised learning methods. Also our results are comparable to those obtained with random forest and one multilayer-perceptron-based classifier.

INDEX TERMS Decision trees, differential evolution, metaheuristics, smallest-position-value rule, supervised learning.

I. INTRODUCTION

Machine learning techniques to build models from known data have gained importance over the past few years due to the growing demand for data analysis in disciplines such as data science, business intelligence, and big data. Decision trees, artificial neural networks, and support vector machines, as well as clustering methods, have been widely-used to build predictive models. Many real-world problems such as the prediction of high-impact weather events [1], the analysis of traffic situations [2], the study of customer feedbacks [3], and the evaluation of credit risks [4], among other diverse applications, have benefited with such models.

The use of one particular machine learning technique to build a model from a specific set of training instances depends on the required level of interpretability, scalability, and robustness of the model produced. In particular, decision trees (DTs) are classification models characterized by their high levels of interpretability and robustness. Knowledge learned via a DT is understandable due to its graphical representation [5], and also DTs can handle noise or data with missing values and to make correct predictions [6].

Although it is known that one greedy criterion does not guarantee to find an optimal solution [7], DTs are ordinarily constructed through a recursive partition strategy that searches an optimal local split of the training set at each stage of their induction process. On the other hand, algorithms implementing a global search strategy are capable of finding near-optimal DTs, but they are computationally expensive [8], and a way of coping with this disadvantage is the use of metaheuristics (MHs) such as

evolutionary algorithms (EAs) and swarm intelligence (SI) methods.

In this paper, a Differential-Evolution-based approach implementing a global search strategy to find a near-optimal axis-parallel DT is described. In this method, named DE-ADT$_{\text{SPV}}$, the internal nodes of a DT are encoded in a vector of real-valued parameters, and a population of them evolves using the training accuracy of each DT as its fitness value. The size of the real-valued vector is estimated a priori according to the characteristics of the dataset whose classification model is constructed, and a scheme to map a feasible axis-parallel DT from this vector is applied, using both the Smallest-Position-Value (SPV) rule and the training instances. The SPV rule [9] has been implemented with several MHs such as Particle Swarm Optimization (PSO) [10] and Differential Evolution (DE) [11] to solve combinatorial optimization problems but, to the best of our knowledge, it has not been used to build a DT. A statistical analysis of the results obtained by the DE-ADT$_{\text{SPV}}$ method suggests that our approach shows a better performance as a DT induction (DTI) method in comparison with other proposed methods. Also our results are comparable to those obtained with random forest and one multilayer-perceptron-based classifier. The DE-ADT$_{\text{SPV}}$ method is publicly available to the community interested in the use of MHs for DTI. It can be freely obtained from the Github site https://github.com/rafaelriveralopez/DE-ADT.

The rest of this document is structured as follows: Section 2 provides a set of basic definitions about DTs, as well as a description of its implemented induction strategies, with emphasis on the use of MHs for this purpose, and on the representation of candidate solutions. The details of the DE algorithm are given in Section 3, and in Section 4 a brief review of existing literature related to MH-based approaches for DTI encoding their candidate solutions as a sequence of values is introduced. Section 5 presents the description of the elements included in the DE-ADT$_{\text{SPV}}$ method, and the experimental results are discussed in Section 6. Finally, Section 7 holds the conclusions and the future work of this proposal.

II. INDUCTION OF DECISION TREES THROUGH METAHEURISTICS

A DT is a white-box classification model representing its decisions through a tree-like structure composed of a set of nodes containing both *test conditions* (internal nodes) and *class labels* (leaf nodes). These nodes are joined by arcs simbolizing the possible outcomes of each test condition in the tree. A DT is a *rooted directed tree* $T = \big(G(V, E), v_1\big)$, where V is the set of nodes, E is the set of edges joining pairs of nodes in V, and v_1 is its *root node* [12]. In particular, if V has m nodes, for any $j = \{1, \ldots, m\}$, the set of successor nodes of $v_j \in V$ is defined as follows:

$$\mathcal{N}^+(v_j) = \Big\{v_k \in V : k = \{1, \ldots, m\} \wedge k \neq j \wedge (v_j, v_k) \in E\Big\}. \quad (1)$$

Furthermore, a DT is a data-driven classification model first induced using a training set and then applied to predict the class membership of new unclassified instances. A *training set* is a group of pre-classified instances described by a vector $a = \big(a_1, a_2, \ldots, a_d\big)$ of d *attributes* representing the variables of one problem and by a vector $c = \big(c_1, \ldots, c_s\big)$ of s class labels used to identify the membership of each instance. Each training instance is composed of a collection of attributes values and one class label. Each k-th attribute in the training set has associated a set of possible values known as its *domain* $D(a_k)$. The domain of a *categorical attribute* is a collection of unordered values and is a set of real numbers or integers for a *numerical attribute* [13].

DTs stand out for their simplicity and their high level of interpretability, and since a DTI process determines the importance of the attributes when builds the test conditions, DTs provide an embedded feature selection mechanism [14]. These characteristics along with its predictive power allow placing to DT as one of the most widely used classifiers. DTs have been applied in several domains of science and engineering such as cellular biology [15], pharmaceutical research [16], public health [17], electrical energy consumption [18], and transport studies [19], among others.

The structure of the test conditions in a DT can be used to determine its type: if a single attribute is evaluated in each test condition, it is known as a univariate DT. Since efficient induction methods such as CART [20] and C4.5 [21] generate univariate DTs, it is the most known type of DTs. Univariate DTs are also called axis-parallel DTs due to their test conditions represent axis-parallel hyperplanes dividing the instance space into several disjoint regions. On the other hand, multivariate DTs use a combination of attributes in their test conditions. Two types of multivariate DTs can be distinguished: 1) oblique DTs that use a linear combination of attributes, and 2) non-linear DTs having test conditions with non-linear combinations of attributes. Multivariate DTs commonly show better performance, and they are smaller than univariate DTs, but they are less expressive and might require more computational effort to induce them.

Most of DTI methods described in the existing literature apply a recursive partitioning strategy implementing some splitting criterion to separate the training instances. This plan is usually complemented with a pruning procedure to improve the performance of the classifier. Several studies point out that this strategy has three fundamental problems: overfitting [22], selection bias towards multi-valued attributes [23] and instability to small changes in the training set [24]. On the other hand, algorithms implementing a global search strategy can ensure an efficient exploration of the solution space although it is known that building optimal DTs is NP-Hard [25]. In particular, the implementation of MH-based approaches for DTI allows constructing DTs that are more accurate than those inducing with traditional methods due to they use intelligent search procedures combining their exploration and exploitation skills, thus providing a better

way to discover the relationships between the attributes used in the training set.

MHs have been previously applied to construct DTs, and several surveys in existing literature describe their implementations [8], [26]–[28]. Some MH-based approaches implement a recursive partitioning strategy in which an MH searches for a near-optimal test condition at each tree internal node; however, the most commonly used technique is to carry out a global search in the solution space applying an MH with the aim of finding near-optimal DTs.

Candidate solutions in MH-based approaches for DTI have been encoded either as sequences of values or as tree structures. Linear representation of candidate solutions is the encoding scheme most commonly used by genetic algorithms (GA), gene expression programming (GEP), grammatical evolution (GE), DE, and PSO, among other MHs. If this representation is used for DTI, a scheme to map the sequence of values into a DT must be applied [29], [30]. Nevertheless, since almost every MH use a fixed-length representation of its candidate solutions, a prior definition of this length is mandatory, which could affect the performance of the induced DTs. On the other hand, tree structures have been used to represent candidate solutions by MHs such as genetic programming (GP) and co-evolutionary algorithms (CEAs). In this case, a set of particular perturbation operators must be defined to generate only feasible solutions or to repair infeasible solutions [31], [32]. An advantage of applying this representation is that DTs with different sizes can be constructed, but it is known that some perturbation operators have a destructive effect on the new candidate solutions created by the MH [33].

It is important to note that to find near-optimal trees, the encoding scheme used in an MH-based approach for DTI must correctly represent the symbolic elements of a DT: test conditions and class labels. The representation schemes used by several MHs such as GA, GE, GP, and GEP are capable of encoding these elements, and they have been commonly applied for DTI implementing a global search strategy. However, this is a challenge for other MHs such as DE and PSO, which have proven to be very efficient in solving complex problems, but they have been designed to handle real-valued representations. This work describes a DE-based approach to induce univariate DTs, encouraged by the fact that DE has demonstrated to be a very competitive and successful method to solve complex problems in comparison with other MH-based approaches.

III. DIFFERENTIAL EVOLUTION

DE is an effective EA designed to solve optimization problems with real-valued parameters. DE evolves a *population* $X = \{(1, x^1), (2, x^2), \ldots, (\mathrm{NP}, x^{\mathrm{NP}})\}$ of NP *chromosomes* by applying mutation, crossover, and selection operators with the aim to reach a near-optimal solution. To build a new chromosome, instead of implementing traditional crossover and mutation operators, DE applies a linear combination of several chromosomes randomly chosen from the current population. Each chromosome in the population is encoded by one real-valued vector $x = (x_1, x_2, \ldots, x_n)$ of n parameters representing a candidate solution. The evolutionary process on DE is guided by a fitness function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ determining the quality value of each chromosome in the population.

In this paper, the standard DE algorithm [34], named DE/rand/1/bin in agreement with the nomenclature adopted to refer DE variants, is used as a procedure to find a near-optimal axis-parallel DT. DE can be considered a three-step process including an initialization phase, the evolutionary process, and the final step determining the result obtained.

The initialization phase involves the selection of a set of uniformly distributed random chromosomes from a finite search space $\Omega \subseteq \mathbb{R}^n$ to build the initial DE population, known as X_0. If for each $j \in \{1, \ldots, n\}$, $x_j^{\min}$ and $x_j^{\max}$ are the minimum and the maximum values of the j-th parameter in Ω, respectively, the j-th value of the chromosome x^i in the initial population is calculated as follows:

$$x_j^i = x_j^{\min} + r\left(x_j^{\max} - x_j^{\min}\right), \tag{2}$$

where $r \in [0, 1]$ is a uniformly distributed random number.

The evolutionary process implements an iterative scheme to evolve the initial population. At each iteration of this process, known as a *generation*, a new population of chromosomes is generated from the previous one. For each $i \in \{1, \ldots, \mathrm{NP}\}$ in the g-th generation, x^i is taken from the X_{g-1} population, and it is used to build a new vector u^i by applying the mutation and crossover operators. Vectors x^i and u^i are known as the *target vector* and the *trial vector*, respectively. These vectors are evaluated by the selection operator to update a new population X_g. In particular, the DE/rand/1/bin algorithm uses the following evolutionary operators:

- *Mutation:* Three randomly chosen candidate solutions from X_{g-1} (x^{r_1}, x^{r_2} and x^{r_3}) are linearly combined to yield a *mutated vector*, as follows:

$$v^i = x^{r_1} + \mathrm{F}\left(x^{r_2} - x^{r_3}\right), \tag{3}$$

  where F is a user-specified value representing a scale factor applied to control the differential variation.
- *Crossover:* The mutated vector is recombined with the target vector to build the trial vector. For each $j \in \{1, \ldots, n\}$, either x_j^i or v_j^i is selected based on a comparison between a uniformly distributed random number $r \in [0, 1]$ and the crossover rate CR. This operator also uses a randomly chosen index $l \in \{1, \ldots, n\}$ to ensure that u^i gets at least one parameter value from v^i, as follows:

$$u_j^i = \begin{cases} v_j^i & \text{if } r \leq \mathrm{CR} \text{ or } j = l, \\ x_j^i & \text{otherwise.} \end{cases} \tag{4}$$

- *Selection:* A one-to-one tournament is applied to determine which vector, between x^i and u^i, is selected as a member of the new population X_g.

