Swarm and Evolutionary Computation 69 (2022) 101006

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Survey Paper

# Induction of decision trees as classification models through metaheuristics

Rafael Rivera-Lopez[a], Juana Canul-Reich[b,*], Efrén Mezura-Montes[c], Marco Antonio Cruz-Chávez[d]

[a] *Departamento de Sistemas y Computación, Tecnológico Nacional de México. Instituto Tecnológico de Veracruz. M. A. de Quevedo 2779, Col. Formando Hogar, 91800, Veracruz, VER, México*
[b] *División Académica de Ciencias y Tecnologías de la Información, Universidad Juárez Autónoma de Tabasco Km. 1 Carretera Cunduacán-Jalpa de Méndez, Col. La Esmeralda, 86690, Cunduacán, TAB, México*
[c] *Centro de Investigación en Inteligencia Artificial, Universidad Veracruzana Sebastián Camacho 5, Centro, 91000, Xalapa-Enriquez, VER, México*
[d] *Centro de Investigación en Ingeniería y Ciencias Aplicadas, Universidad Autónoma del Estado de Morelos Av. Universidad 1001, Col. Chamilpa, 62209, Cuernavaca, MOR, México*

A R T I C L E   I N F O

*Keywords:*
Machine learning
Single-solution-based metaheuristics
Evolutionary algorithms
Swarm intelligence methods

A B S T R A C T

The induction of decision trees is a widely-used approach to build classification models that guarantee high performance and expressiveness. Since a recursive-partitioning strategy guided for some splitting criterion is commonly used to induce these classifiers, overfitting, attribute selection bias, and instability to small training set changes are well-known problems in them. Other approaches, such as incremental induction, classifier ensembles, and the global search in the decision-tree-space, have been implemented to overcome these problems. In particular, metaheuristics such as simulated annealing, genetic algorithms, genetic programming, and ant colony optimization have been used to induce compact and accurate decision trees. This paper presents a state-of-the-art review of the use of single-solution-based metaheuristics and swarm and evolutionary computation algorithms to build decision trees as classification models. We outline the decision-tree-induction process components and detail the existing literature studies on metaheuristic-based approaches to building these classifiers. Several timelines showing the chronological order in which these approaches were introduced in the literature are included. A summary analysis of these studies is also conducted, focusing on their internal components and experimental studies. This work provides a useful reference point for future research in this field.

## 1. Introduction

Machine learning is an exciting artificial intelligence area whose objective is that an artificial entity becomes able to improve its performance (it learns) from the previously obtained results (its experience). Machine learning techniques have gained importance over the past few decades due to the growing demand for data analysis in diverse disciplines such as data science, business intelligence, and big data. The most representative machine learning approaches are supervised learning (a model is learned from labeled data) and unsupervised learning (a model is obtained from unlabeled data). The main supervised techniques are classification and regression, while clustering is the most used unsupervised technique.

*Han* et al. [1] point out that data classification is a two-step process (Fig. 1) where a classification model is first built (learning step) and then is used to predict the class membership of new unclassified instances (classification step). The learning step uses a *training set* with several pre-classified instances. Each instance is composed of a collection of attribute values and one label identifying its class membership. In the classification step, the model performance is typically evaluated with its predictive accuracy, which is computed using the *test set*. Other criteria such as size, speed, robustness, scalability, and others can also be applied [2].

*Abbreviations:* AF, Aggregating fitness function; AP, Axis-parallel; B&B, Branch-and-bound; BNF, Backus-Naur form; CART, Classification and Regression Trees; CV, Cross-validation; DT, Decision tree; DTI, Decision tree induction; EA, Evolutionary algorithm; FF, Fitness function; GS, Global search; HH, Hyper-heuristic; LDA, Lineal discriminant analysis; MEP, Multi-expression programming; MF, Multi-objective fitness function; MH, Metaheuristic; MO, Multi-objective; MSMT, Multisurface method-tree; NL, Non-linear; OB, Oblique; PGM, Probabilistic graphical model; RH&R, Ramped half-and-half; RP, Recursive partitioning; RPART, Recursive and partition trees; S-Exprs, S-Expressions; SF, Soft; SI, Swarm intelligence; SO, Subsequent optimization; SPV, Smallest-position-value; SS, Single-solution; UCI, University of California in Irving; UF, Single-objective fitness function.

* Corresponding author.
*E-mail addresses:* rafael.rl@veracruz.tecnm.mx (R. Rivera-Lopez), juana.canul@ujat.mx (J. Canul-Reich), emezura@uv.mx (E. Mezura-Montes), mcruz@uaem.mx (M.A. Cruz-Chávez).

https://doi.org/10.1016/j.swevo.2021.101006
Received 29 April 2021; Received in revised form 6 September 2021; Accepted 14 October 2021
Available online 23 October 2021
2210-6502/© 2021 Elsevier B.V. All rights reserved.

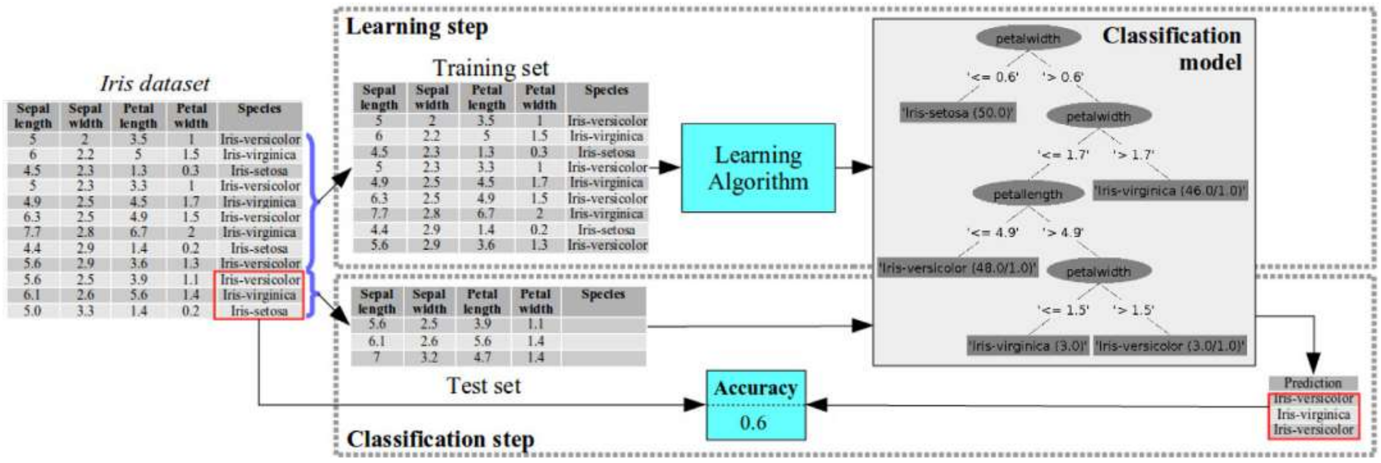

**Fig. 1.** The learning and classification steps in a classification process.

Among the most widely used classification methods are Bayesian classifiers, decision trees, artificial neural networks, and kernel-based algorithms. In particular, decision trees (DTs) are characterized by their simplicity in construction and interpretability, making them one of the most used classification methods. However, the vast majority of decision-tree-induction (DTI) procedures described in the existing literature implement a recursive-partitioning strategy through a greedy heuristic that fails to achieve desirable performance for some problems [3].

An effective strategy to search in large and complex solution spaces involves applying some swarm or evolutionary computation method. In particular, since genetic programming is an evolutionary algorithm using a tree representation to its candidate solutions, it is commonly used to find near-optimal DTs [4–6]. Furthermore, since genetic algorithms encode candidate solutions as sequences of values, their application for DTI is associated with the challenge of mapping a DT from a linear chromosome [7,8]. However, with the implementation of specialized genetic operators, they can use a tree encoding scheme [9,10]. Alternatively, there are few swarm-intelligence-based methods and other metaheuristics (MHs) for DTI. Among them, the use of simulated annealing [11], and ant-colony-optimization-based algorithms [12,13] stands out.

This paper presents a state-of-the-art review and a summary analysis of MH-based approaches for DTI. This review describes the studies concerning the construction of DTs using MHs and provides several timelines showing the chronological order in which these studies were published. Furthermore, the comparative analysis is conducted with a focus on their internal components and experimental studies. The rest of this document is organized as follows: Section 2 outlines the elements commonly considered when a DTI method is implemented and describes the main drawbacks of greedy heuristics for DTI. Furthermore, an overview of single-solution-based MHs and swarm and evolutionary computation methods is presented in Section 3, along with a description of (1) their implementation strategies, (2) their algorithm components (fitness function types, encoding schemes of candidates solutions, variation operators, and initialization procedures), and (3) their experimental studies elements (performance measures, sampling methods, and statistical tests). The state-of-the-art review of the use of single-solution-based MHs and swarm and evolutionary computation algorithms to build DTs is presented in Sections 4, 5, and 6, respectively, and the use of hyper-heuristic-based approaches to make DTI methods is depicted in Section 7. Section 8 develops a comparative analysis of the algorithms components and the elements considered in their experimental studies. A general summary of the classification of the methods described in this review is also presented. Finally, Section 9 holds the open questions and conclusions of this review.

## 2. Decision tree induction

A DT is a white-box classification model representing its decisions through a tree-like structure composed of a set of nodes containing *test conditions* (internal nodes) and *class labels* (leaf nodes). Nodes are joined by arcs, symbolizing the possible outcomes of each test condition. DTs stand out for their simplicity and high interpretability level. Since the DTI process determines the importance of the attribute when builds test conditions, it provides a built-in feature selection mechanism [14]. These characteristics, along with its predictive power, allow placing DT as one of the most widely used classifiers.

The DTI procedure quality affects both the performance and expressiveness of induced DTs. This procedure involves (1) the splitting criterion measuring test condition quality, (2) the scheme to deal with numerical and categorical attributes, and (3) the mechanism to handle missing values. Furthermore, tree pruning techniques are also used to remove overfitting tree branches, trying to improve the induced tree's predictive power. Several studies have been conducted describing, analyzing, categorizing, and comparing DTI techniques such as those of *Mingers* [15], *Safavian & Landgrebe* [16], *Brodley & Utgoff* [17], *Esposito* et al. [18], *Breslow & Aha* [19], *Murthy* [20], *Rokach & Maimon* [21], *Kotsiantis* [22], *Lomax & Vadera* [23], *Loh* [24], *Barros* et al. [25], and *Krętowski* [26], among others.

### 2.1. Types of decision trees

According to the number of attributes evaluated in each test condition, two DT types can be induced: univariate and multivariate DTs. In a univariate DT, each test condition evaluates a single attribute to split the training set. Two advantages of univariate DTs are their comprehensibility and the simplicity of their induction algorithms. However, when the training instance distribution is complex, induced DTs include many internal nodes. Alternatively, a combination of attributes is used in each multivariate DT test condition. These classifiers commonly show better performance, and they are smaller than univariate DTs. Nevertheless, they are less expressive and require more computational effort to induce them.