In the final step, when a *stop condition* is fulfilled, DE returns the best chromosome in the current population. The Algorithm 1 shows the structure of the classical DE/rand/1/bin method.

Algorithm 1 Classical DE algorithm introduced in [34]

function DIFFERENTIAL EVOLUTION (CR, F, NP)

Input: The crossover rate (CR), the scale factor (F), and the population size (NP).

Output: The best chromosome in current population (x^{best}).

$g \leftarrow 0$
$X_g \leftarrow \varnothing$
for each $i \in \{1, \ldots, \text{NP}\}$ **do**
  for each $j \in \{1, \ldots, \text{n}\}$ **do**
    $x_j^i \leftarrow$ A randomly generated parameter using (2)
  end for
  $X_g \leftarrow X_g \cup \{(i, x^i)\}$
end for
while stop condition is not fullfilled **do**
  $g \leftarrow g + 1$
  $X_g \leftarrow \varnothing$
  for each $i \in \{1, \ldots, \text{NP}\}$ **do**
    $x^i \leftarrow$ Target vector from X_{g-1}
    $v^i \leftarrow$ Mutated vector generated using (3)
    $u^i \leftarrow$ Trial vector constructed using (4)
    $X_g \leftarrow X_g \cup \begin{cases} \{(i, u^i)\} & \text{if } f(u^i) \text{ is better than } f(x^i) \\ \{(i, x^i)\} & \text{otherwise} \end{cases}$
  end for
end while
$x^{\text{best}} \leftarrow$ The best chromosome in X_g
return x^{best}

DE has several advantages in comparison with other MHs such as the simplicity of its implementation, its ability to produce better results than those obtained by the others, and its low space complexity [35]. DE exhibits a good trade-off between its exploitation and exploration skills, i.e., it is more explorative at the beginning, but it is more exploitative as the evolutionary process progresses [36]. On the other hand, although DE requires the definition of a smaller number of parameters compared to other MHs, its performance is sensitive to the values selected for CR, F, and NP.

DE has been utilized to implement classification methods in conjunction with support vector machines [37], artificial neural networks [38], Bayesian classifiers [39] and instance-based classifiers [40]. In the case of its use with DTs, DE is applied in the DEMO (DE for multi-objective optimization) algorithm [41] to find the most suitable parameters so that the J48 method [42] yields more accurate and small DTs. DE also is used in the PDT (Perceptron Decision Tree) algorithm [43] to find a near-optimal oblique DT. Each chromosome in the PDT method encodes the coefficients of all possible hyperplanes of one fixed-height oblique DT.

IV. LINEAR REPRESENTATION OF CANDIDATE SOLUTIONS IN METAHEURISTICS

Linear representation of candidate solutions has been used with MH-based approaches implementing either a recursive partitioning strategy to build a DT or to perform a global search in the solution space with the aim of finding near-optimal DTs. In the first case, this representation is commonly applied to build multivariate DTs, and when it is used with MH-based algorithms implementing a global search, the sequence of values can encode both univariate and multivariate DTs.

A. RECURSIVE PARTITIONING STRATEGY TO BUILD A DECISION TREE

In this strategy, a candidate solution representing a test condition of a DT is altered with some perturbation operator to construct a new test condition. Single-solution-based MHs such as Simulated Annealing (SA), Tabu Search (TS) and the Greedy Randomized Adaptive Search Procedure (GRASP) have been used to induce DTs through a recursive partitioning strategy. SA is applied to find a near-optimal hyperplane used as test condition of an oblique DT in two approaches: altering one randomly chosen hyperplane coefficient in the SA for DT (SADT) method [44], and perturbing simultaneously several of them in the OC1-SA method [45]. Furthermore, TS also has been utilized for the same purpose in the LDTS (Linear Discriminant and TS) method [46] applying a linear discriminant analysis [47] with several subsets of attributes provided by TS. It also has been used in the Linear Discrete Support vector DT (LDSDT_{TS}) method [48] by combining a discrete support vector machine with TS. Finally, a GRASP-based method [49] finds near-optimal test conditions of an axis-parallel DT. In each iteration, instead of choosing the attribute with the maximum information gain (IG), GRASP randomly chooses one attribute from a subset of attributes with the highest IG values.

On the other hand, EAs such as Evolutionary Strategies (ES) and GA have been applied to build an oblique DT through this strategy. The OC1-ES algorithm [45] and the Multimembered ES Oblique DT (MESODT) method [50] obtain a near-optimal hyperplane using the $(1 + 1)$-ES and the (μ, λ)-ES, respectively. Furthermore, GA evolves a population of hyperplanes encoded: 1) with a binary chromosome in the Binary Tree-GA (BTGA) algorithm [51], and in the HereBoy for DT (HBDT) method [52], and 2) with a real-valued chromosome in the OC1-GA algorithm [45] and in the procedures described by Krętowski [53] and Pangilinan and Janssens [54].

B. GLOBAL SEARCH STRATEGY TO FIND NEAR-OPTIMAL DECISION TREES

Linear representation of candidate solutions also has been used to encode a DT when a global search strategy is applied. Several MH-based approaches for DTI implementing this strategy have been described in the existing literature:

Caltrop [29] is a GA-based approach evolving a population of complete DTs represented as a set of sub-trees. Each sub-tree with three nodes (a caltrop) is encoded as a sequence of integer values referring to binary attributes used as test conditions. Furthermore, in other GA-based approaches, the chromosome encodes either the nodes of a complete DT, or the elements used to build them: attributes, threshold values of the numerical attributes, and class labels.

In the first case, Cha and Tappert [55] encode both test conditions and leaf nodes in an integer-valued chromosome. Each test condition evaluates a binary attribute represented by the index of its location in an ordered list of attributes. Furthermore, in the method described by Bandar *et al.* [56], each test condition is represented by an index identifying the attribute, either categorical or numerical, used in it. In this algorithm, both the threshold values used in the internal nodes and the class labels assigned to the leaf nodes of the DT are determined by evaluating the training set. Finally, in the Evolutionary Classifier with Cost Optimization (ECCO) method [57] a binary chromosome encodes the set of test conditions of a complete DT through an index identifying each possible test condition.

On the other case, the Evolutionary Algorithm for DTI (EVO-Tree) method [30] and the work of Smith [58] implement two similar approaches. Both construct two arrays to encode the elements used by the nodes of a binary DT: one identifying both attributes and class labels, and the other storing the threshold values.

Other EA-based approaches using a linear representation of candidate solutions have been applied for DTI. A grammar to map axis-parallel DTs from binary chromosomes is defined in the Grammatical Evolution DTs (GEDT) method [59]. Furthermore, several GEP-based approaches such as the GEPDT method [60], and those described by Ferreira [61] and Wang *et al.* [62] have been implemented to yield axis-parallel DTs. In these methods the linear chromosome encodes the attributes, the threshold values and the class labels used to build a DT.

Finally, two PSO-based approaches have been implemented to find near-optimal DTs: the Tree Swarm Optimization (TSO) method [63] and a multi-objective PSO-based approach for DTI [64]. In these algorithms, a particle moving in the continuous space represents the symbols used in both test conditions and leaf nodes of a complete DT.

Except for the GEDT method [59] in which the candidate solutions are encoded using a variable-length linear chromosome, the other MH-based approaches for DTI require a prior definition of the size of the sequence of values. Almost all of them use the tree height h to define this size, either as an user-specified value [30] or as a value previously established in the algorithm [55]–[57], [63], [64]. Only the method described by Smith [58] determines the tree height in function of the number of attributes in the training set used to induce the DT ($h = d + 1$).

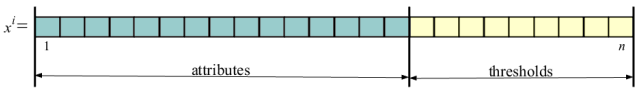

FIGURE 1. The structure of a chromosome to encode a sequence of attributes and a sequence of threshold values.

V. GLOBAL SEARCH OF AXIS-PARALLEL DECISION TREES THROUGH DIFFERENTIAL EVOLUTION

In this work, a **D**ifferential-**E**volution-based approach to build **A**xis-parallel **D**ecision **T**rees using the **SPV** rule (DE-ADT$_{\text{SPV}}$) is described. The estimated height of a binary DT whose number of internal nodes is not less than the number of attributes in the training set is used to define the size of the chromosomes in population and, although this size is based on a binary DT, the DE-ADT$_{\text{SPV}}$ method is used to build general DTs. The best DT in population is refined by replacing some leaf nodes with sub-trees to improve its accuracy. A detailed description of the DE-ADT$_{\text{SPV}}$ elements is provided in the following paragraphs.

A. LINEAR ENCODING SCHEME OF CANDIDATE SOLUTIONS

Each test condition of an axis-parallel DT evaluates only one attribute to divide the training set. If a categorical attribute is evaluated, the training set is split into as many subsets as values there are in the domain of the attribute. On the other hand, if the evaluated attribute has numerical values, a threshold value is used to split the training set into two subsets, and the DTI method must determine a suitable threshold value optimizing some splitting criterion.

The linear encoding scheme proposed in this work associates each parameter of a chromosome with each attribute and with each threshold value used in the test conditions of an axis-parallel DT. Therefore, if the vectors y^i and z^i are used to represent the sequence of attributes and the sequence of threshold values, respectively, a chromosome x^i in the population is the concatenation of y^i followed by z^i, i.e., $x^i = y^i \frown z^i$, as is shown in Fig. 1. Furthermore, if n_y and n_z are the numbers of elements in y^i and z^i, respectively, the size of x^i is $n = n_y + n_z$.

Considering that the size of a DT is related to the structure of the training set used to induce it, in the DE-ADT$_{\text{SPV}}$ method the height of a complete binary DT is estimated based on the number of attributes and the number of class labels in the training set. This height is used to compute the size of x^i. To ensure that each attribute can be evaluated at least by one test condition and that also each class label is present on at least one leaf node, d and s must be used as lower bounds for the number of internal nodes and the number of leaf nodes, respectively. Since the number of internal nodes of a complete binary DT with height h is $2^h - 1$, and the number of leaf nodes of the same DT is 2^h, two heights can be obtained as follows:

$$h_i = \lceil \log_2 (d + 1) \rceil, \tag{5}$$

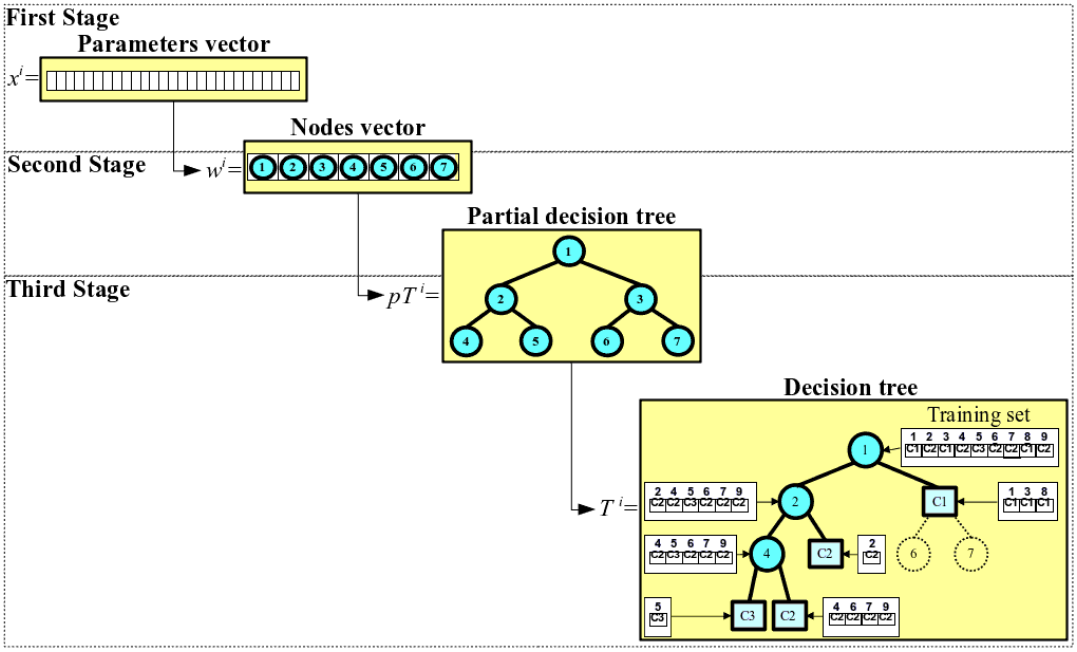

FIGURE 2. Three stages procedure to map an axis-parallel DT from x^i.