Univariate DTs are also known as axis-parallel-DTs since their test conditions represent axis-parallel (AP) hyperplanes dividing the instance-space, similar to those shown in Fig. 2(a). If the test condition includes a numerical attribute, a hyperplane is defined as Eqn. (1) .

$$x_i \leq c, \quad (1)$$

where $x_i$ is the $i$-th attribute value, and $c$ is a threshold value used to define the partition.

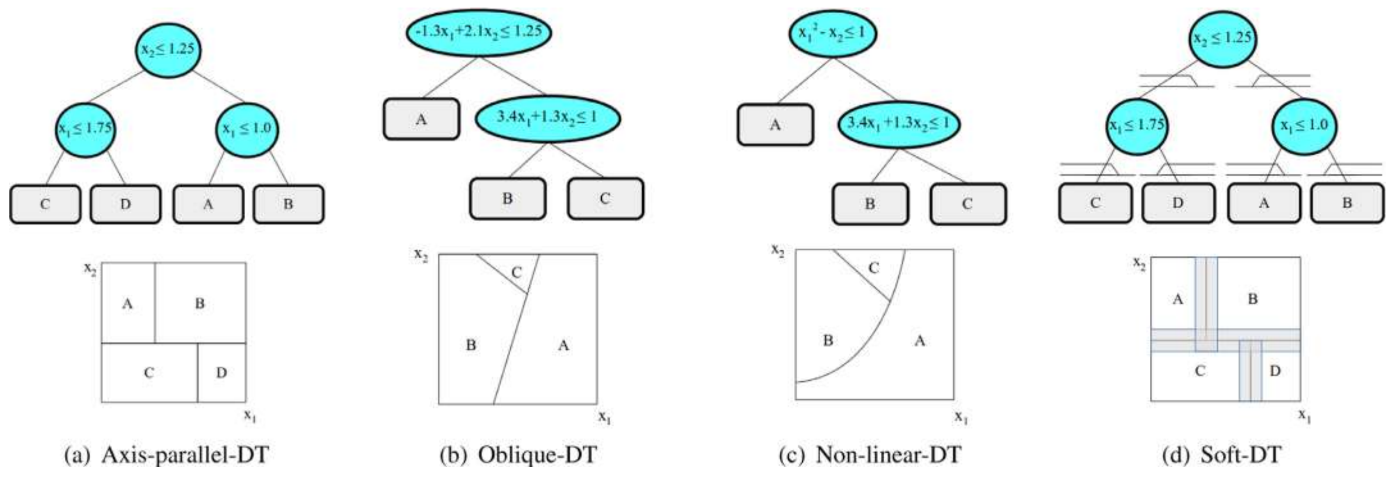

**Fig. 2.** Decision tree types (Adapted from [27] and [28]).

When a categorical attribute is evaluated, the following criteria are applied:

- *Multi-branching:* as many branches are created as values exist in the attribute domain,
- *Binary-branching:* the values of the attribute are grouped into two subsets,
- *Numerical mapping:* each categorical value is mapped as an integer value, and the attribute is treated as a numerical one, and
- *Binary mapping:* The categorical attribute is transformed into as many binary attributes as values exist in the attribute domain.

In the case of a linear combination of attributes, DTs are named oblique-DTs as their test conditions represent hyperplanes having an oblique (OB) orientation relative to the instance-space axes. An oblique hyperplane is defined as Eqn. (2).

$$\sum_{i=1}^{d} w_i x_i \le \theta \tag{2}$$

where $w_i$ is a real-valued coefficient corresponding to the $i$-th attribute value $x_i$ of a training set with $d$ attributes, and $\theta$ is the independent term. Fig. 2(b) shows an example of one oblique-DT. Similarly, non-linear-DTs produce curved hypersurfaces as they use a non-linear (NL) combination of attributes. For example, the test condition used in the DT's root-node shown in Fig. 2(c) represents a quadratic curve.

Moreover, *Wang* et al. [29] argue that uncertainty such as fuzziness and ambiguity should be incorporated into the DTI process. Soft-DTs implement a soft (SF) test at each internal node representing the probability that a branch of the node is selected based on the evaluation of its test condition. The soft-DT in Fig. 2(d) depicts a fuzzy-DT.

### 2.2. Splitting criteria

The splitting criterion applied to measure test condition quality is perhaps the element having the most significant impact on one classifier's effectiveness and expressiveness. These criteria can measure the partition impurity, estimate some other discriminant value or evaluate some cost value. Besides, considering the presence of uncertainty and ambiguity in the information, a soft measure should be included in a splitting criterion.

A typical way of grouping the vast number of splitting criteria found in the existing literature is as information-theory-based, distance-based, and other splitting criteria. Table 1 shows a representative set of these criteria.

Details of these splitting criteria are discussed in several surveys such as those of *Safavian & Landgrebe* [16], *Murthy* [20], *Rokach & Maimon* [21], *Lomax & Vadera* [23], *Lee* et al. [40], and *Barros* et al. [25], among others.

### 2.3. Tree-pruning approaches

*Kotsiantis* [22] indicates that tree pruning permits to generalize the previously induced DTs by removing nodes and subtrees, avoiding overfitting, and improving the DT comprehensibility level. Cost-complexity pruning [34], reduced-error pruning [41], pessimistic-error pruning [41], and error-based pruning [30], are considered the most typical pruning methods. Detailed studies on tree-pruning have been carried out by *Mingers* [15], *Reed* [42], *Esposito* et al. [18], and *Breslow & Aha* [19], among others.

### 2.4. Recursive-partitioning problems

Most DTI methods described in the existing literature apply a recursive-partitioning strategy implementing some splitting criterion to separate the training instances. This plan is typically complemented by a pruning procedure, trying to improve the classifier performance. CART (Classification and Regression Trees) [34] and C4.5 [30] are the most common DTI algorithms implementing this induction strategy. However, several studies point out that recursive partitioning has three fundamental problems: overfitting, selection bias toward multi-valued attributes, and instability to small changes in the training set.

A DT suffers overfitting when its classification performance is lower than its learning performance and is more complex than necessary. *Mitra & Acharya* [43] indicate that this problem occurs when data contain noise or irrelevant attributes or a small training set exists. Additionally, several studies have shown that some splitting criteria are biased in choosing an attribute type over others, even though this selection affects DT performance [44,45]. For example, the information gain criterion and the Gini index are biased in favor of multi-valued attributes [46]. Finally, *Strobl* et al. [47] argued that the main problem of DTI methods is their instability to small changes in the training set. They remark that in recursive-partitioning, the exact position of the cutpoints and selection of the splitting attribute strongly depends on the particular training instances distribution.

The incremental induction of trees [48], using classifier ensembles such as bagging, boosting, and random forest [49], as well as applying modern mixed-integer optimization techniques [50] are alternatives to avoiding these problems. Also, algorithms implementing a global-search strategy can ensure efficient solution-space exploration, although it is known that building optimal DTs is NP-Hard [51]. In particular, MH-based approaches for DTI allow creating more accurate DTs than those induced with traditional methods. MHs use intelligent search procedures

**Table 1**
Splitting criteria used for inducing decision trees.

| Acronym | Description |
|---|---|
| *Information-theory-based splitting criteria:* | |
| GR | Gain ratio [30] |
| IG | Information gain [31] |
| *Distance-based splitting criteria:* | |
| ClusterS | Cluster separation measure [32] |
| Dipolar | Mixed and pure dipoles (A dipole is a pair of training instances represented as vectors) [33] |
| Gini | Gini index [34] |
| LinearS | Degree of linear separability [35] |
| MarginS | Margin of separation [36] |
| Twoing | Twoing rule [34] |
| *Other splitting criteria:* | |
| MaxM | Max minority [37] |
| MDL | Minimum description length [38] |
| SSRE | Sum of square-root error [39] |
| SumM | Sum minority [37] |
| SumV | Sum of variances [37] |

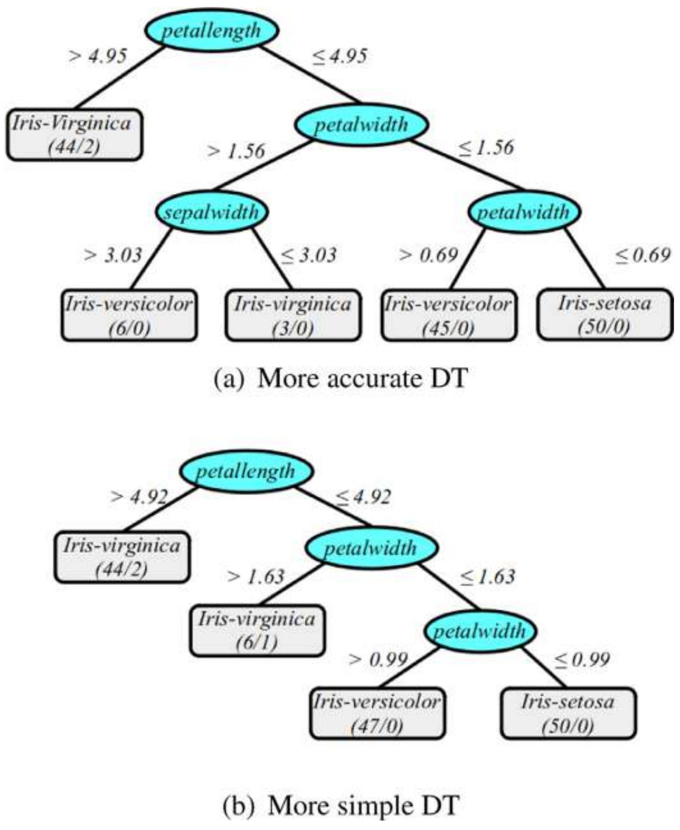

**Fig. 3.** Two DTs induced by the DE-ADT$^{SPV}$ method [8].

combining their exploration and exploitation skills, thus providing a better way of discovering relationships between training set attributes. Fig. 3 shows two DTs induced from the well-known iris dataset with an MH-based approach for DTI [8]. The first is more accurate, and the second is more compact than the DT induced by the J48 algorithm [52].

## 3. Metaheuristics for DTI

MHs are algorithmic templates simulating intelligent processes and behavior observed in both nature and other disciplines. They can be easily adapted to solve any complex problem [53,54]. MHs are characterized by combining the exploration of the solution space to identify promising areas with the exploitation of these areas to improve the known solution or solutions. They might provide or not provide optimal solutions. However, MHs are typically largely satisfactory, in contrast to other techniques failing to solve the problem or spending excessive time finding the best solution.

**Table 2**
MHs used to induce decision trees.

| Acronym | Description |
|---|---|
| *Single-solution-based MHs:* | |
| GRASP | Greedy Randomized Adaptive Search Procedure [57] |
| SA | Simulated Annealing [58] |
| SLS | Stochastic Local Search [59] |
| TS | Tabu Search [60] |
| VNS | Variable Neighborhood Search [61] |
| *Evolutionary Algorithms:* | |
| CEA | Co-Evolutionary Algorithms [62,63] |
| DE | Differential Evolution [64] |
| EDA | Estimation of Distribution Algorithm [65] |
| ES | Evolution Strategies [66,67] |
| GA | Genetic Algorithms [68] |
| GE | Grammatical Evolution [69] |
| GEP | Gene Expression Programming [70] |
| GGP | Grammar-based Genetic Programming [71] |
| GP | Genetic Programming [72] |
| TGP | Strongly-typed Genetic Programming [73] |
| *Swarm Intelligence Methods:* | |
| ACO | Ant Colony Optimization [74] |
| BA | Bat Algorithm [75] |
| PSO | Particle Swarm Optimization [76] |

*Talbi* [55] classifies these procedures as single-solution-based (SS-based) and population-based MHs. SS-based MHs conduct an intelligent search that iteratively replaces a solution with a neighboring one to reach a near-optimal solution. In contrast, population-based MHs use a group of candidate solutions in each step of their iterative process. Some generate new solutions by recombining information from the current population, and others update the solution properties. The most commonly used population-based MHs are related to evolutionary and swarm intelligence algorithms. Table 2 enumerates the MHs used to induce DTs.

Evolutionary algorithms (EAs) are inspired by theories synthesizing Darwinian evolution with Mendelian genetic inheritance. In each iteration of its evolutionary process (known as a generation), a group of candidate solutions (individuals) evolves by applying selection, recombination, and mutation operators. New populations are created until a stop condition is reached, then the best solution in the last population is returned. Commonly, two individuals (parents) are selected from the current population, and their values (genes) are recombined and typically mutated, creating new solutions (offsprings). Offsprings are considered part of the new generation.

Swarm intelligence (SI) methods are inspired by the collective behavior of some groups of animals such as ants, bees, or birds. *Vicsek & Zafeiris* [56] indicate that the main feature of collective behavior is that others' influence dominates the individual action. In SI methods, a group of candidate solutions (particles, agents) is moved in the solution-space by updating their properties, combining the local and global information of swarm transmitted by some communication type.

Finally, *Du* et al. [54] point out that a hyper-heuristic (HH) is a search procedure implemented to build algorithms that efficiently solve complex search problems. HHs explore one algorithm set to solve a problem instead of searching in the solution space.

Several surveys and reviews describing the implementation of MH-based approaches for DTI have been previously published. *Galea* et al. [77] analyze different EA-based strategies to automated fuzzy knowledge acquisition through DTs and classification rules. *Espejo* et al. [78] survey the existing literature on genetic-programming-based approaches generating DTs, classification rules, and discriminant functions. *Jabeen & Baig* [79] review several genetic-programming-based classification algorithms using decision trees, neural networks, and other rule induction methods. *Kokol* et al. [80] describe several EA-based approaches for DTI with a focus on their application in medical domains. *Barros* et al. [28] provide a detailed description of EAs inducing classification and regression trees. Also, they describe EAs implementing pruning methods and handling cost-sensitive mechanisms. *Kolçe & Frasheri* [81] summarize some approaches using genetic-algorithms, simulated annealing, and tabu search to inducing DTs. Finally, *Krętowski* [26] describes several EA-based DTI approaches, grouping them in evolutionary split searching, evolutionary meta-learning, and global evolution, *Kozak* [82] conducts a review of ant-colony-optimization-based algorithms with emphasis on decision rules, clustering, and decision trees, and *Bida & Aouat* [83] provide a brief analysis of the existing swarm-based DTI methods.

Unlike them, in this work:

- The three types of MHs and the HH-based methods to build DTI algorithms are included, not just EA-based approaches for DTI.
- Three types of implementation strategies are introduced, according to the place where an MH is used in the DTI process.
- The differences in the solution representation are highlighted, as they impact the MH-based implementation.
- The principal components of each method are detailed, such as fitness measures and variation operators.
- An analysis of the experimental studies conducted in these methods is provided.

Although DTs are used in other strategies such as classifier ensembles [49], multi-trees [84], multi-test tree [85], and regression trees [86], these models are excluded in this review. The first three determine the class membership of a new instance using different schemes than those defined for a DT, and regression trees use continuous class values.

In this review, each study is associated with some of the following implementation strategies:

- Recursive partitioning (RP): A DT is created like any traditional induction method, but an MH replaces the standard splitting criterion to obtain a better training instances separation. A candidate solution is a test condition, and, in this case, the MH is invoked as many times as internal nodes are required when inducing a DT.
- Global search (GS): One MH performs a global search in the decision-tree-space to find near-optimal DTs. Here, a candidate solution is a DT represented as a hierarchical structure or one sequence of values.
- Subsequent optimization (SO): Taking a DT previously induced by another classification method, the MH optimizes either the tree structure or some of its elements. Here, the MH conducts a post-processing task since it does not induce one DT.

Fig. 4 shows a graphical scheme of these strategies. MH-based approaches for data preprocessing are not considered in this review. Although some MH-based techniques to manipulate the data involve creating a classifier (such as the wrapper schemes for feature subset selection), the MH is not applied on the DT but the data.

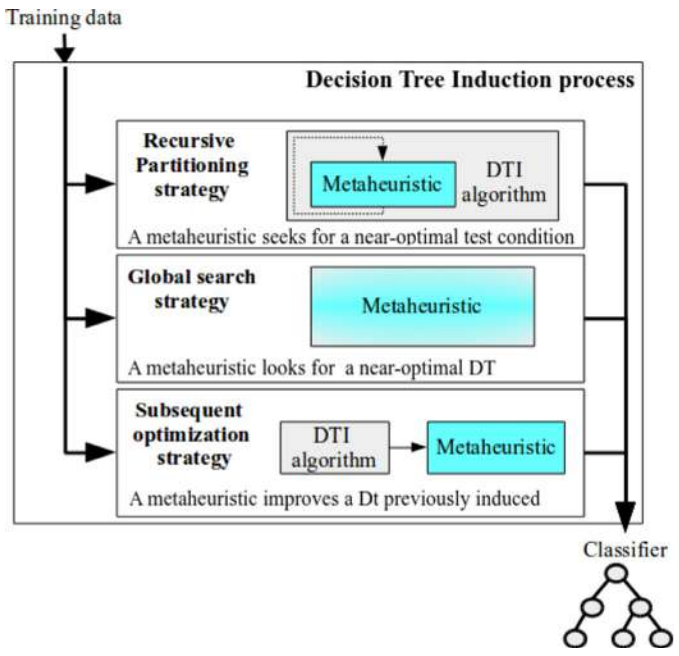
Training data
Decision Tree Induction process
Recursive Partitioning strategy
Metaheuristic
DTI algorithm
A metaheuristic seeks for a near-optimal test condition
Global search strategy
Metaheuristic
A metaheuristic looks for a near-optimal DT
Subsequent optimization strategy
DTI algorithm
Metaheuristic
A metaheuristic improves a Dt previously induced
Classifier

**Fig. 4.** Strategies implemented by the MH-based approaches for DTI.

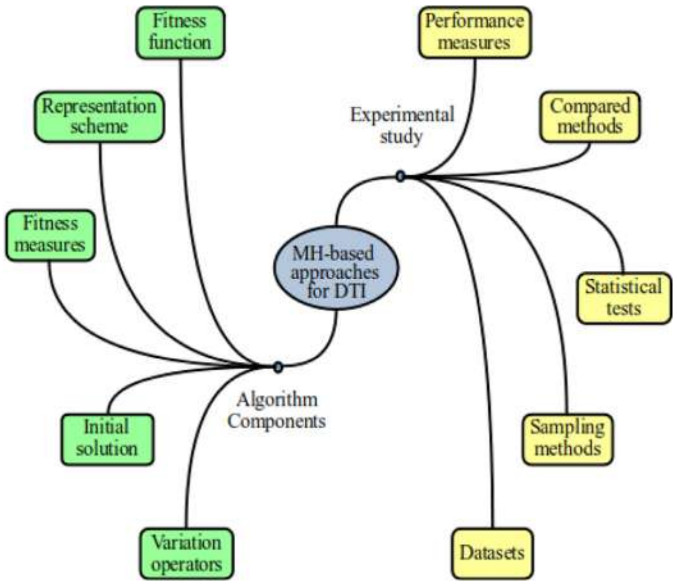

**Fig. 5.** Algorithm components and elements of experimental studies analyzed in this review.

This work is organized according to the type of MH used and the type of DT induced. In each study, the implemented strategy, its main components, and the elements of its experimental studies are described (Fig. 5). In particular, two types of studies can be distinguished: those providing only a general description of their implementations [87–89] and others conducting several experiments to compare the method performance [8,10,11]. For the last case, the results obtained are compared with those of other methods such as C4.5 or CART. Table 3 shows the methods commonly used in the experimental studies reported in the existing literature.

**Table 3**
Main methods used in the experimental studies of MH-based approaches for DTI.

| Method | Description |
|---|---|
| *Methods to build univariate DTs:* | |
| C4.5 | *Quinlan* [30] |
| C5.0 | *Quinlan* |
| CHAID | Chi-square automatic interaction detection [90] |
| ID3 | *Quinlan* [31] |
| J48 | *Witten & Frank* [52] |
| REPTree | Reduced-error pruning tree [52] |
| RTree | Random tree [52] |
| *Methods to build multivariate DTs:* | |
| CART | Classification and regression trees [34] |
| CRUISE | Classification rule with unbiased interaction selection and estimation method [91] |
| QUEST | Quick, unbiased, efficient, statistical tree [92] |
| *Other methods:* | |
| CN2 | Clark-Niblett method [93] |
| kNN | $k$-nearest neighbors [94] |
| LR | Logistic regression [95] |
| MLP | Multi-layer perceptron [96] |
| NB | Naïve Bayes |
| RBF-NN | Radial-basis-function neural network [97] |
| RF | Random Forest [49] |
| SVM | Support Vector Machines [98] |

**Table 4**
Performances measures used in the summary analysis.

| Name | Description |
|---|---|
| *Confusion-matrix-based performance measures:* | |
| Accuracy | Accuracy |
| Error | Error Rate or Misclassification Rate |
| Miss-rate | Miss rate or False Negative Rate |
| F-Score | F-Score, F-Measure or $F_1$ Score |
| Fall-out | Fall out or False Positive Rate |
| Precision | Precision or Positive Predictive Value |
| Sensitivity | Sensitivity, Recall or True Positive Rate |
| Specificity | Specificity or True Negative Rate |
| *Other performance measures:* | |
| AUC | Area under cuve of ROC curve [108] |
| Fidelity | Fidelity [109] |
| HV | Hypervolume [110] |
| J-Measure | J-Measure [111] |
| MC | Misclassification cost |
| ROC | ROC curve [112] |
| Size | Complexity |
| Stability | Stability [113] |
| Time | Running Time |
| Speedup | Speedup-ratio [114] |

### 3.1. Components of the MH-based approaches for DTI

*Fitness function (FF):* This function defines the quality measure used to guide the search for a near-optimal solution. Several splitting criteria (Table 1) and different performance measures (Table 4) have been used as quality measures. Both single-objective and multi-objective fitness functions have been used with these approaches. Only one quality measure is applied with a single-objective fitness function (UF), and two or more measures are assessed with a multi-objective fitness function.

Following the existing multi-objective (MO) optimization literature [99,100], two evaluation schemes are described in this review: On the one hand, an aggregating fitness function (AF) defines a weighted combination of measures to compute the fitness value. Weighting coefficients represent the relative importance of each measure in the combination. Alternatively, a multi-objective fitness function (MF) first evaluates each measure separately and then applies (1) a lexicographic-ordering criterion to rank the measures in order of importance, or (2) a Pareto-based fitness assignment strategy to find a set of non-dominated solutions. A solution is non-dominate if another solution does not exist with better quality measure than the current one without worsening another quality measure [99].