and

$$h_l = \lceil \log_2 (s) \rceil. \tag{6}$$

The estimated height of a complete binary DT in which each attribute and each class label of the training set can be used at least once is defined as $h_e = \max\{h_i, h_l\} + 1$. The DE-ADT$_{\text{SPV}}$ method applies this estimated height to calculate the size of y^i as follows:

$$n_y = 2^{h_e} - 1. \tag{7}$$

Since n_y is not less than d, each attribute in the training set is associated with one or more elements of y^i through an auxiliary vector p. For each $j \in \{1, \ldots, n_y\}$, the location of each attribute in the vector a is stored in p as follows:

$$p_j = j \bmod d. \tag{8}$$

On the other hand, as z^i identifies the threshold values of the numerical attributes associated with y^i, its size depends on the size of y^i. If the number of numerical attributes in the training set is computed as follows:

$$d_r = \left| \left\{ k \in \{1, \ldots, d\} : D(a_k) \subseteq \mathbb{R} \right\} \right|, \tag{9}$$

n_z is obtained using the following equation:

$$n_z = d_r \left\lfloor \frac{n_y}{d} \right\rfloor + \left| \left\{ k \in \{1, \ldots, d\} : D(a_k) \subseteq \mathbb{R} \wedge d \left\lfloor \frac{n_y}{d} \right\rfloor + k \leq n_y \right\} \right|. \tag{10}$$

The first term of (10) refers the amount of numerical attributes used when a is entirely associated with y^i, and the second one represents the number of these attributes used when a is partially associated with y^i.

Once the size of x^i is calculated using the structure of the training set, DE evolves a population of chromosomes using the training accuracy of the constructed DTs as their fitness values.

B. INDUCTION OF FEASIBLE AXIS-PARALLEL DECISION TREES

The DE-ADT$_{\text{SPV}}$ method implements a three-stages procedure to map an axis-parallel DT from a chromosome of the population. First, x^i is used to build the vector w^i which encodes a sequence of candidate nodes of a DT. Next, w^i is utilized to create a partial DT pT^i composed only of internal nodes. Finally, to complete the DT, a set of leaf nodes are added in pT^i using the training set. This procedure allows inducing feasible axis-parallel DTs with a different number of nodes, although they are represented using a fixed-length parameters vector. Fig. 2 shows a graphical representation of this procedure.

1) NODES VECTOR CONSTRUCTION

The DE-ADT$_{\text{SPV}}$ method uses the SPV rule to build an ordered sequence of attributes from x^i. This rule creates an integer-valued vector o^i based on the elements of y^i: the location of the lowest value in y^i is the first element of o^i, the location of the next lowest value in y^i is the second element of o^i, and so on. Formally, for each $j \in \{1, \ldots, n_y\}$, the SPV rule assigns the k-th location of y^i as the j-th element

Algorithm 2 Algorithm to Build w^i From x^i

function NVConstruction(x^i)
Input: The real-valued parameter vector (x^i).
Output: The nodes vector (w^i).

$y^i \leftarrow \left(x^i_1, x^i_2, \ldots, x^i_{n_y}\right)$
$z^i \leftarrow \left(x^i_{n_y+1}, x^i_{n_y+2}, \ldots, x^i_{n_y+n_z}\right)$
for each $j \in \{1, \ldots, n_y\}$ **do**
  $o^i_j \leftarrow j$-th minor element of y^i determined using (11)
end for
for each $j \in \{1, \ldots, n_y\}$ **do**
  $k \leftarrow o^i_j$
  if $D(a_{p_k}) \subseteq \mathbb{R}$ **then**
    $q \leftarrow$ Location in z^i of the threshold value associated with a_{p_k}, computed by (12)
    $t^i_q \leftarrow$ Threshold value of a_{p_k} obtained using (13)
    $w^i_j \leftarrow \left(a_{p_k}, t^i_q\right)$ ▷ *A node with a numerical attribute*
  else
    $w^i_j \leftarrow \left(a_{p_k}\right)$ ▷ *A node with a categorical attribute*
  end if
end for
return w^i

of o^i using the following equation:

$$o^i_j = \min\left\{k \in \{1, \ldots, n_y\} \setminus \{o^i_1, \ldots, o^i_{j-1}\} : y^i_k = \min\left\{y^i_l : l \in \{1, \ldots, n_y\} \setminus \{o^i_1, \ldots, o^i_{j-1}\}\right\}\right\}. \tag{11}$$

On the other hand, to adjust the threshold values represented by z^i so that they belong to the domains of the numerical attributes in a, another auxiliary vector t^i is constructed. If for each $j \in \{1, \ldots, n_y\}$, q is the location in z^i of the threshold value associated with the numerical attribute $a_{p_{o^i_j}}$, that is computed as follows:

$$q = d_r\left\lfloor \frac{j}{d} \right\rfloor + \left|\left\{k \in \{1, \ldots, d\} : D(a_k) \subseteq \mathbb{R} \wedge k \leq p_{o^i_j}\right\}\right|, \tag{12}$$

then t^i_q represents the threshold value of $a_{p_{o^i_j}}$, obtained applying the following equation:

$$t^i_q = \min\{D(a_{p_{o^i_j}})\} + \frac{\left(z^i_q - x^{\min}_j\right)\left(\max\{D(a_{p_{o^i_j}})\} - \min\{D(a_{p_{o^i_j}})\}\right)}{x^{\max}_j - x^{\min}_j}. \tag{13}$$

Once o^i contains the ordered locations of y^i, and t^i holds the threshold values associated with the numerical attributes encoded in y^i, these vectors are used to build the vector w^i representing the sequence of candidate internal nodes of a partial DT. For each $j \in \{1, \ldots, n_y\}$, the j-th element of w^i is:

$$w^i_j = \begin{cases} \left(a_{p_{o^i_j}}, t^i_q\right) & \text{if } D(a_{p_{o^i_j}}) \subseteq \mathbb{R}, \\ \left(a_{p_{o^i_j}}\right) & \text{otherwise.} \end{cases} \tag{14}$$

The Algorithm 2 outlines the process to build w^i from x^i. Once w^i is completed, it is used to create a partial DT with only internal nodes.

Algorithm 3 Construction of pT^i from w^i

function DTConstruction(w^i)
Input: The nodes vector (w^i).
Output: The partial DT with only feasible internal nodes (pT^i).

$V \leftarrow \{w^i_1\}$
$E \leftarrow \varnothing$
for each $j \in \{1, \ldots, n_y\}$ **do**
  if $w^i_j \in V$ **then**
    $b \leftarrow$ Number of possible successor nodes of w^i_j computed using (15)
    $k \leftarrow j + 1$
    while $k \leq n_y \wedge \left|\mathcal{N}^+(w^i_j)\right| < b$ **do**
      $\alpha \leftarrow$ Attribute assigned in w^i_k
      if α satisfies R_1 and $R_2 \wedge w^i_k$ satisfies R_3 **then**
        $V \leftarrow V \cup \{w^i_k\}$
        $E \leftarrow E \cup \{(w^i_j, w^i_k)\}$
      end if
      $k \leftarrow k + 1$
    end while
  end if
end for
$pT^i \leftarrow \left(G(V, E), w^i_1\right)$
return pT^i

2) PARTIAL DECISION TREE CONSTRUCTION

A straightforward procedure is applied to construct the partial DT from w^i: First, the element in the initial location of w^i is used as the root node of pT^i. Next, the remaining elements of 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. The number of successor nodes b of an internal node is calculated based on the domain of the attribute used in its test condition, as follows:

$$b = \begin{cases} 2 & \text{if } D(\alpha) \subseteq \mathbb{R}, \\ \left|D(\alpha)\right| & \text{otherwise,} \end{cases} \tag{15}$$

where α is the attribute assigned in w^i.

Since pT^i is constructed using the ordered sequence of elements of y^i, it is likely to contain one or more redundant nodes, i.e., nodes whose test condition does not split the instances set. To ensure that pT^i does not hold any redundant node, the following rules are applied:

Algorithm 4 Completion of a DT Using Both pT^i and the Training Set

function DTCompletion(pT^i, ι, τ)
Input: The partial DT (pT^i), the training set (ι), and the threshold value used to assign a leaf node (τ).
Output: The DT (T^i) mapped from the i-th chromosome in the population.

$\alpha \leftarrow$ Attribute asigned to the root node of pT^i
$\omega' \leftarrow (\alpha, \iota)$ ▷ *The root node with the training instances*
$V' \leftarrow \{\omega'\}$
$E' \leftarrow \varnothing$
$Q \leftarrow$ Empty queue
Enqueue(Q, ω')
while Q is not empty **do**
  $\omega \leftarrow$ Dequeue(Q)
  $\alpha \leftarrow$ Attribute asigned to ω
  $\phi \leftarrow$ Instances set assigned in ω
  $\varphi \leftarrow |\mathcal{N}^+(\omega)|$
  $b \leftarrow$ Number of possible successor nodes of ω computed using (15).
  $\Phi \leftarrow$ Classify(ϕ, α)
  for each $j \in \{1, \ldots, b\}$ **do**
    $\zeta \leftarrow$ MajorityClass(ϕ^j)
    $\psi \leftarrow$ The number of instances in ϕ^j with ζ as class label
    if $(j \leq \varphi \wedge |\phi^j| \neq \psi \wedge |\phi^j| > \tau)$ **then**
      $\alpha \leftarrow$ The attribute used by $\mathcal{N}_j^+(\omega)$
      $\omega_j \leftarrow (\alpha, \phi^j)$ ▷ *An internal node*
      Enqueue(Q, ω_j)
    else
      $\omega_j \leftarrow (\zeta, \phi^j)$ ▷ *A leaf node*
    end if
    $V' \leftarrow V' \cup \{(\omega_j)\}$
    $E' \leftarrow E' \cup \{(\omega, \omega_j)\}$
  end for
end while
$T^i \leftarrow (G(V', E'), \omega')$
return T^i

R_1: A categorical attribute can only be evaluated once in each branch of the tree.
R_2: A numerical attribute can be evaluated several times in the same branch of the tree if and only if it uses coherent threshold values.
R_3: The successor nodes of one internal node with two branches of the tree cannot use the same categorical attribute.

Therefore, when an element in w^i does not satisfy the previous rules, it is not used to create an internal node, and the procedure continues analyzing the next item in it. The Algorithm 3 shows the steps applied to create pT^i from w^i. In this algorithm can be observed that V is the set of valid internal nodes of pT^i, and E is the set of edges representing each possible outcome of each test condition in pT^i.

Algorithm 5 General Structure of DE-ADT$_{\text{SPV}}$ method

procedure DE-ADT$_{\text{SPV}}$(*trainingSet*, CR, F, NP, τ)
Input: The training set (*trainingSet*), the DE parameters (CR, F and NP), and the threshold value used to assign a leaf node (τ).

$(a, c, \iota) \leftarrow$ ReadTrainingSet(*trainingSet*)
$d \leftarrow |a|$
$s \leftarrow |c|$
$n_y \leftarrow$ Number of estimated internal nodes computed using (7)
for each $j \in \{1, \ldots, n_y\}$ **do**
  $p_j \leftarrow$ Position of an attribute in a using (8)
end for
$n_z \leftarrow$ Number of threshold values computed using (10)
$n \leftarrow n_y + n_z$
$x^{\text{best}} \leftarrow$ DifferentialEvolution(CR, F, NP)
$w \leftarrow$ NVConstruction(x^{best})
$pT \leftarrow$ DTConstruction(w)
$T \leftarrow$ DTCompletion(pT, ι, τ)
$T \leftarrow$ DTRefinement(T, τ)
$T \leftarrow$ DTPruning(T, τ)

3) DECISION TREE COMPLETION

The final stage of the mapping scheme is responsible to add leaf nodes in pT^i using the training set. In this stage, one instances set ϕ is assigned to one internal node ω (the training set for the root node), and evaluating each element in ϕ with the test condition associated to ω, several instances subsets are created and assigned to the successor nodes of ω. This assignment is repeated for each node in pT^i. Two cases should be considered:

1) If ω is located at the end of a branch of pT^i, then as many nodes are created as possible instances subsets are obtained when the elements in ϕ are evaluated, and they are designated as successor nodes of ω. Each instances subset is assigned to each created node, and each one is labeled as a leaf node using as its class label the one that has the highest number of occurrences in the instances subset assigned to it.
2) If the number of instances assigned to ω is less than one previously defined threshold value τ, or if all instances assigned to it belong to the same class, then ω is labeled as a leaf node. The majority class ζ in its instances set is assigned as the class label of the leaf node, and its successor nodes are removed, if they exist.

The algorithm 4 summarizes the process to complete the DT from pT^i. This procedure uses a first-in-first-out (FIFO) queue to assign the instances of the training set in each node of the DT. Furthermore, this algorithm uses the following methods:

- Classify(ϕ, α): This method splits ϕ using the atribute α, generating a collection Φ with as many instances subsets as possible successor nodes of ω.

- MajorityClass(ϕ^j): This method returns the class label ζ associated with the largest number of instances belonging to the same class in ϕ^j.

In this algorithm can be observed that V' is the set of both internal nodes and leaf nodes of T', each one associated with an instances subset.

C. GENERAL STRUCTURE OF THE DE-ADT$_{SPV}$ METHOD

The Algorithm 5 shows the structure of the DE-ADT$_{SPV}$ method proposed in this work. This procedure requires to identify the training set used to induce an axis-parallel DT, as well as the three control parameters applied by the DE algorithm, and the threshold value τ used to determine if a node is labeled as a leaf node.

First, the DE-ADT$_{SPV}$ method uses the ReadTrainingSet(*trainingSet*) method to get the attributes vector a, the vector of class labels c, and the instances set ι. Next, the values of d and s are computed, as well as the values of n_y, n_z, and n used to build the initial population of chromosomes. Then, the DE algorithm evolves this population to obtain the best candidate solution x^{best}. After that, a near-optimal DT is constructed by applying the three stage procedure described in the previous paragraphs. Finally, this DT is refined to replace non-optimal leaf nodes with sub-trees, as well as it is pruned to reduce the possible overfitting generated by applying this refinement.

The DE-ADT$_{SPV}$ method implements two modifications in the DE/rand/1/bin algorithm:

- *Variable Scale Factor:* The F parameter gradually decreases as the evolutionary process progresses. This decrement allows more exploration of the search space at the beginning of the evolutionary process, and with the passage of the generations, it tries to make a better exploitation of promising areas of this space [65].
- *Mixed parameters:* The parameters in a chromosome representing an axis-parallel DT can be constrained or not. Unconstrained parameters are associated with the sequence of attributes, and constrained elements represent the threshold values used to build the test conditions with numerical attributes. When a parameter value violates a constraint, it is adjusted to the midpoint between its previous value and the boundary-value of the violated constraint as follows:

$$u_j^i \leftarrow \begin{cases} \frac{1}{2}\left(x_j^i + x_j^{\max}\right) & \text{if } u_j^i > x_j^{\max}, \\ \frac{1}{2}\left(x_j^i + x_j^{\min}\right) & \text{if } u_j^i < x_j^{\min}, \\ u_j^i & \text{otherwise.} \end{cases} \tag{16}$$

  This mechanism to handle constraints allows asymptotically approach the space boundaries [66].

Since the DE-ADT$_{SPV}$ method uses an a priori definition of the size of the chromosome, it is possible that some leaf nodes in the DT do not meet the following conditions: that the size of its instances subset is less than τ, or that all the instances in the subset belong to the same class. In this case, the DE-ADT$_{SPV}$ method applies the TreeGrowing procedure to replace this node with a sub-tree whose leaf nodes fulfill these conditions. This method implements a recursively partitioning strategy guided by some splitting criterion. However, it is desirable that this refinement is used only when the estimated number of nodes n_y does not permit to build a DT with an acceptable accuracy. The Algorithm 6 shows the procedure to refine the best T^i constructed with the DE-ADT$_{SPV}$ method. In this work the Gain Ratio [21] is used for the TreeGrowing procedure as splitting criterion.

Algorithm 6 Refinement of a DT

function DTRefinement(T^i, τ)
Input: The DT (T^i) and the threshold value used to assign a leaf node (τ).
Output: The refined DT (T^i).

$\omega' \leftarrow$ Root node of T^i
$V' \leftarrow$ Nodes set of T^i
$E' \leftarrow$Edges set of T^i
$Q \leftarrow$ Empty queue
Enqueue(Q, ω')
while Q is not empty **do**
  $\omega \leftarrow$ Dequeue(Q)
  $\varphi \leftarrow \left|\mathcal{N}^+(\omega)\right|$
  if $\varphi > 0$ **then** ▷ *The node is an internal node*
    for each $j \in \{1, \ldots, \varphi\}$ **do**
      $\omega_j \leftarrow \mathcal{N}_j^+(\omega)$
      Enqueue(Q, ω_j)
    end for
  else ▷ *The node is a leaf node*
    $\phi \leftarrow$ Instances set in ω
    $\zeta \leftarrow$ MayorityClass(ϕ)
    $\psi \leftarrow$ The number of instances in ϕ with ζ as class label
    if $\left(|\phi| \neq \psi \wedge |\phi| > \tau\right)$ **then**
      $(V', E') \leftarrow (V', E') \cup$ TreeGrowing(ω, τ)
    end if
  end if
end while
$T^i \leftarrow \left(G\left(V', E'\right), \omega'\right)$
return T^i

VI. EXPERIMENTAL STUDY

In this section the experimental study carried out to analyze the performance of the DE-ADT$_{SPV}$ method is detailed. First, a description of the datasets used in this study, as well as the definition of the parameters of the DE-ADT$_{SPV}$ method are given. Then, both the model validation technique used in the experiments and the statistical tests applied to evaluate the results obtained are outlined. Finally, a discussion about the performance of the DE-ADT$_{SPV}$ method is provided.

TABLE 1. Description of datasets used in the experiments.

Dataset	Instances	Numerical attributes	Categorical attributes	Classes	Class distribution
car	1728	0	6	4	1210|384|69|65
molecular-p	106	0	57	2	53|53
tic-tac-toe	958	0	9	2	332|626
glass	214	9	0	7	70|76|17|0|13|9|29
pima-diabetes	768	8	0	2	500|268
balance-scale	625	4	0	3	288|49|288
heart-statlog	270	13	0	2	150|120
iris	150	4	0	3	50|50|50
australian	690	14	0	2	307|383
ionosphere	351	34	0	2	126|225
wine	178	13	0	3	59|71|48
sonar	208	60	0	2	97|111
vehicle	846	18	0	4	212|217|218|199
liver-disorder	345	6	0	2	145|200
page-blocks	5473	10	0	5	4913|329|28|88|115
lymph	148	3	15	4	2|81|61|4
credit-g	1000	7	13	2	700|300
cmc	1473	2	7	3	629|333|511
haberman	306	2	1	2	225|81
dermatology	366	1	33	6	112|61|72|49|52|20

A. EXPERIMENTAL SETUP

A benchmark of 20 datasets chosen from the UCI machine learning repository [67] is used to carry out the experimental study. These datasets have been selected as they attributes are numerical, categorical, or a combination of them, also their instances are classified into two or more classes, and most of them are imbalanced datasets. Table 1 shows the description of these datasets. To ensure that the comparison of the results achieved by the DE-ADT$_{\text{SPV}}$ method with those produced by other approaches is not affected by the treatment of the data, all datasets used in this study do not have missing values.

The DE-ADT$_{\text{SPV}}$ method is implemented in the Java language using the JMetal library [68]. The parameters used in the experiments are described in Table 2. The mutation scale factor is linearly decreased from 0.5 to 0.1 as the evolutionary process progresses, and the crossover rate is fixed in 0.9. Furthermore, following the suggestion of Storn and Price [34], the population size is adjusted to $5n$, with 250 and 500 chromosomes as lower and upper bound, respectively. These bounds are used to ensure that the population is not so small as not to allow a reasonable exploration of the search space and it is not so large as to impact the runtime of the algorithm. The fitness function used in the DE-ADT$_{\text{SPV}}$ method computes the training accuracy of each DT in population.

On the other hand, since the best DT obtained by the evolutionary process is refined with a procedure implementing a recursive partitioning strategy, it must be pruned to reduce the possible overfitting generated by applying this refinement. In the DE-ADT$_{\text{SPV}}$ method, the Error-Based Pruning (EBP) approach [21] is implemented since it produces DTs with an improved accuracy using only the training set [69]. Finally, the DE-ADT$_{\text{SPV}}$ method needs to define a threshold value to determine whether a node should be labeled as one leaf node.

To obtain reliable estimates of the predictive performance of the DE-ADT$_{\text{SPV}}$ method and to compare its results with

TABLE 2. Parameters used in the experiments conducted with the DE-ADT$_{\text{SPV}}$ method.

Parameter	Value
Mutation scale factor	$[0.1, 0.5]$
Crossover rate	0.9
Population size	$250 \leq 5n \leq 500$
Number of generations	200
Fitness value	Training accuracy
Pruning method	Error-based pruning
Threshold value used to label leaf nodes	2 instances

those got by other supervised learning approaches, a repeated stratified 10-fold cross-validation (CV) procedure is applied in this experimental study. In a 10-fold CV, the training set is randomly divided into ten roughly equal disjoint folds. For each $k \in \{1, \ldots, 10\}$, the k-th fold is retained (the test set), and the remaining folds are used to induce a DT. Once the DT has been constructed, the retained fold is used to calculate its test accuracy. Finally, when all folds have been used in the induction phase, the overall test accuracy of the model is computed. In particular, in a stratified CV the proportion of the different classes in each fold must be very similar to those in the complete dataset, and in a repeated CV several runs of the CV process are conducted, and the average test accuracy of these runs is used as the final estimated yield of the model.