*Representation schemes:* Candidate solutions have been encoded either with sequences of values or tree structures. Sequences of values are commonly used by several EAs such as genetic algorithms, grammatical evolution, and differential evolution, but genetic programming uses tree structures.

*Variation operators:* They are used to create new solutions through two strategies: (1) by altering the current solution values and (2) merging the information on several solutions. Each MH-based approach defines its variation operators based on the representation scheme adopted.

*Initialization procedure:* The random generation of initial solutions is the strategy most commonly implemented with these approaches. However, other MH-based methods for DTI reported in the existing literature start their search process either with a fixed initial solution [101] or create several variants of a DT induced by C4.5 as their initial population [102,103].

### 3.2. Elements of experimental studies

If the performance of one proposed method is evaluated in some studies, an experimental test is conducted using one or several datasets. When this performance is compared with those of similar procedures, one sampling method is used. Also, in some cases, statistical tests are conducted.

*Datasets:* Two types of datasets have been used for performance comparison. The University of California, Irvine (UCI) machine learning repository [104] provides the datasets collection most commonly employed to compare classifiers. Also, several studies evaluate datasets from diverse sources (artificial or private) in their experiments.

*Performance measures:* A performance measure such as accuracy, complexity, and running time is used to compare and evaluate the predictive power of a classifier [105–107]. Table 4 shows a list of performance measures.

*Sampling methods:* Sampling methods are accepted as validation schemes to determine the classifier generalization power [107]. Both cross-validation (CV) and hold-out are the most commonly used sampling methods [115,116]. Several CV variants such as k-fold CV [117] and $5 \times 2$-CV [118] have been applied in these induction methods. Furthermore, bootstrap [119] is another method where a dataset is randomly sampled with replacement, generating small subsets of data. Although DTI using the complete dataset is not suitable to validate classi-

**Table 5**
Statistical tests used to compare the performance of the MH-based approaches for DTI.

| Name | Description |
|---|---|
| ANOVA | Analysis of variance [120] |
| Avg | Average result |
| Avg-R | Average ranks |
| Bergmann | Bergmann-Hommel test [123] |
| Bonferroni | Bonferroni-Dunn test [124] |
| Friedman | Friedman test [121] |
| Holm | Holm test [125] |
| Hommel | Hommel test [126] |
| Iman-D | Iman-Davenport test [127] |
| Kruskall | Kruskall-Wallis test [128] |
| Mann-W | Mann-Whitney *U* test [129] |
| Nemenyi | Nemenyi test [130] |
| *t*-test | Paired *t*-test |
| Tukey | Tukey test [131] |
| Shaffer | Shaffer test [132] |
| Wilcoxon | Wilcoxon test [133] |
| WTL | Counts of wins, ties and losses |

**Table 6**
Acronyms used to identify several SS-based MH for DTI.

| Acronym | Description |
|---|---|
| *Stochastic Local Search:* | |
| OC1 | Oblique Classifier 1 [134] |
| OC1-AP | OC1 for axis-parallel DTs [135] |
| *Simulated Annealing:* | |
| SACS | SA Classifier System [136] |
| SADT | SA of Decision Trees [37] |
| OC1-SA | OC1-based SA [137] |
| *Tabu search:* | |
| EPTS | Extreme Point TS [138] |
| $\text{LDSDT}_{\text{TS}}$ | Linear discrete support vector DT method [36] |
| LDTS | Linear Discriminant and TS [139] |

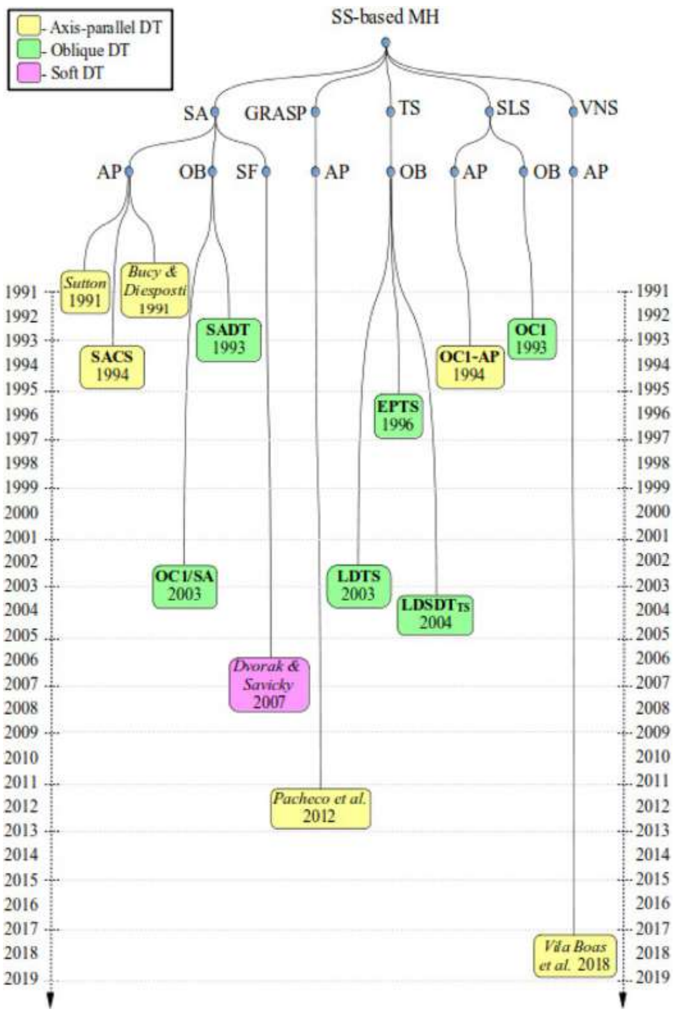

**Fig. 6.** Timeline of the SS-based MHs for DTI.

fier performance, it has been applied in early-proposed methods, mainly to demonstrate its implementation feasibility.

*Statistical tests:* Table 5 describes the statistical tests employed to compare the performance of MH-based approaches. ANOVA [120] and Friedman test [121] are the statistical tests most commonly used to examine differences between experimental results. *Luengo* et al. [122] indicate that exist numerous post-hoc tests to determine whether an algorithm has statistical differences concerning other methods.

## 4. Single-solution-based metaheuristics for DTI

Table 6 lists the acronyms used to name SS-based MHs for DTIs, as defined by their authors. Others do not provide a name for their algorithms, and in this review, these last are identified with the authors reference. The timeline of these methods is shown in Fig. 6.

### 4.1. Axis-Parallel-DTs

*Simulated Annealing:* Three SA-based methods implement a subsequent-optimization strategy to improve a previously induced DT:

1. *Bucy & Diesposti* [101,140] reconfigure a fixed-length binary DT created at random, either (1) exchanging two test conditions, (2) replacing a branch with a new set of test conditions, or (3) creating a new DT. After each modification, a pruning process is used to eliminate unfeasible subtrees.
2. *Sutton* [141] alters a DT induced by CART by shifting its partition boundaries.
3. *Lutsko & Kuijpers* [136] optimize a binary DT produced by ID3 in their SA Classifier System (SACS). SACS randomly selects one test condition using a previously defined weighted distribution and replaces its attribute with an unused one.

In these methods, reconfigured DT is accepted as a new solution through the Boltzmann criterion.

*GRASP: Pacheco* et al. [142] implement one recursive-partitioning strategy using GRASP to build a binary DT. In each iteration, instead of choosing the attribute with the maximum information gain, GRASP randomly selects one from a set of attributes with the highest information gain values. A user-specified parameter defines the set size.

*Variable Neighborhood Search: Vilas Boas* et al. [143] apply VNS to improve the performance of a DT induced by C4.5. This method implements a subsequent-optimization strategy where an attribute is randomly changed in a test condition also selected at random, with leaf nodes' re-optimization.

### 4.2. Oblique-DTs

Several SS-based MHs have been used to build an oblique-DT in one recursive-partitioning strategy, as shown in Fig. 7. In this figure, $\mathbf{w}^T\mathbf{x} = \theta$ represents the hyperplane, $\mathbf{w}$ is the vector of hyperplane coefficients, $\mathbf{x}$ is the vector of attribute values, and $\theta$ is the independent term. Also, $\mathbf{w}^s$ and $\mathbf{x}^s$ are subsets of coefficients and attribute values, respectively, and $\mathbf{w}'$ is the new vector of coefficients produced by the algorithm.

*Stochastic Local Search: Murthy* et al. [134] introduce the Oblique Classifier 1 (OC1) applying a two-step process to find a near-optimal hyperplane. First, by using a deterministic rule, OC1 adjusts the hyperplane

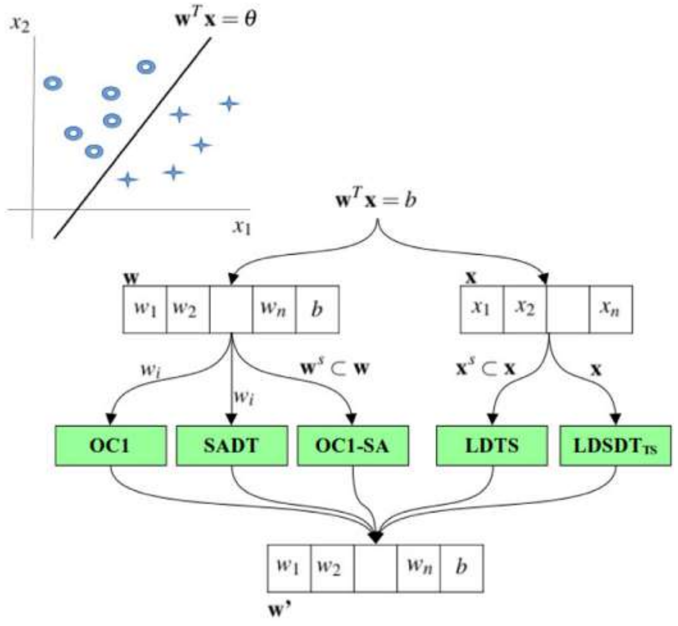
$x_2$
$\mathbf{w}^T\mathbf{x} = \theta$
$x_1$
$\mathbf{w}^T\mathbf{x} = b$
w
$w_1$ $w_2$ $w_n$ $b$
x
$x_1$ $x_2$ $x_n$
$w_i$
$w_i$
$\mathbf{w}^s \subset \mathbf{w}$
$\mathbf{x}^s \subset \mathbf{x}$
x
OC1
SADT
OC1-SA
LDTS
LDSDT$_{TS}$
$w_1$ $w_2$ $w_n$ $b$
w'

**Fig. 7.** Methods to build a new hyperplane using SS-based MHs.

coefficients, taking one by one and looking for its optimal value. Next, it applies SLS to jump out of a locally optimal solution. Also, an OC1 variant to induce an axis-parallel-DT, known as OC1-AP, is described in *Murthy* et al. [135].

*Simulated Annealing: Heath* et al. [37] describe the SA of Decision Trees (SADT) method that, starting with a fixed initial hyperplane, applies SA to improve its fitness value. In each iteration, SA modifies a single hyperplane coefficient that is chosen at random to adjust the hyperplane orientation. Furthermore, *Cantú-Paz & Kamath* [137] implement an OC1 variant known as OC1-SA, where SA simultaneously modifies several hyperplane coefficients. OC1-SA starts with the best axis-parallel hyperplane found by OC1-AP.

*Tabu search:* The Linear Discriminant and TS (LDTS) method of *Li* et al. [139] first selects at random a subset of attributes from the dataset. Then, one hyperplane is created using this subset through a Linear Discriminant Analysis (LDA). Finally, LDTS iteratively replaces an attribute in this subset with another one outside it. The new attribute is included in the subset while improving the hyperplane quality. The replaced attribute is added to the tabu list. Also, *Orsenigo & Vercellis* [36] use TS in the discrete support vector DT method (LDSDT$_{TS}$), implementing a linear discrete SVM (LDSVM) as its splitting criterion. A hyperplane is modeled as a mixed-integer linear program, which is solved using both LDSVM and TS.

On the other hand, *Bennet & Blue* [138] describe the Extreme Point TS (EPTS) algorithm modifying a DT previously induced by the Multisurface Method-Tree (MSMT) [144]. DT is represented as a system of disjunctive linear inequalities, and TS is applied in the pivoting procedure of the Simplex method used to solve this system. An attribute is inserted into the tabu list when it leaves the basis of the system. The neighbors used by TS are the nonbasic attributes of the linear constraints. The best neighbor is one that, when it is considered one basic attribute, improves the fitness hyperplane value.

### 4.3. Soft-DTs

*Simulated Annealing: Dvořák & Savický* [145] implement a subsequent-optimization strategy to find the soft threshold values of the test conditions of an axis-parallel-DT induced by CART. Here, SA perturbs a subset of these values in each step of its iterative process.

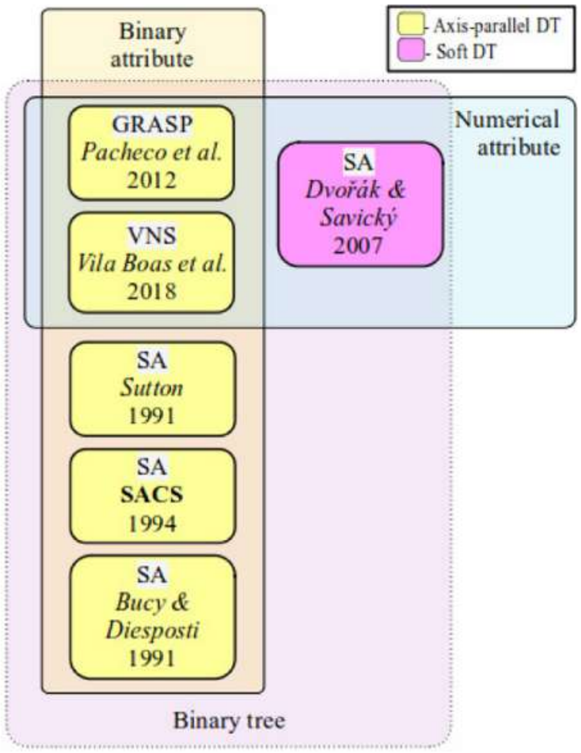

**Fig. 8.** DTs and attribute types used with SS-based MHs inducing axis-parallel-DTs.

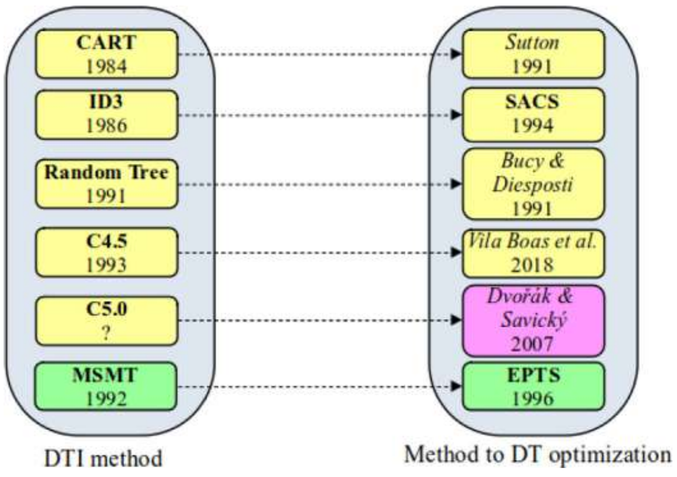

**Fig. 9.** Scheme used to subsequent-optimization with SS-based MHs.

### 4.4. Discussion

All methods described in this section induce binary DTs only. Fig. 8 shows the attribute and branch types of axis parallel DTs generated using these methods. In particular, the GRASP-based algorithm described by *Pacheco* et al. transforms categorical attributes into several binary attributes. Finally, Fig. 9 shows the scheme used for the methods implementing a subsequent-optimization strategy.

The components of these methods are described in Table 7, and the experimental analysis reported is summarized in Table 8. Two studies use an aggregating fitness function, and the remaining apply single-objective fitness functions. Six methods use a splitting criterion such as information gain and twoing rule as fitness measures. Several algorithms use performance measures such as error-rate and size. Furthermore, this table shows that the linear representation of candidate solu-

**Table 7**
Components of SS-based MHs for DTI.

| Strategy | DT | FF | MH | | Studies | Year | Repr. scheme | Fitness measures | Initial solution |
|---|---|---|---|---|---|---|---|---|---|
| RP | AP | UF | GRASP | · | *Pacheco* et al. [142] | 2012 | Tree | IG | - |
| | OB | UF | SLS | · | OC1 [134,135] | 1993 | Linear | IG, MaxM, SumM, SumV, Gini, Twoing | The best axis-parallel hyperplane found by OC1-AP |
| | | | SA | · | SADT [37] | 1993 | Linear | SumM | A fixed hyperplane |
| | | | | · | OC1-SA [137] | 2003 | Linear | Twoing | The best axis-parallel hyperplane found by OC1-AP |
| | | | TS | · | LDTS [139] | 2003 | Linear | IG | A hyperplane constructed using LDA with a subset of attributes randomly selected from the dataset |
| | | AF | TS | · | LDSDT$_{TS}$ [36] | 2004 | Linear | Error+MarginS | A feasible hyperplane modeled as a linear mixed integer problem and solved using a truncated B&B |
| SO | AP | UF | SA | · | *Sutton* [141] | 1991 | Tree | Gini | A DT induced by CART |
| | | | | · | SACS [136] | 1994 | Tree | MDL | A DT induced by ID3 |
| | | AF | SA | · | *Bucy & Diesposti* [101,140] | 1991 | Tree | Error+Size | A fixed-length DT randomly created |
| | | | VNS | · | *Vila Boas* et al. [143] | 2018 | Tree | Accuracy+Size | A DT induced by C4.5 |
| | OB | UF | TS | · | EPTS [138] | 1996 | Tree | Error | A DT induced by MSMT |
| | SF | UF | SA | · | *Dvořák & Savický* [145] | 2007 | Tree | Error | A DT induced by C5.0 |

**Table 8**
Experimental analysis reported by SS-based MHs for DTI.

| Strategy | DT type | MH | | Studies | Datasets UCI | Datasets other | Sampling method | Performance measures | Statistical tests | Compared methods |
|---|---|---|---|---|---|---|---|---|---|---|
| RP | AP | GRASP | · | *Pacheco* et al. [142] | 17 | - | 10-f CV | Accuracy, Size | Avg, WTL, *t*-test | Traditional DTI method |
| | OB | SLS | · | OC1 [134,135] | 2 | 2 | 10-f CV | Accuracy, Size | - | ID3, SADT, kNN, MLP |
| | | SA | · | SADT [37] | 2 | 2 | 10-f CV | Accuracy, Size | - | ID3 |
| | | | · | OC1-SA [137] | 10 | - | 5-f CV | Accuracy, Size, Time | *t*-test | CART, OC1, OC1-ES, OC1-GA |
| | | TS | · | LDTS [139] | 13 | - | 10-f CV | Error, Size, Time | Avg, Tukey, ANOVA | C4.5, CART, OC1, QUEST, Linear tree [146] |
| | | | · | LDSDT$_{TS}$ [36] | 6 | 2 | 10-f CV | Accuracy, Size, Time | - | C4.5, OC1, SVM, QUEST, LDSDT$_{B\&B}$ [36] |
| SO | AP | SA | · | *Bucy & Diesposti* [101,140] | - | 3 | Complete dataset | Error, Size | - | Huffman algorithm [147] |
| | | | · | *Sutton* [141] | - | - | - | - | - | - |
| | | | · | SACS [136] | - | 10 | Hold-out | Error, Size | - | Gelfand-Ravishankar-Delp method [148] |
| | | VNS | · | *Vila Boas* et al. [143] | - | 308 | 10f-CV | Accuracy, Size, Time | - | Locally weighted learning [149], Sequential Minimal Optimization [150], Random Committee [52] |
| | OB | TS | · | EPTS [138] | 5 | 3 | 5-f CV | Error | - | MSMT, Frank-Wolfe method [151] |
| | SF | SA | · | *Dvořák & Savický* [145] | - | 1 | Hold-out | Error | - | C5.0, CART |

tions has only been used to build oblique-DTs. Finally, several strategies have been implemented to generate the first candidate solution: two studies create it randomly, and others start their search procedure with a previously induced DT.

K-fold CV is the sampling method most used in these studies, and only two apply the hold-out scheme. Furthermore, accuracy, error-rate, and size are the performance measures most commonly adopted in these methods. Finally, three studies use a statistical test to analyze their experimental results, and only one of them conducts one post-hoc analysis.

The use of only five SS-based MHs has been explored for DTI. Except for the GRASP-based method, they try improving an initial solution (internal node or decision tree) by applying an iterative strategy. All these approaches can suffer the same problems of traditional DTI procedures since they use one recursive-partitioning strategy or, in the subsequent-optimization case, start with a previously induced DT, limiting the search for solutions to that tree's neighborhood.

## 5. Evolutionary algorithms for DTI

EAs are the most widely used population-based MHs to induce DTs. Numerous EA-based approaches conduct a global search to find near-optimal DTs. However, recursive-partitioning and subsequent-optimization strategies have also been implemented with these MHs. Genetic algorithms and genetic programming methods are the EAs most commonly used for DTI.

Following the definitions and interpretations provided in the existing literature, two types of GAs-based approaches are described in this review: Genetic algorithms with linear chromosomes (LGA) and tree-based genetic algorithms (TGA), although there exists disagreement about whether to consider TGAs as genetic algorithms or genetic programming methods. Since one TGA disturbs the tree population in its evolutionary process, it can be regarded as a GP algorithm. Nevertheless, *Koza* [152] points out that GP individuals are generated by combining elements taken from function and terminal sets, and TGAs do not define any set to represent candidate solutions. In contrast, some authors point out that they implement a GP-based approach but build their trees without using the previously referred sets. In this review, analyzed studies are classified according to the definition provided in them. Table 9 illustrates the acronyms used to identify various GA-based MHs for DTIs, as defined by their authors. The timeline of GA-based approaches for DTI is shown in Fig. 10. Furthermore, the genetic operators used by these studies are described in Table 10.