According to the previous paragraph, the DE-ADT$_{\text{SPV}}$ method is run for each iteration of the 10-fold CV procedure. Since the evolutionary process in the DE-ADT$_{\text{SPV}}$ method uses the training accuracy of each DT as its fitness value, the DTs in the final population are overfitted to the training set, so the DT with the best training accuracy would have a decreased test accuracy. In this work, with the aim of mitigating the effects of this overfitting, a subset of instances of the dataset is used to determine an independent accuracy for each DT in the final population and to select the best one. This value is referred in this work as the *selection accuracy*, so the DT with the best selection accuracy in the final population is used to calculate the test accuracy of the fold. To implement this strategy, 20% of the instances in the dataset are used to compute the selection accuracy, and the remaining are used in the CV procedure. Fig. 3 depicts this cross-validation scheme.

The CV procedure applied to estimate the test accuracy of the classifier constructed by the DE-ADT$_{\text{SPV}}$ method is similar to the one proposed by Murthy *et al.* [70]: For each fold, the selection accuracy of each DT in the population is calculated, and the DT with the best selection accuracy is used to compute the number of test instances correctly classified. The ratio between the correct classifications of all folds and the number of training instances is taken as the overall test accuracy of the classifier. Furthermore, the DT size is defined as the average number of leaf nodes of the DTs constructed by all folds.

In this study, the Friedman test [71] is applied to carry out a statistical analysis of the results produced by the DE-ADT$_{\text{SPV}}$ method when comparing them with those obtained by other classification methods. This non-

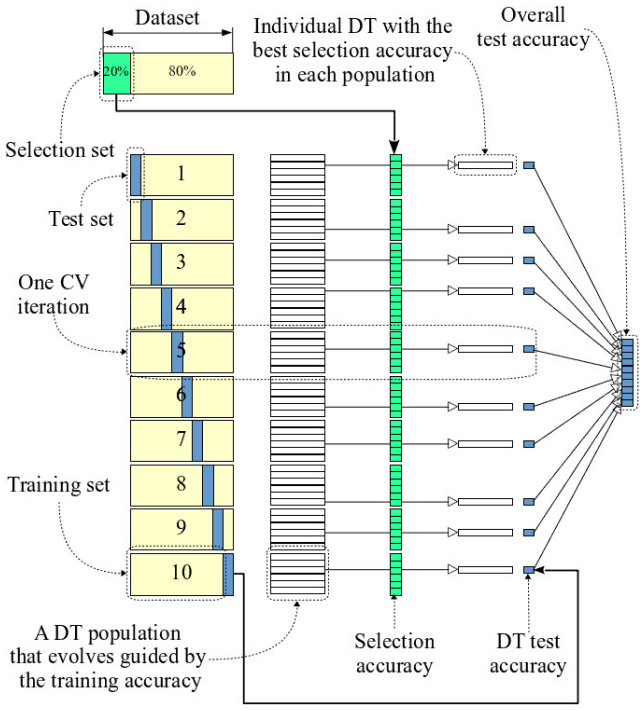

FIGURE 3. Adapted cross-validation procedure to determine the overall test accuracy of each dataset in the experimental study.

parametric statistical test evaluates the statistical significance of the experimental results through computing the p-value without making any assumptions about the distribution of the analyzed data. This p-value is used to accept or to reject the null hypothesis H_0 of the experiment which holds that the performance of the compared algorithms does not present significant differences. If the p-value does not exceed a predefined significance level, H_0 is rejected and the Bergmann-Hommel (BH) post-hoc test [72] is conducted to detect the differences between all existing pairs of algorithms. These statistical tests are applied using the *scmamp* R library [73].

B. METHODOLOGY APPLIED TO ANALYZE THE PERFORMANCE OF THE DE-ADT$_{SPV}$ METHOD

Two DE-ADT$_{SPV}$ variants are evaluated in this experimental study:

- DE-ADT$^{B}_{SPV}$: This is the first variant of the method which returns the DT with the best selection accuracy in the population, without applying the refinement of the non-optimal leaf nodes.
- DE-ADT$^{R}_{SPV}$: This variant returns the refined version of the DT with the best selection accuracy in the population.

The results obtained with the DE-ADT$_{SPV}$ variants are compared with those achieved by several supervised learning methods available on the WEKA data mining software [74]. First, the accuracy and size of the DTs gotten by these variants are compared with those obtained by the following DTI methods:

- J48 [42]: It is a Java implementation of the C4.5 algorithm.

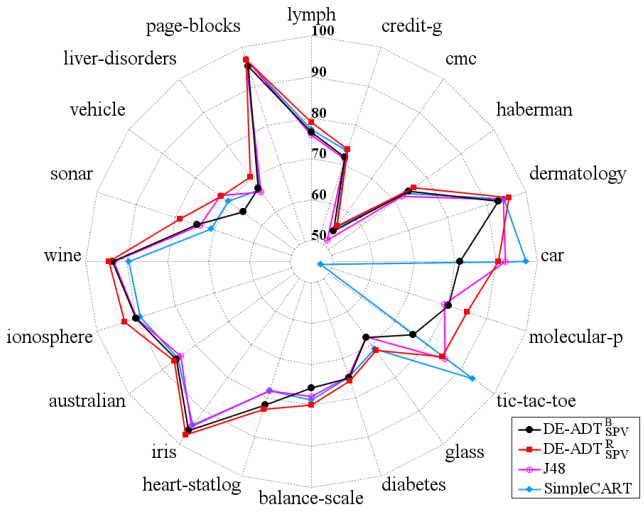

FIGURE 4. Graphical comparison of the average accuracies obtained by the DTI methods.

- sCART (SimpleCART) [20]: This is a Java implementation of the CART method.

Next, the accuracy of the DTs constructed with the DE-ADT$_{SPV}$ variants are compared with those achieved using the following classification methods:

- NB (Naïve Bayes) [75]: This is a probabilistic classifier based on the Bayes theorem.
- MLP (Multilayer Perceptron) [76]: MLP is a feed-forward artificial neural network (FF-ANN) applying backpropagation to classify instances. The MLP has one or more hidden layers of nodes using sigmoid functions.
- RBF-NN (Radial Basis Function Neural Network) [77]: This is also an FF-ANN using a set of Gaussian radial basis functions (RBF) in its hidden layer.
- RF (Random Forest) [78]: It is an ensemble learning method constructing a multitude of DTs. RF uses a voting scheme to predict the class membership of new unclassified instances.

Finally, both height and size of the induced DT with the DE-ADT$_{SPV}$ variants are analyzed to evaluate the advantages of implementing the proposed scheme. The number of refinements of non-optimal leaf nodes is also assessed, due to the desire that the number of branches inserted in the evolved DT be reduced.

C. RESULTS

In this section, the results of the DE-ADT$_{SPV}$ variants are described and compared with those got by other classification methods.

1) COMPARISON WITH DTI METHODS

In Table 3 and Fig. 4 are shown the average accuracies of the DTs induced by the DTI methods as well as those achieved by the DE-ADT$_{SPV}$ variants. In Table 3, the best result for each

TABLE 3. Average accuracies obtained by the DTI methods.

Dataset	J48	sCART	DE-ADT$^{B}_{SPV}$	DE-ADT$^{R}_{SPV}$
car	92.22 (2)	**97.37** (1)	81.19 (4)	90.53 (3)
molecular-p	79.06 (3)	47.17 (4)	80.12 (2)	**84.88** (1)
tic-tac-toe	85.28 (2)	**93.57** (1)	75.49 (4)	84.46 (3)
glass	67.62 (4)	71.26 (2)	67.82 (3)	**71.67** (1)
diabetes	74.49 (4)	74.56 (3)	74.73 (2)	**75.48** (1)
balance-scale	77.82 (3)	78.74 (2)	75.70 (4)	**80.00** (1)
heart-statlog	78.15 (3)	78.07 (4)	81.81 (2)	**82.92** (1)
iris	94.73 (3)	94.20 (4)	95.92 (2)	**96.95** (1)
australian	84.35 (4)	85.19 (3)	85.57 (2)	**86.42** (1)
ionosphere	89.74 (3)	88.86 (4)	89.97 (2)	**92.88** (1)
wine	93.20 (3)	89.49 (4)	93.47 (2)	**94.37** (1)
sonar	73.61 (3)	70.67 (4)	74.19 (2)	**78.74** (1)
vehicle	**72.28** (1)	69.91 (3)	65.49 (4)	72.21 (2)
liver-disorders	65.83 (4)	66.64 (3)	66.96 (2)	**70.33** (1)
page-blocks	**96.99** (1)	96.76 (2)	95.10 (4)	96.74 (3)
lymph	75.81 (4)	77.16 (2)	76.50 (3)	**78.83** (1)
credit-g	71.25 (4)	73.43 (2)	71.63 (3)	**73.65** (1)
cmc	51.44 (4)	55.21 (2)	54.18 (3)	**55.57** (1)
haberman	72.16 (4)	73.24 (3)	74.29 (2)	**75.76** (1)
dermatology	94.10 (3)	94.43 (2)	92.81 (4)	**95.70** (1)
Average ranking	3.1	2.75	2.8	1.35

TABLE 4. p-values for multiple comparisons among DTI methods.

Method	AR	DE-ADT$^{B}_{SPV}$		DE-ADT$^{R}_{SPV}$	
		Unadjusted	BH	Unadjusted	BH
J48	3.10	4.6243e-01	1.0000e+00	1.8143e-05	**1.0885e-04**
sCART	2.75	9.0252e-01	1.0000e+00	6.0517e-04	**1.2103e-03**
DE-ADT$_{SPV}$	2.80	-	-	3.8266e-04	**1.1480e-03**
DE-ADT$^{R}_{SPV}$	1.35	3.8266e-04	**1.1480e-03**	-	-

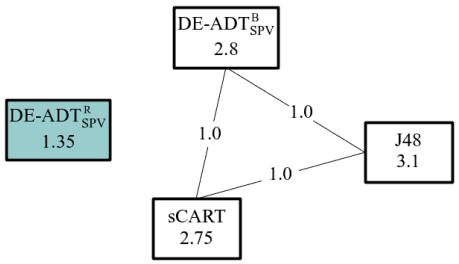

FIGURE 5. p-values graph of the DTI methods.

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 this table indicates the average ranking of each method. It is observed that the DE-ADT$_{SPV}$ variants produce better results than those generated by the other DTI methods. In particular, the DE-ADT$^{R}_{SPV}$ variant obtains the best results from this experiment, as it yields higher average accuracies than those got by the compared DTI techniques in 16 datasets.

A statistical test of the experimental results is conducted to evaluate the performance of the DE-ADT$_{SPV}$ variants. First, the Friedman test is run and its resulting statistic value is 22.02 for four methods and 20 datasets, which has a p-value of 6.461×10^{-5}. When evaluating this p-value with a significance level of 5%, H_0 is rejected. Next, the BH

TABLE 5. Average DT sizes of several DTI methods.