### 5.1. Genetic algorithms with linear chromosomes

LGA can use binary, integer, or real-valued chromosomes. Since these chromosomes represent DTs, the next mapping is commonly applied (Fig. 11): First, the initial chromosome element is used as the tree root-node. Next, remaining elements are inserted in the tree as successor nodes of those previously added so that each new tree-level is completed before placing new nodes at the next one, similar to a breadth-first-search strategy. In univariate DTs, the number of successors of an internal node is calculated based on the attribute domain used in its test condition. For a multivariate DT, each internal node has two successor nodes.

#### 5.1.1. Axis-Parallel-DTs

*Kennedy* et al. [7] introduce the Caltrop method, a global-search strategy evolving a population of DTs represented as sets of sub-trees. Each sub-tree with three nodes (a caltrop) is encoded with an integer-valued vector referring to binary attributes used as test conditions.

Chromosomes used in other approaches encode (1) nodes of a complete DT or (2) tree nodes' internal elements (attributes, class labels, and threshold values). In the first case:

1. *Cha & Tappert* [175,176] encode test conditions and leaf nodes with an integer-valued vector. One test condition is identified with the relative location of the binary attribute in a list of ordered attributes. Similarly, one leaf node is encoded with the location of its class label in its corresponding list.
2. *Bandar* et al. [87] represent one internal node with an index identifying the evaluated attribute. Here, threshold values and leaf nodes are determined by evaluating the training set.
3. Evolutionary Classifier with Cost Optimization (ECCO) of *Omielan & Vadera* [155] use a binary chromosome to represent the test conditions of a complete DT by indexes identifying them. ECCO is used to induce cost-sensitive multi-branch DTs.

In the other case, the works of *Smith* [177] and *Ersoy* et al. [178], as well as the Evolutionary Algorithm for DTI (EVO-Tree) of *Jankowski & Jackowski* [157] implement similar approaches. They use two arrays to store the internal elements of nodes making up a binary DT: the one identifies attributes and class labels, and the other threshold values. Also, in the Bi-level GA (BiLeGA), *Adibi* [153] encodes only attributes indexes and threshold values in a numeric-valued chromosome.

Finally, *Grubinger* et al. [3] introduce the evtree (Evolutionary Tree) package using a variable-length numerical chromosome with attributes and splitting rules of a binary DT. Attributes are identified by an index of its dataset relative position, and splitting-rules contain threshold values.

Fig. 12 shows an example of these encoding schemes. A complete DT with three test conditions and four leaf-nodes is depicted. This tree is induced from a hypothetical training set with three binary attributes and two class labels.

*András & Dumitrescu* [171] apply an alternative encoding scheme named Multi-Expression Programming (MEP)[1] in the MEP-based DTI (MEPDTI) method where test conditions are encoded as functional symbols, and leaf nodes as terminals.

#### 5.1.2. Oblique-DTs

LGA-based approaches commonly use a linear chromosome representing the hyperplane coefficients of one oblique-DT induced through a recursive-partitioning strategy. *Chai* et al. [154] implement the Binary Tree-Genetic Algorithm (BTGA) encoding these coefficients in a binary sequence. Furthermore, four studies use real-valued chromosomes: an OC1 variant (OC1-GA) of *CantȦ-Paz & Kamath* [11,137], the methods proposed by *Krętowski* [33] and *Pangilinan & Janssens* [180], and the Real-Coded GA-based Linear DT Algorithm with k-D Trees (RCGA-kDT) of *Ng & Leung* [173].

In particular, *Struharik* et al. [169] apply a GA variant[2] in the HereBoy for DT (HBDT) method using a single fixed-length binary chromosome. *Vukobratovic & Struharik* [156] apply the same GA variant in the Evolutionary Full Tree Induction (EFTI) method with a single variable-length linear chromosome. EFTI grows the DT structure during its evolutionary process, starting with a one-node DT randomly created.

Alternatively, global search is conducted with the Generalized DT Inducer (GDTI) of *Dumitrescu & András* [167] that uses MEP to represent oblique-DTs.

#### 5.1.3. Non-linear-DTs

*Llorà & Garrell* [162] and *Llorà & Wilson* [182] implement the Genetic and Artificial Life Environment (GALE), a parallel global-search strategy evolving a population of DTs placed in a two-dimensional grid. DTs can be axis-parallel, oblique, or non-linear DTs. In particular, non-linear-DTs use hyper-spheres as test conditions.

Alternatively, two studies implement a recursive-partitioning strategy to build one non-linear DT:

[1] MEP [179] defines a linear chromosome composed of variable-length genes. Each gene encodes a terminal or a functional symbol. Functional symbols contain functions taking as arguments the indices of other elements.

[2] The HereBoy algorithm (HBA) [181] is a particular GA evolving a unique binary chromosome using a mutation operator only.

**Table 9**
Acronyms used to identify several GA-based MH for DTI.

| Acronym | Description |
|---|---|
| BiLeGA | Bi-level GA [153] |
| BTGA | Binary Tree-GA [154] |
| Caltrop | Caltrop algorithm [7] |
| ECCO | Evolutionary Classifier with Cost Optimization [155] |
| EFTI | Evolutionary Full Tree Induction [156] |
| EVO-Tree | Evolutionary Algorithm for DTI [157] |
| evtree | Evolutionary Tree [3] |
| FVBDT | Fuzzy Variable-Branch DT [158] |
| G-DT | Genetically optimized fuzzy-DT [159] |
| GA-FID3 | GA for Fuzzy ID3 [160] |
| GA-QDT | GA-based Quadratic DT method [161] |
| GAIT | GA inducing trees [102] |
| GALE | Genetic and Artificial Life Environment [162] |
| GATree | Genetically Evolved DTs [163] |
| GC-SDT | Genetically optimized Cluster oriented Soft DTs [39] |
| GDT | Global EA for DTI [9] |
| GDT-MA | GDT with local-search [164] |
| GDT-MC | GDT for cost-sensitive classification [165] |
| GDT-Mix | Global Induction of Mixed DTs [166] |
| GDTI | Generalized DT Inducer [167] |
| GEA-ODT | Global EA for oblique DTI [166] |
| genTrees | Genetic decision trees [168] |
| HBDT | HereBoy for DT method [169] |
| IIVFDT | Ignorance functions based Interval-Valued Fuzzy DT with genetic tuning [170] |
| LEGAL-Tree | Lexicographic GA for Learning DTs [10] |
| MEPDTI | MEP-based DTI method [171] |
| MS | Meta-Silvae [113] |
| NLDT | Non-linear DT method [172] |
| OC1-GA | OC1-based Genetic Algorithm [11] |
| RCGA-kDT | Real-Coded GA-based linear DT algorithm with k-D Trees [173] |
| TARGET | Tree Analysis with Randomly Generated and Evolved Trees [174] |

**Table 10**
Genetic operators used by EA-based approaches of DTI.

| Acronym | Description |
|---|---|
| *Selection operators:* | |
| Tournament | Tournament-based selection |
| Roulette | Roulette-wheel-based selection |
| Exponential ranking | Exponential-ranking-based selection |
| Linear ranking | Linear-ranking-based selection |
| Truncate | Truncate-based selection |
| Assortative | Assortative mating |
| *Crossover operators:* | |
| Double-point | Double-point crossover |
| Uniform | Uniform crossover |
| Arithmetic | Arithmetic crossover |
| Single-point | Single-point crossover |
| Binomial | Binomial crossover |
| *Mutation operators:* | |
| Bit-string | Bit-string mutation |
| Flip-bit | Flip-bit mutation |
| Uniform | Uniform mutation |
| Non-uniform | Non-uniform mutation |

1. *Ng & Leung* [161,183] introduce the GA-QDT (GA-based Quadratic DT) method to find a near-optimal hypersurface used as test condition. Each hypersurface is modeled as Eqn. (3)

$$\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} > \theta \tag{3}$$

where $\mathbf{x}$ is a vector of attribute values, $\mathbf{A}$ is a symmetric matrix, $\mathbf{b}$ is a vector, and $\theta$ is the independent term in the inequality. Each GA-QDT chromosome encodes the values of $\mathbf{A}$, $\mathbf{b}$, and $\theta$.
2. *Dhebar & Deb* [172] use a bilevel GA in their NLDT method. A population of linear combinations of power-law rules containing the dataset attributes evolve to find a near-optimal test condition. A power-law rule $\mathbf{B}$ is defined as Eqn. (4).

$$\mathbf{B} = \prod_{i=1}^{d} x_i^{b_i} \tag{4}$$

where $x$ is an attribute value, $d$ is the number of attributes, and $b = \{-3, -2, \dots, 2, 3\}$.

### 5.1.4. Soft-DTs

*Janikow* [184] and *Chang* et al. [160] implement similar recursive-partitioning strategies encoding the membership-functions used by one soft-DT test condition in a real-valued chromosome: The first uses the corners of trapezoidal functions, and the others represent the mean and variance of Gaussian functions in their GA for Fuzzy ID3 (GA-FID3).

The subsequent-optimization strategy is also used to create soft-DTs. Here, an LGA evolves chromosomes encoding fuzzy regions associated with all soft-DT test conditions. *Crockett* et al. [185] and *Sanz* et al. [170] use real-valued chromosomes: the first by optimizing piecewise-linear functions of a DT induced by ID3, and the second, in the IIVFDT (Ignorance functions based Interval-Valued Fuzzy DT with genetic tuning) method, improving the ignorance-degree[3] associated with triangular functions of a DT induced by Fuzzy-ID3 [187]. Furthermore, in the Genetically optimized fuzzy-DT (G-DT) method, *Pedrycz & Sosnowski* [159] use a binary chromosome encoding membership-functions (piecewise-linear or Gaussian) of a DT induced by C4.5.

Alternatively, *Kim & Ryu* [188] implement a global-search strategy to create fuzzy-DTs, using a real-valued vector representing triangular functions.

DTI by clustering is also used to build soft-DTs in two approaches implementing a recursive-partitioning strategy, as shown in Fig. 13. The GC-SDT (Genetically optimized Cluster oriented Soft DTs) inducer of *Shukla & Tiwari* [39] encodes the centroids of instances groups, and the FVBDT (Fuzzy Variable-Branch DT) method of *Yang* [158] computes the number of branches outgoing from one internal node.

[3] Ignorance degree quantifies the uncertainty to assign membership values in fuzzy sets when a classifier is being trained [186].

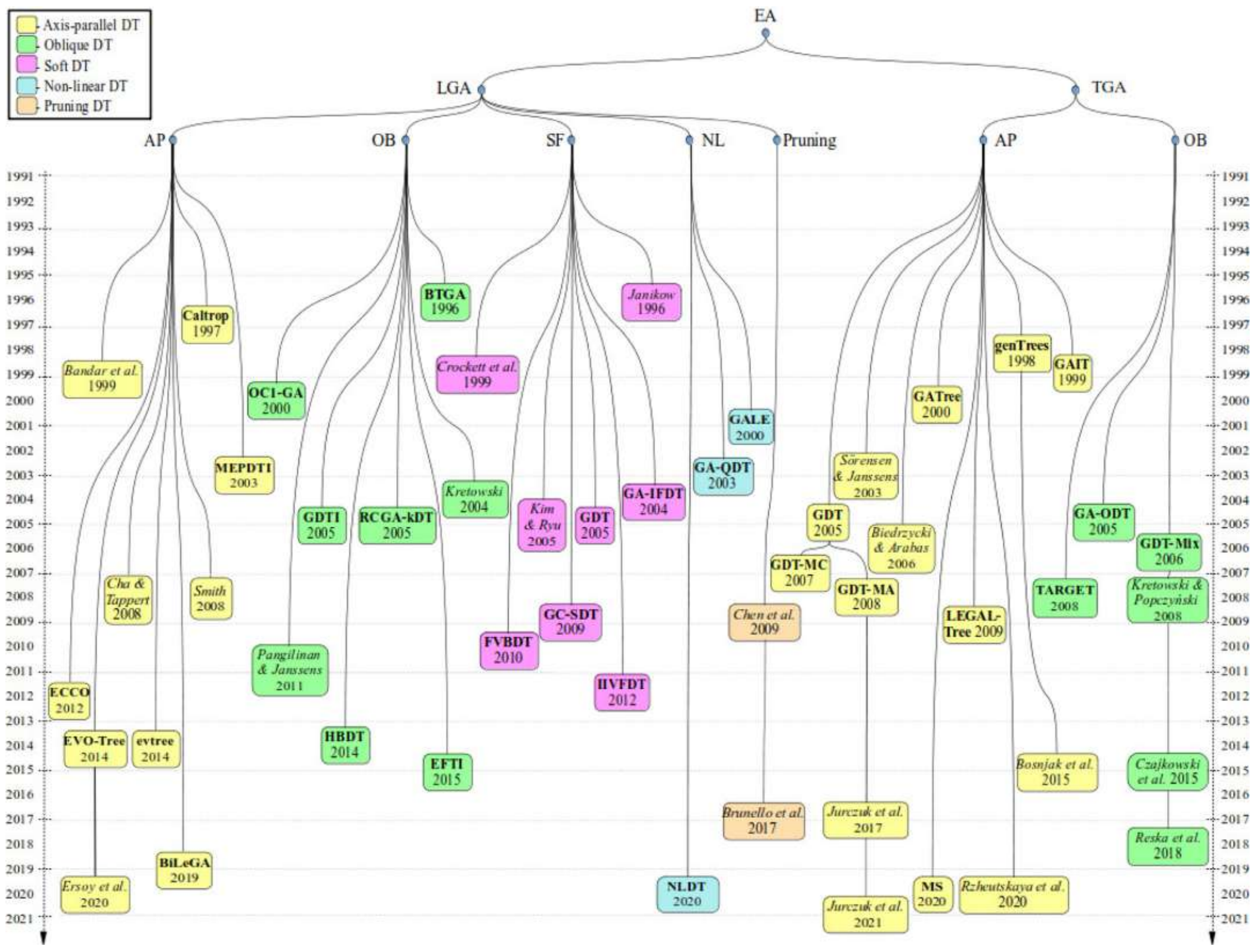

**Fig. 10.** Timeline of GA-based approaches for DTI.

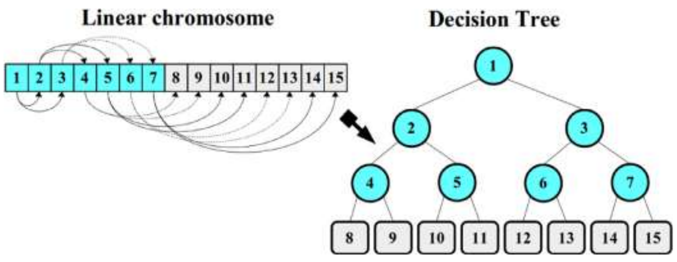

**Fig. 11.** Mapping strategy to build a DT from a linear chromosome.

**Table 11**
Criteria used to handle categorical attributes with LGA-based methods.

| Criteria | Study |
|---|---|
| Multi-branching | ECCO [155], MEPDTI [171] |
| Binary-branching | evtree [3] |
| Numerical mapping | *Ersoy* et al. [178], EVO-Tree [157] |

### 5.1.5. DT Pruning

Two studies using binary-valued vectors have been conducted for DT pruning: *Chen* et al. [189] encode the edges, and *Brunello* et al. [190] represent the test conditions, of a DT induced by ID3 and J48, respectively. In these chromosomes, value 1 indicates that the element is removed from the DT, and its associated sub-tree is pruned.

### 5.1.6. Discussion

Most of the methods that build axis-parallel DTs use a breadth-first strategy to map a DT from a chromosome. However, two algorithms first apply a depth-first strategy to place internal nodes in a DT and then insert leaf nodes evaluating the training set [3,178].

Fig. 14 shows the attribute and branch types of axis parallel DTs induced by previous studies, and Table 11 lists the criteria used to handle categorical attributes.

Fig. 15 shows the methods used in the studies implementing a subsequent-optimization strategy. Finally, since LGA-based methods encoding one axis-parallel DT on a linear chromosome, the chromosome's size is a priori defined (Table 12). However, some studies use a variable-length chromosome ([156,162,171]).

Table 13 summarizes the principal components of LGA-based methods for DTI. The single-objective fitness function is the tree-quality evaluation procedure most used in these studies, but seven works implement an aggregating fitness function, and only one uses a multi-objective fitness function. Ten studies evaluate some splitting criterion as fitness measure, and accuracy and size are most commonly utilized in those implementing global-search or subsequent-optimization strategies.

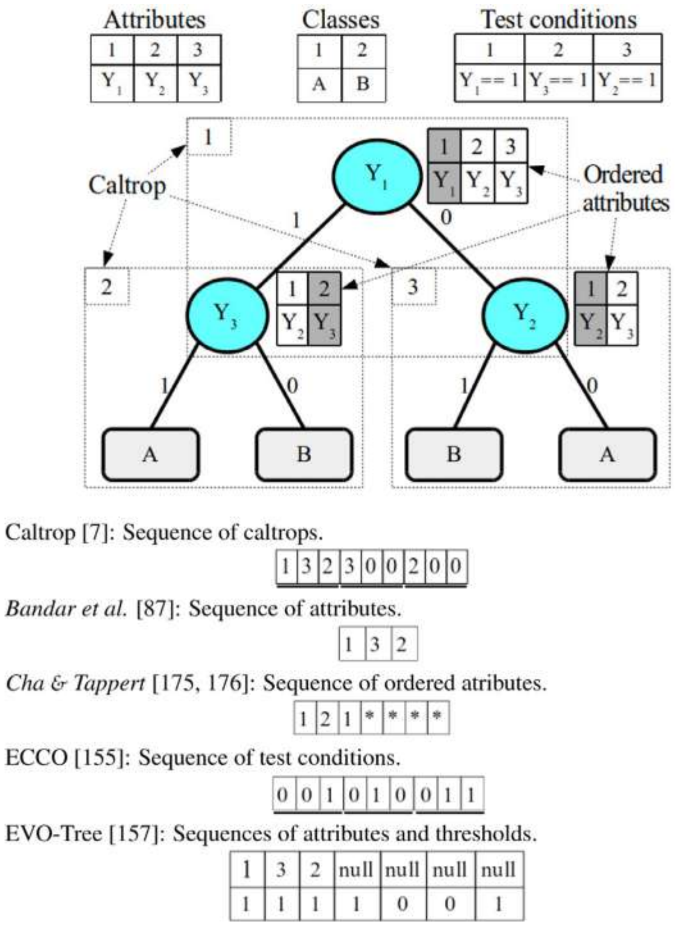
Attributes
Classes
Test conditions
Caltrop
Ordered attributes
Caltrop [7]: Sequence of caltrops.
1 3 2 3 0 0 2 0 0
Bandar et al. [87]: Sequence of attributes.
1 3 2
Cha & Tappert [175, 176]: Sequence of ordered atributes.
1 2 1 * * * *
ECCO [155]: Sequence of test conditions.
0 0 1 0 1 0 0 1 1
EVO-Tree [157]: Sequences of attributes and thresholds.
1 3 2 null null null null
1 1 1 1 0 0 1

**Fig. 12.** Linear chromosome representation of one DT.

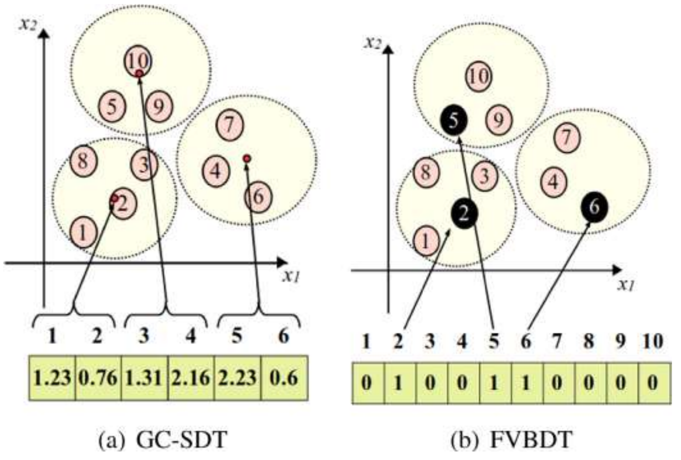

**Fig. 13.** Linear chromosome representation for clusters-based DTI.

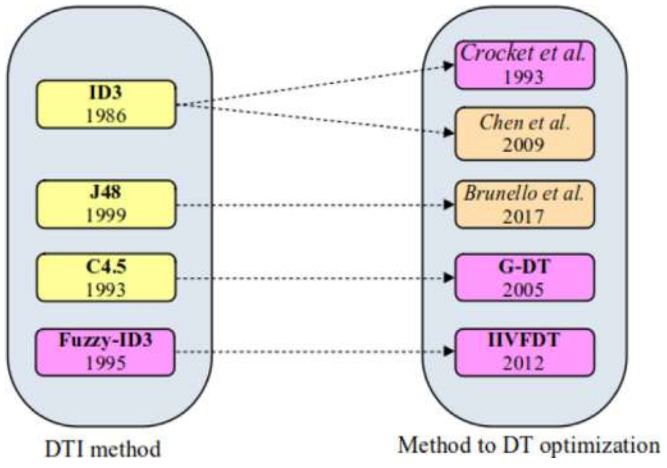

**Fig. 15.** Scheme used to subsequent-optimization with LGA-based MHs.

Furthermore, some authors describe the genetic operators applying in their works, but others only report using some GA-based library such as GENESIS (Genetic Search Implementation System) [191], GENOCOP [192], and GALib [193]. Tournament and roulette-based selection, single-point and double-point crossover, as well as bit-string and uniform mutation, are the genetic operators most commonly applied by these LGAs, although several authors create ad hoc operators. Finally, the initial population in these works is commonly created at random, but only OC1-GA introduces several copies of the best axis-parallel hyperplane found by OC1-AP in its initial population.[4]

[4] A bisecting hyperplane is that containing the normal vector to the segment connecting two instances with different class labels. This hyperplane cuts that segment into two equal parts.