Dataset	J48	sCART	DE-ADT$^{B}_{SPV}$	DE-ADT$^{R}_{SPV}$
car	122.05 (4)	58.00 (2)	**15.57** (1)	105.33 (3)
molecular-p	16.90 (4)	**1.00** (1)	11.29 (2)	11.62 (3)
tic-tac-toe	88.04 (4)	31.00 (2)	**27.12** (1)	79.80 (3)
glass	23.58 (4)	**8.00** (1)	8.88 (2)	15.93 (3)
diabetes	22.20 (4)	**3.00** (1)	6.71 (2)	18.55 (3)
balance-scale	41.60 (4)	13.00 (2)	**10.04** (1)	18.43 (3)
heart-statlog	17.82 (4)	16.00 (3)	**9.60** (1)	9.77 (2)
iris	4.64 (3)	5.00 (4)	**4.10** (1)	4.35 (2)
australian	25.75 (4)	**5.00** (1)	7.47 (2)	13.54 (3)
ionosphere	13.87 (3)	**3.00** (1)	7.86 (3)	7.79 (2)
wine	5.30 (4)	**5.00** (1)	5.39 (4)	5.25 (2)
sonar	14.45 (4)	**10.00** (1)	10.34 (2)	10.40 (3)
vehicle	69.50 (3)	80.00 (4)	**11.50** (1)	61.84 (2)
liver-disorders	25.51 (4)	**3.00** (1)	7.58 (2)	14.29 (3)
page-blocks	42.91 (4)	22.00 (2)	**8.04** (1)	40.45 (3)
lymph	17.30 (4)	**9.00** (1)	13.01 (2)	13.14 (3)
credit-g	90.18 (4)	**7.00** (1)	35.67 (2)	55.57 (3)
cmc	149.75 (4)	18.00 (2)	**15.61** (1)	38.87 (3)
haberman	15.32 (4)	**3.00** (1)	10.16 (3)	9.09 (2)
dermatology	27.06 (4)	**9.00** (1)	19.14 (2)	24.07 (3)
Average ranking	3.85	1.65	1.8	2.7

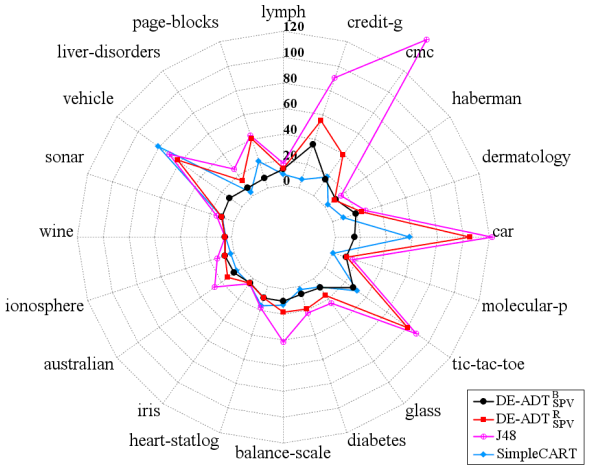

FIGURE 6. Graphical comparison of the average DT sizes obtained by the DTI methods.

post-hoc test is applied to find all the possible hypotheses which cannot be rejected. In Table 4 is shown both the average rank (AR) of the results yielded by each method and the p-values computed by comparing the average accuracies achieved by the DE-ADT$_{SPV}$ variants versus those obtained by the other DTI methods. The p-values highlighted with bold numbers indicate that H_0 is rejected for this pair of methods since they show different performance. Unadjusted p-values are calculated with the average ranks of the two methods being compared, as is described by Demšar [79]. These values are used by the BH post-hoc test to compute the corresponding adjusted p-values. Table 4 shows that the DE-ADT$^{R}_{SPV}$ method has a better performance than the other DTI methods since it has the lowest average rank (1.35) and its results are statistically different than the others.

Fig. 5 shows a graph where the nodes represent the compared methods and the edges joining two nodes indicate that

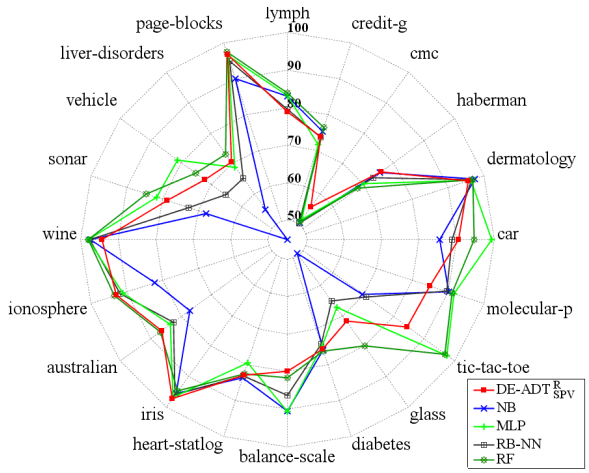

FIGURE 7. Graphical comparison of the average accuracies obtained by several classification methods.

TABLE 6. Average accuracies obtained by several classification methods.

Dataset	NB	MLP	RBF-NN	RF	DE-ADT$^{R}_{SPV}$
car	85.46 (5)	**99.41** (1)	88.80 (4)	94.63 (2)	90.53 (3)
molecular-p	90.19 (3)	**91.70** (1)	89.53 (4)	91.13 (2)	84.88 (5)
tic-tac-toe	69.65 (5)	**97.39** (1)	70.88 (4)	96.93 (2)	84.46 (3)
glass	49.44 (5)	67.29 (3)	65.09 (4)	**79.95** (1)	71.67 (2)
diabetes	75.76 (2)	74.75 (4)	74.04 (5)	**76.18** (1)	75.48 (3)
balance-scale	90.53 (2)	**90.69** (1)	86.34 (3)	81.71 (4)	80.00 (5)
heart-statlog	**83.59** (1)	79.41 (5)	83.11 (2)	82.41 (4)	82.92 (3)
iris	95.53 (4)	96.93 (2)	96.00 (3)	94.73 (5)	**96.95** (1)
australian	77.19 (5)	83.42 (3)	82.55 (4)	**86.77** (1)	86.42 (2)
ionosphere	82.17 (5)	91.05 (4)	91.71 (3)	**93.39** (1)	92.88 (2)
wine	97.47 (4)	**98.03** (1.5)	97.70 (3)	**98.03** (1.5)	94.37 (5)
sonar	67.69 (5)	81.59 (2)	72.60 (4)	**84.47** (1)	78.74 (3)
vehicle	44.68 (5)	**81.11** (1)	65.35 (4)	75.14 (2)	72.21 (3)
liver-disorders	54.87 (5)	68.72 (3)	65.04 (4)	**72.99** (1)	70.33 (2)
page-blocks	90.01 (5)	96.28 (3)	94.91 (4)	**97.54** (1)	96.74 (2)
lymph	83.11 (3)	83.24 (2)	79.66 (4)	**83.92** (1)	78.83 (5)
credit-g	75.16 (2)	71.58 (5)	73.58 (4)	**76.27** (1)	73.65 (3)
cmc	50.48 (4)	51.53 (2)	50.19 (5)	50.63 (3)	**55.57** (1)
haberman	75.36 (2)	70.29 (4)	73.14 (3)	68.17 (5)	**75.76** (1)
dermatology	**97.40** (1)	96.45 (4)	96.58 (3)	96.86 (2)	95.70 (5)
Average ranking	3.65	2.625	3.7	2.075	2.95

TABLE 7. p-values for multiple comparisons among several classification methods.

Method	AR	DE-ADT$^{R}_{SPV}$ Unadjusted	DE-ADT$^{R}_{SPV}$ BH_T
NB	3.650	1.6151e-01	5.3445e-01
MLP	2.625	5.1563e-01	1.0000e+00
RBF-NN	3.700	1.3361e-01	5.3445e-01
RF	2.075	8.0118e-02	3.2047e-01
DE-ADT$^{R}_{SPV}$	2.950	-	-

the performance of these methods does not present significant differences. The values shown in the edges are the p-values computed by the BH post-hoc test. This figure is based on that obtained using the *scmamp* library, and in it is observed that the DE-ADT$^{B}_{SPV}$ variant, the J48 algorithm and the sCART procedure are not statistically different.

On the other hand, the average sizes of the DTs constructed by the DE-ADT$_{SPV}$ variants and also of those induced by

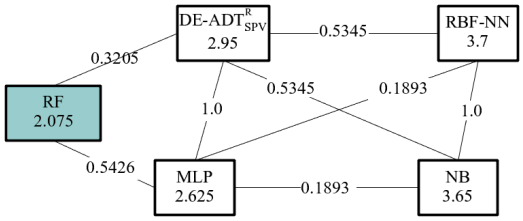

FIGURE 8. p-values graph of the classifiction methods.

TABLE 8. Average DT height of DE-ADT variants.

Dataset	Predefined DT height	DE-ADT$^{B}_{SPV}$	DE-ADT$^{R}_{SPV}$
car	4	4.00	**6.00**
molecular-p	7	3.50	3.57
tic-tac-toe	5	5.00	**6.84**
glass	5	5.45	**7.73**
diabetes	5	5.25	**9.61**
balance-scale	4	5.00	**6.74**
heart-statlog	5	5.68	5.70
iris	4	3.99	4.01
australian	5	5.75	**7.49**
ionosphere	7	6.16	6.76
wine	5	4.02	4.09
sonar	7	5.83	6.15
vehicle	6	6.02	**20.04**
liver-disorders	4	4.79	**7.25**
page-blocks	5	5.25	**13.08**
lymph	6	4.89	4.89
credit-g	6	4.98	**7.24**
cmc	5	5.35	**6.50**
haberman	3	3.36	3.50
dermatology	7	6.00	7.15

J48 and sCART methods are shown in Table 5 and Fig. 6. These results indicate that the sCART method produces the most compact DTs but to the detriment of their accuracies. Also, it is observed that the size of the DTs built for the DE-ADT$_{SPV}$ variants has less complexity than the size of the DTs yielded by the J48 method. As the DE-ADT$^{R}_{SPV}$ variant applies a recursive partition strategy to refine the best DT generated by the evolutionary process, the average sizes of its constructed DTs are similar to the sizes of the DTs induced by the J48 method, although they are always smaller than the latter.

2) COMPARISON WITH OTHER CLASSIFICATION METHODS

In Table 6 and Fig. 7 are shown the average accuracies got by several classification methods as well as those obtained by the DE-ADT$^{R}_{SPV}$ method. In this Table can be observed that the RF algorithm and the MLP method construct more accurate classifiers than the others, and also that the DE-ADT$^{R}_{SPV}$ induces DTs with better accuracy than the models built by both the RBF-NN algorithm and the NB method.

The Friedman statistics computed by analyzing the results got by these five methods with 20 datasets is 10.4, and the corresponding p-value is 0.034222 so that H_0 is rejected. The BH post-hoc test is then applied to find all possible hypotheses that can not be refused. Table 7 shows the results of these tests, and Fig. 8 shows the graph corresponding to these p-values.

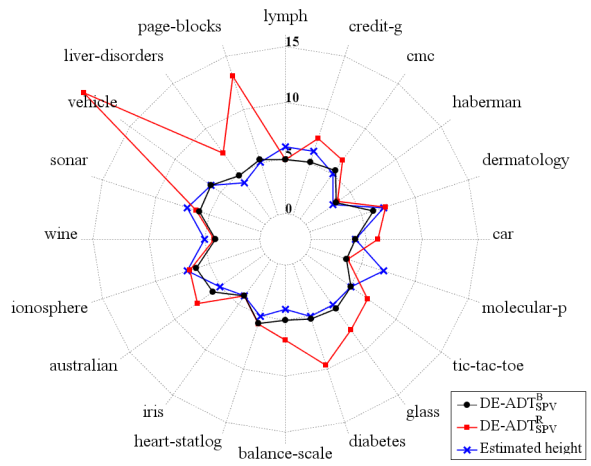

FIGURE 9. Graphical comparison of the average DT heigths obtained by the DE-ADT$_{SPV}$ variants.

TABLE 9. Refinements percentages of the DTs constructed with the DE-ADT$^R_{SPV}$ variant.

Dataset	Leaf nodes of the best DT in the population	Leaf nodes of the refined DT	Number of refinements	% of refinements
car	15.92	105.33	12.92	**81.16**
molecular-p	11.26	11.62	0.35	3.11
tic-tac-toe	25.30	79.80	15.48	**61.19**
glass	9.10	15.93	3.14	**34.51**
diabetes	6.44	18.55	2.11	**32.76**
balance-scale	10.11	18.43	7.24	**71.61**
heart-statlog	9.03	9.77	0.86	9.52
iris	4.35	4.35	0.00	0.00
australian	7.03	13.54	2.27	**32.29**
ionosphere	6.00	7.79	1.55	25.83
wine	5.18	5.25	0.11	2.12
sonar	7.87	10.40	2.41	30.62
vehicle	10.40	61.84	7.42	**71.35**
liver-disorders	7.68	14.29	2.28	**29.69**
page-blocks	7.83	40.45	6.85	**87.48**
lymph	12.91	13.14	0.96	7.44
credit-g	34.41	55.57	7.85	**22.81**
cmc	14.96	38.87	2.80	**18.72**
haberman	8.85	9.09	0.46	5.20
dermatology	18.46	24.07	3.69	19.99

The p-values obtained by the BH post-hoc test point out that the RF method is statistically different only with the RBF-NN algorithm and the NB method, and the comparison between the remaining pairs of algorithms indicates that they have a similar performance. The RF method is the best ranked in this comparison, and the AR of the DE-ADT$^R_{SPV}$ variant places it as the third best classification method.

3) PREDEFINED HEIGHT AND REFINEMENT RATE

For evaluating the relevance of using the dataset information to define the size of chromosomes evolving in the DE-ADT$_{SPV}$ method, both the average heights of the constructed DTs and the number of refinements applied to the best DT in the last population of the evolutionary process are analyzed. Table 8 and Fig. 9 show the average heights produced by the DE-ADT$_{SPV}$ variants, and also the predefined height computed before to apply the evolutionary process. In Table 9 is shown the refinements percentages of the DTs constructed by the DE-ADT$^R_{SPV}$ variant, where the number of refinements represents the non-optimal leaf nodes of the best DT in the final population of the DE algorithm, which are replaced with several sub-trees.

In Table 8 is observed that the average heights of the DTs constructed are less than the predefined height in six datasets, and they surpass it in two or more levels in nine datasets. Two characteristics persist in the datasets with deep DTs: they have more than 600 training instances and more than two class labels. When the refinement percentage is analyzed, it is observed that this value is higher than 25% for these datasets.

VII. CONCLUSIONS

In this paper, the DE-ADT$_{SPV}$ method implementing a global search strategy to find near-optimal axis-parallel DTs represented using real-valued chromosomes is introduced. This method estimates the size of the chromosomes based on the characteristics of the dataset whose model is constructed and applies a scheme to map feasible DTs from them. In light of the experimental results, it can be affirmed that the DE-ADT$^R_{SPV}$ method is an efficient DTI procedure since it constructs DTs with high accuracy and a smaller number of leaf nodes. The refinement applied in the best DT in the final population permits to improve the training accuracy of the model. Notwithstanding the results yielded by the DE-ADT$_{SPV}$ method are not better than those produced by the RF algorithm and the MLP-based classifier, they are statistically equivalent. An advantage of the DE-ADT$^R_{SPV}$ variant is that it constructs models whose decisions and operations are easily understood, and although the RF method also builds DTs, its voting scheme makes it very difficult to trace the way in which the model takes its decisions. DE algorithm is an effective approach for constructing axis-parallel DTs when a rule to map a DT from a real-valued chromosome is implemented. An advantage of this approach is that the DE operators can be applied without any modification, and the chromosomes in population represent only feasible DTs.

In this paper, an analysis of the run-time of the algorithms is not performed, since it is known that MHs consume more computational time than other approaches because they work with a group of candidate solutions, unlike the traditional methods where only one DT is induced from the training set. It is important to mention that for many practical applications, the construction of the model is conducted in one offline procedure, so the time of its construction is not a parameter that usually impacts the efficiency of the built model.

Since DE algorithm is one of the most powerful MH for solving both real-valued and discrete optimization problems, in this work it is applied for inducing axis-parallel DTs. Furthermore, although a considerable number of heuristic algorithms and classification approaches have been described in the existing literature, in this work a global search strategy to find a classifier that is compact, accurate, robust, and interpretable is implemented. A DT induced by this method can be used as a predictive model in disciplines such as biology, medicine, and finances, among others in

which the interpretability level of the model has a crucial importance.

Finally, it is interesting to analyze the implementation of alternative techniques to map a DT from a real-valued vector and evaluate its impact on the built model. It is also important to conduct a study on the benefit of including the number of training instances to compute the chromosome size and consider other ideas to refine the DT resulting from the evolutionary process.

REFERENCES

[1] A. McGovern *et al.*, "Using artificial intelligence to improve real-time decision making for high-impact weather," *Bull. Amer. Meteorol. Soc.*, vol. 98, no. 10, pp. 2073–2090, 2017, doi: 10.1175/BAMS-D-16-0123.1.
[2] E. V. Sekar, J. Anuradha, A. Arya, B. Balusamy, and V. Chang, "A framework for smart traffic management using hybrid clustering techniques," *Cluster Comput.*, pp. 1–12, May 2017.
[3] J. B. Sathe and M. P. Mali, "A hybrid sentiment classification method using neural network and fuzzy logic," in *Proc. 11th Int. Conf. Intell. Syst. Control (ISCO)*, Coimbatore, India, Jan. 2017, pp. 93–96.
[4] A. B. Hens and M. K. Tiwari, "Computational time reduction for credit scoring: An integrated approach based on support vector machine and stratified sampling method," *Expert Syst. Appl.*, vol. 39, no. 8, pp. 6774–6781, 2012.
[5] J. Huysmans, K. Dejaeger, C. Mues, J. Vanthienen, and B. Baesens, "An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models," *Decision Support Syst.*, vol. 51, no. 1, pp. 141–154, 2011.
[6] D. F. Nettleton, A. Orriols-Puig, and A. Fornells, "A study of the effect of different types of noise on the precision of supervised learning techniques," *Artif. Intell. Rev.*, vol. 33, no. 4, pp. 275–306, 2010.
[7] T. Grubinger, A. Zeileis, and K. P. Pfeiffer, "evtree: Evolutionary learning of globally optimal classification and regression trees in R," *J. Statist. Softw.*, vol. 61, no. 1, pp. 1–29, 2014.
[8] R. C. Barros, M. P. Basgalupp, A. C. P. L. F. de Carvalho, and A. A. Freitas, "A survey of evolutionary algorithms for decision-tree induction," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 3, pp. 291–312, May 2012.
[9] M. F. Tasgetiren, M. Sevkli, Y.-C. Liang, and G. Gencyilmaz, "Particle swarm optimization algorithm for single machine total weighted tardiness problem," in *Proc. Congr. Evol. Comput. (CEC)*. Portland, OR, USA, Jun. 2004, pp. 1412–1419.
[10] B. Niu, F. Zhang, L. Li, and L. Wu, "Particle swarm optimization for yard truck scheduling in container terminal with a cooperative strategy," in *Proc. 20th Asia–Pacific Symp. Intell. Evol. Syst. (IES)*, vol. 8. 2017, pp. 333–346.
[11] M. Tasgetiren, Y. Liang, M. Sevkli, and G. Gencyilmaz, "Differential evolution algorithm for permutation flowshop sequencing problem with makespan criterion," in *Proc. 4th Int. Symp. Intell. Manuf. Syst. (IMS)*, Sakarya, Turkey, 2004, pp. 442–452.
[12] L. Rokach and O. Maimon, "Top-down induction of decision trees classifiers—A survey," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 35, no. 4, pp. 476–487, Nov. 2005.
[13] S. Morishita, "On classification and regression," in *Proc. 1st Int. Conf. Discovery Sci. (DS)*, Fukuoka, Japan, 1998, pp. 40–57, doi: 10.1007/3-540-49292-5_4.
[14] T. Lal, O. Chapelle, J. Weston, and A. Elisseeff, "Embedded methods," in *Feature Extraction: Foundations and Applications* (Studies in Fuzziness and Soft Computing), vol. 207, I. Guyon, M. Nikravesh, S. Gunn, and L. A. Zadeh, Eds. Berlin, Germany: Springer, 2006, pp. 137–165, doi: 10.1007/978-3-540-35488-8_6.
[15] C. Suenderhauf, F. Hammann, and J. Huwyler, "Computational prediction of blood-brain barrier permeability using decision tree induction," *Molecules*, vol. 17, no. 9, pp. 10429–10445, 2012.
[16] P. E. Blower and K. P. Cross, "Decision tree methods in pharmaceutical research," *Current Topics Med. Chem.*, vol. 6, no. 1, pp. 31–39, 2006.
[17] Z. Xin *et al.*, "Reanalysis and external validation of a decision tree model for detecting unrecognized diabetes in rural chinese individuals," *Int. J. Endocrinol.*, vol. 2017, Art. no. 3894870, May 2017.
[18] G. K. F. Tso and K. K. W. Yau, "Predicting electricity energy consumption: A comparison of regression analysis, decision tree and neural networks," *Energy*, vol. 32, no. 9, pp. 1761–1768, 2007.
[19] M. Hu, Y. Liao, W. Wang, G. Li, B. Cheng, and F. Chen, "Decision tree-based maneuver prediction for driver rear-end risk-avoidance behaviors in cut-in scenarios," *J. Adv. Transp.*, vol. 2017, Art. no. 7170358, Feb. 2017.
[20] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. London, U.K.: Chapman & Hall, 1984.
[21] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA, USA: Morgan Kaufmann, 1993.
[22] D. M. Hawkins, "The problem of overfitting," *J. Chem. Inf. Comput. Sci.*, vol. 44, no. 1, pp. 1–12, May 2004.
[23] T. Hothorn, K. Hornik, and A. Zeileis, "Unbiased recursive partitioning: A conditional inference framework," *J. Comput. Graph. Statist.*, vol. 15, no. 3, pp. 651–674, 2006.
[24] C. Strobl, J. Malley, and G. Tutz, "An introduction to recursive partitioning: Rationale, application, and characteristics of classification and regression trees, bagging, and random forests," *Psychol. Methods*, vol. 14, no. 4, pp. 323–348, 2009.
[25] L. Hyafil and R. L. Rivest, "Constructing optimal binary decision trees is NP-complete," *Inf. Process. Lett.*, vol. 5, no. 1, pp. 15–17, 1976.
[26] P. G. Espejo, S. Ventura, and F. Herrera, "A survey on the application of genetic programming to classification," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 40, no. 2, pp. 121–144, Mar. 2010.
[27] P. Kokol, S. Pohorec, G. Štiglic, and V. Podgorelec, "Evolutionary design of decision trees for medical application," *Data Mining Knowl. Discovery*, vol. 2, no. 3, pp. 237–254, 2012.
[28] E. Kolçe and N. Frasheri, "The use of heuristics in decision tree learning optimization," *Int. J. Comput. Eng. Res. Trends*, vol. 1, no. 3, pp. 127–130, 2014.
[29] H. Kennedy, C. Chinniah, P. V. G. Bradbeer, and L. Morss, "The construction and evaluation of decision trees: A comparison of evolutionary and concept learning methods," in *Proc. AISB Int. Workshop Evol. Comput.*, Manchester, U.K., 1997, pp. 147–162, doi: 10.1007/BFb0027172.
[30] D. Jankowski and K. Jackowski, "Evolutionary algorithm for decision tree induction," in *Proc. 13th IFIP TC8 Int. Conf. Comput. Inf. Syst. Ind. Manag. (CISIM)*, Ho Chi Minh City, Vietnam, 2014, pp. 23–32, doi: 10.1007/978-3-662-45237-0_4.
[31] M. Krętowski and M. Grześ, "Mixed decision trees: An evolutionary approach," in *Proc. 8th Int. Conf. Data Warehousing Knowl. Discovery (DaWaK)*, Krakow, Poland, 2006, pp. 260–269, doi: 10.1007/11823728_25.
[32] M. P. Basgalupp, R. C. Barros, A. C. P. L. F. de Carvalho, and A. A. Freitas, "Evolving decision trees with beam search-based initialization and lexicographic multi-objective evaluation," *Inf. Sci.*, vol. 258, pp. 160–181, Feb. 2014.
[33] A. A. Freitas, *Data Mining and Knowledge Discovery With Evolutionary Algorithms*. Secaucus, NJ, USA: Springer, 2002, doi: 10.1007/978-3-662-04923-5.
[34] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.
[35] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.
[36] F. Neri and V. Tirronen, "Recent advances in differential evolution: A survey and experimental analysis," *Artif. Intell. Rev.*, vol. 33, nos. 1–2, pp. 61–106, 2010.
[37] J. Li, L. Ding, and B. Li, "Differential evolution-based parameters optimisation and feature selection for support vector machine," *Int. J. Comput. Sci. Eng.*, vol. 13, no. 4, pp. 355–363, 2016.
[38] N. Leema, H. Nehemiah, and A. Kannan, "Neural network classifier optimization using differential evolution with global information and back propagation algorithm for clinical datasets," *Appl. Soft Comput.*, vol. 49, pp. 834–844, Dec. 2016.
[39] K. Geetha and S. S. Baboo, "An empirical model for thyroid disease classification using evolutionary multivariate Bayseian prediction method," *Global J. Comput. Sci. Technol.*, vol. 16, no. 1, pp. 1–9, 2016.
[40] S. García, J. Derrac, I. Triguero, C. J. Carmona, and F. Herrera, "Evolutionary-based selection of generalized instances for imbalanced classification," *Knowl.-Based Syst.*, vol. 25, no. 1, pp. 3–12, 2012.
[41] T. Tušar, "Optimizing accuracy and size of decision trees," in *Proc. 16th Int. Electrotech. Comput. Sci. Conf. (ERK)*, Portorož, Slovenia, 2007, pp. 81–84.
[42] I. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S. Cunningham, "Weka: Practical machine learning tools and techniques with Java implementations," Dept. Comput. Sci., Univ. Waikato, Hamilton, New Zealand, Tech. Rep. 11, 1999.

[43] R. A. Lopes, A. R. R. Freitas, R. C. P. Silva, and F. G. Guimarães, "Differential evolution and perceptron decision trees for classification tasks," in *Proc. 13th Int. Conf. Intell. Data Eng. Auto. Learn. (IDEAL)*, Natal, Brazil, 2012, pp. 550–557, doi: 10.1007/978-3-642-32639-4_67.
[44] D. G. Heath, S. Kasif, and S. Salzberg, "Induction of oblique decision trees," in *Proc. 13th Int. Joint Conf. Artif. Intell. (IJCAI)*, Chambéry, France, 1993, pp. 1002–1007.
[45] E. Cantú-Paz and C. Kamath, "Inducing oblique decision trees with evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, no. 1, pp. 54–68, Feb. 2003.
[46] X.-B. Li, J. R. Sweigart, J. T. C. Teng, J. M. Donohue, L. A. Thombs, and S. M. Wang, "Multivariate decision trees using linear discriminants and tabu search," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 33, no. 2, pp. 194–205, Mar. 2003.
[47] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.
[48] C. Orsenigo and C. Vercellis, "Discrete support vector decision trees via tabu search," *Comput. Statist. Data Anal.*, vol. 47, no. 2, pp. 311–322, 2004.
[49] J. Pacheco, E. Alfaro, S. Casado, M. Gámez, and N. García, "A GRASP method for building classification trees," *Expert Syst. Appl.*, vol. 39, no. 3, pp. 3241–3248, 2012.
[50] K. Zhang, Z. Xu, and B. P. Buckles, "Oblique decision tree induction using multimembered evolution strategies," *Proc. SPIE*, vol. 5812, pp. 263–270, Mar. 2005.
[51] B.-B. Chai, X. Zhuang, Y. Zhao, and J. Sklansky, "Binary linear decision tree with genetic algorithm," in *Proc. 13th Int. Conf. Pattern Recognit. (ICPR)*, vol. 4. Aug. 1996, pp. 530–534.
[52] R. Struharik, V. Vranjković, S. Dautović, and L. Novak, "Inducing oblique decision trees," in *Proc. 12th Int. Symp. Intell. Syst. Inform. (SISY)*, Subotica, Serbia, Sep. 2014, pp. 257–262.
[53] M. Kretowski, "An evolutionary algorithm for oblique decision tree induction," in *Proc. 7th Int. Conf. Artif. Intell. Soft Comput. (ICAISC)*, Zakopane, Poland, 2004, pp. 432–437, doi: 10.1007/978-3-540-24844-6_63.
[54] J. M. Pangilinan and G. K. Janssens, "Pareto-optimality of oblique decision trees from evolutionary algorithms," *J. Global Optim.*, vol. 51, no. 2, pp. 301–311, 2011.
[55] S.-H. Cha and C. Tappert, "Constructing binary decision trees using genetic algorithms," in *Proc. Int. Conf. Genetic Evol. Methods (GEM)*, 2008, pp. 49–54.
[56] Z. Bandar, H. Al-Attar, and D. McLean, "Genetic algorithm based multiple decision tree induction," in *Proc. 6th Int. Conf. Neural Inf. Process. (ICONIP)*, vol. 2. Nov. 1999, pp. 429–434.
[57] A. Omielan and S. Vadera, "ECCO: A new evolutionary classifier with cost optimisation," in *Proc. 7th Int. Conf. Intell. Inf. Process. VI (IIP)*, vol. 385. 2012, pp. 97–105.
[58] S. F. Smith, "RNA search acceleration with genetic algorithm generated decision trees," in *Proc. 7th Int. Conf. Mach. Learn. Appl. (ICMLA)*, San Diego, CA, USA, Dec. 2008, pp. 565–570.
[59] A. A. Motsinger-Reif, S. Deodhar, S. J. Winham, and N. E. Hardison, "Grammatical evolution decision trees for detecting gene-gene interactions," *BioData Mining*, vol. 3, no. 1, p. 8, 2010.
[60] L. Qu, Y. Min, W. Weihong, and C. Xiaohong, "Dynamic split-point selection method for decision tree evolved by gene expression programming," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Trondheim, Norway, May 2009, pp. 736–740.
[61] C. Ferreira, "Decision tree induction," in *Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence*. Secaucus, NJ, USA: Springer, 2006, pp. 337–380, doi: 10.1007/3-540-32849-1_9.
[62] W. Wang, Q. Li, S. Han, and H. Lin, "A preliminary study on constructing decision tree with gene expression programming," in *Proc. 1st Int. Conf. Innov. Comput., Inf. Control (ICICIC)*, vol. 1. Beijing, China, 2006, pp. 222–225.
[63] C. Veenhuis, M. Koppen, J. Kruger, and B. Nickolay, "Tree swarm optimization: An approach to PSO-based tree discovery," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2. Scotland, U.K., Sep. 2005, pp. 1238–1245.
[64] J. E. Fieldsend, "Optimizing decision trees using multi-objective particle swarm optimization," in *Swarm Intelligence for Multi-objective Problems in Data Mining* (SCI), vol. 242, C. A. Coello-Coello, S. Dehuri, and S. Ghosh, Eds. Berlin, Germany: Springer, 2009, pp. 93–114, doi: 10.1007/978-3-642-03625-5_5.
[65] S. Das, A. Konar, and U. Chakraborty, "Two improved differential evolution schemes for faster global search," in *Proc. 7th Annu. Conf. Genetic Evol. Comput. (GECCO)*, Washington, DC, USA, 2005, pp. 991–998.
[66] G. C. Onwubolu and D. Davendra, "Scheduling flow shops using differential evolution algorithm," *Eur. J. Oper. Res.*, vol. 171, no. 2, pp. 674–692, 2006.
[67] M. Lichman, "UCI Machine Learning Repository," School Inf. Comput. Sci., Univ. California, Irvine, CA, USA, 2013. [Online]. Available: http://archive.ics.uci.edu/ml
[68] J. J. Durillo and A. J. Nebro, "jMetal: A Java framework for multi-objective optimization," *Adv. Eng. Softw.*, vol. 42, no. 10, pp. 760–771, 2011.
[69] L. A. Breslow and D. W. Aha, "Simplifying decision trees: A survey," *Knowl. Eng. Rev.*, vol. 12, no. 1, pp. 1–40, 1997.
[70] S. K. Murthy, S. Kasif, and S. Salzberg, "A system for induction of oblique decision trees," *J. Artif. Intell. Res.*, vol. 2, no. 1, pp. 1–32, 1994.
[71] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *J. Amer. Statist. Assoc.*, vol. 32, no. 200, pp. 675–701, 1937.
[72] G. Hommel, "A stagewise rejective multiple test procedure based on a modified Bonferroni test," *Biometrika*, vol. 75, no. 2, pp. 383–386, 1988.
[73] B. Calvo and G. Santafé, "scmamp: Statistical comparison of multiple algorithms in multiple problems," *R J.*, vol. 8, no. 1, pp. 248–256, 2016.
[74] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *ACM SIGKDD Explorations Newslett.*, vol. 11, no. 1, pp. 10–18, 2009.
[75] G. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," in *Proc. 11th Conf. Uncertainty Artif. Intell. (UAI)*, San Francisco, CA, USA, 1995, pp. 338–345.
[76] F. Murtagh, "Multilayer perceptrons for classification and regression," *Neurocomputing*, vol. 2, nos. 5–6, pp. 183–197, 1991.
[77] E. Frank, "Fully supervised training of Gaussian radial basis function networks in WEKA," Dept. Comput. Sci., Univ. Waikato, Hamilton, New Zealand, Tech. Rep. 04, 2014.
[78] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[79] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.

RAFAEL RIVERA-LOPEZ was born in Poza Rica, México, in 1965. He received the B.S. degree in computer systems engineering from the Instituto Tecnológico de Veracruz, Veracruz, México, in 1989, and the M.S. degree in computer sciences from the Instituto Tecnológico y de Estudios Superiores de Monterrey, Cuernavaca, México, in 2000. He is currently pursuing the Ph.D. degree in computer sciences with the Universidad Juárez Autónoma de Tabasco, Cunduacán, México.

From 1991 to 1998, he was a Systems Analyst and a Software Developer with several companies in Mexico. Since 1992, he has been a Research Professor with the Computer Systems Department, Instituto Tecnológico de Veracruz, México. His research interests include the study and application of metaheuristics for solving complex problems and the implementation of object-oriented models in machine learning procedures.

JUANA CANUL-REICH received the Ph.D. degree in computer science and engineering from the University of South Florida in 2010. She is currently a Faculty Member with the Academic Division of Informatics and Systems, Universidad Juárez Autónoma de Tabasco, México, where she is a Graduate Advisor of master's and Ph.D. students. Her research interests include data mining and machine learning with focus on microarray data analysis, medical data analysis, seismic data analysis, and performance evaluation of algorithms. She is mainly interested in problems of dimensionality reduction, classification, data exploration, preprocessing, and visualization. In 2015, she received the membership of the National System of Researchers level 1, México. She is an active Reviewer of the *Journal of Ambient Intelligence and Smart Environments*, for *Sinectica-a Mexican Journal in Education*, and for diverse International conferences.