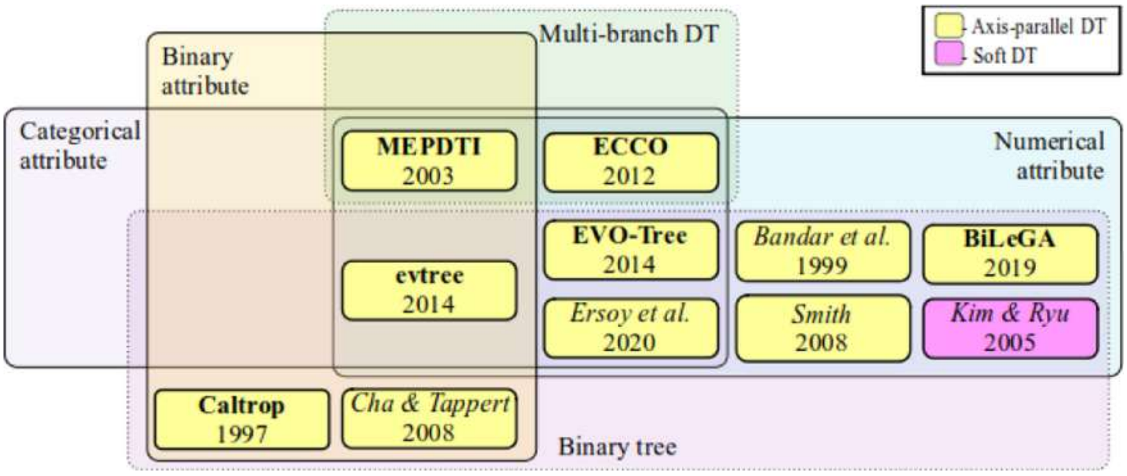

**Fig. 14.** DTs and attribute types used with LGA-based MHs inducing axis-parallel-DTs.

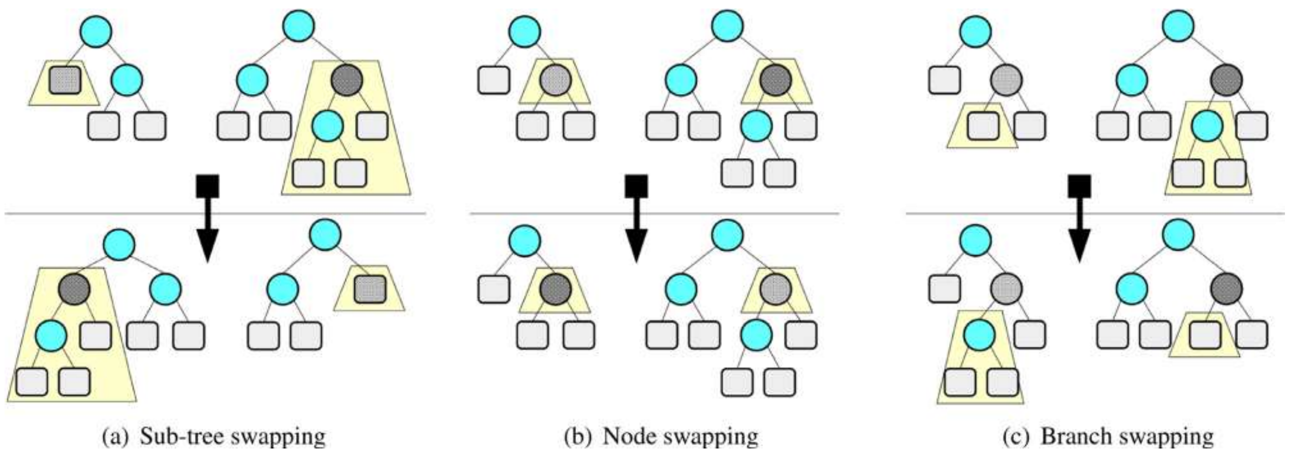

**Fig. 16.** Crossover operators used by tree-based chromosomes (Adapted from [163], [88], [204], [9], and [205]).

**Table 12**
Size of a linear chromosome encoding axis-parallel DTs.

| Study | Number of genes |
|---|---|
| Caltrop [7] | 72 |
| *Bandar* et al. [87] | 63 |
| MEPDTI [171] | Variable |
| *Cha & Tappert* [175] | 127 |
| *Smith* [177] | $2^{h+1} - 1$ \| $h$ is the number of attributes |
| ECCO [155] | $2^h - 1$ \| $h \in [5, 11]$ or 50,000 |
| evtree [3] | Variable |
| EVO-Tree [157] | $2^h - 1$ \| $h$ is a user-specific value |
| BiLeGA [153] | $2(2^h - 1)$ \| $h$ is the number of attributes |
| *Ersoy* et al. [178] | $2(2^h - 1)$ \| $h \in [2, 5]$ |

Table 14 summarizes the experimental studies of these approaches. K-fold CV and hold-out are the sampling methods most commonly applied in these works. Test accuracy and the tree size are computed to determine the method performance in most of these studies. Finally, only eleven studies report applying a statistical test to compare their experimental results, and three of them apply a post-hoc analysis after finding statistical differences.

LGA-based approaches for DTI are competitive methods to build precise classifiers. However, some only induce DTs with a particular attribute type (binary or numerical attributes, for example). In particular, LGA is the MH most used to induce soft-DTs. A drawback for some LGA-based methods is that the chromosome size is defined without considering the dataset characteristics, affecting the model performance.

## 5.2. Tree-based genetic algorithms (TGA)

When GAs use tree-like chromosomes, several specialized genetic operators to build valid offsprings have been implemented in the studies found in the existing literature. Table 15 details these operators, and Figs. 16 and 17 show a graphical scheme for each one.

### 5.2.1. Axis-Parallel DTs

Several approaches implementing a global search of DTs with numerical and categorical attributes have been developed, such as:

1. The GATree (Genetically Evolved DTs) method of *Papagelis & Kalles* [163,206].
2. Global EA for DTI (GDT) and its variants for cost-sensitive classification (GDT-MC), and with local-search (GDT-MA) of *Krętowski & Grześ* [9,164,165], as well as its parallel versions of *Jurczuk* et al. [207–209].
3. The genetic decision trees (genTrees) induction method of *Podgorelec & Kokol* [168].

In particular, to reduce processing time: (a) all possible thresholds are pre-calculated in GDT [9], (b) the reuse of fitness values is applied in the GATree latest version [206], and (c) the implementation of diverse parallel approaches are conducted [207–209].

Furthermore, *Basgalupp* et al. [10] propose LEGAL-Tree (Lexicographic GA for Learning DTs), a multi-objective GA inducing DTs encoded as a collection of decision stumps[5] A lexicographic scheme using tolerance thresholds is used to select either accuracy or size as the fitness value. They improve LEGAL-Tree by including a beam-search-procedure to create the initial population and applying a statistical test in the selection operator [205]. Finally, to prevent premature algorithm convergence, *Bosnjak* et al. [210] introduce diversity in the population through a modified selection operator: One DT is selected based on its fitness value, and the other according to its similarity level.

Alternatively, some studies induce DTs with specific attribute types:

1. *Fu* [102], *Fu & Mae* [211] and *Fu* et al. [204,212–214] implement several variants of GAIT, a method to evolve binary DTs with numerical attributes.
2. *Ranzato & Zanella* [113] also evolve DTs with numerical attributes in their Meta-Silvae (MS) method.
3. *Sörensen & Janssens* [88] evolve DTs with binary attributes.
4. *Biedrzycki & Arabas* [215] use categorical attributes, only.

Different from previous methods, *Rzheutskaya* et al. [216] implement a recursive-partitioning strategy to build a DT by selecting the best split criterion (information gain, Gini, and others) for each test condition.

### 5.2.2. Oblique DTs

In Global EA for oblique DTI (GEA-ODT) of *Krętowski & Grześ* [217,218] a population of oblique DTs is evolved. GEA-ODT implements a global-search strategy to find the DT structure and the hyperplane coefficients used as test conditions. Furthermore, they implement the GDT-Mix (Global Induction of Mixed DTs) method to build DTs with univariate and oblique test conditions [166]. GDT-Mix has been used to build three parallel versions by *Krętowski & Popczyński* [219], *Czajkowski* et al. [114], and *Reska* et al. [220]. Finally, *Gray & Fan* [174] modify their regression tree induction approach, known as TARGET (Tree Analysis with Randomly Generated and Evolved Trees), to induce near-optimal oblique DTs. The hyperplanes can be constructed using up to three attributes.

### 5.2.3. Discussion

Fig. 18 shows the attribute and branch types of the axis-parallel DTs induced with the studies described previously. Table 16 lists the criteria used to handle categorical attributes.

[5] A decision stump is a DT with one internal and two leaf nodes.

**Table 13**
Components of LGA-based approaches for DTI.

| Stra-tegy | DT | FF | Studies | Year | Fitness measure | Genetic operators | | | Initial population |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Selection | Crossover | Mutation | |
| RP | OB | UF | · BTGA [154] | 1996 | Gini | Roulette | Double-point | Bit-string | Bisecting hyperplanes[4] randomly created |
| | | | · OC1-GA [11,137] | 2000 | Twoing | Tournament | Uniform | N/A | Hyperplanes randomly created with several copies of the best axis-parallel hyperplane found by OC1-AP |
| | | | · *Krętowski* [33] | 2004 | Dipolar | Roulette | Double-point | Special | Bisecting hyperplanes randomly created |
| | | | · RCGA-kDT [173] | 2005 | IG | Roulette | Arithmetic | Special | Bisecting hyperplanes randomly created |
| | | | · *Pangilinan & Janssens* [180] | 2011 | Twoing | Tournament | Arithmetic | Non-uniform | Hyperplanes randomly created with one axis-parallel hyperplane randomly created |
| | | | · HBDT [169] | 2014 | IG | N/A | N/A | Bit-string | One bisecting hyperplane passing through the centre point of a set of training instances |
| | NL | UF | · GA-QDT [161,183] | 2003 | Gini | Roulette | Double-point | Non-uniform | Quadric hypersurfaces randomly created |
| | | | · NLDT [172] | 2020 | Gini | Tournament | Special | Special | Linear combinations of power-law functions with one attribute randomly created |
| | SF | UF | · *Janikow* [184] | 1996 | IG | | GENOCOP genetic operators | | Numerical chromosomes randomly creared |
| | | | · GC-SDT [39] | 2009 | SSRE | Tournament | Double-point | Bit-string | Binary chromosomes randomly created |
| | | AF | · GA-FID3 [160] | 2004 | Accuracy+Size | - | - | - | Numerical chromosomes randomly created |
| | | | · FVBDT [158] | 2010 | Error+Size | Roulette | Double-point | Bit-string | Binary chromosomes randomly created |

(*continued on next page*)

**Table 13** (*continued*)

| Strategy | DT | FF | Studies | Year | Fitness measure | Genetic operators | | | Initial population |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Selection | Crossover | Mutation | |
| GS | AP | UF | · Caltrop [7] | 1997 | Size | Assortative | Single-point | Uniform | Integer chromosomes randomly created |
| | | | · *Bandar* et al. [87] | 1999 | Accuracy | - | - | - | Integer chromosomes randomly created |
| | | | · MEPDTI [171] | 2003 | Accuracy | Tournament | Double-point | Special | Integer chromosomes randomly created |
| | | | · *Cha & Tappert* [175,176] | 2008 | Size | - | Single-point | Uniform | Integer chromosomes randomly created |
| | | | · ECCO [155] | 2012 | MC | | GENESIS genetic operators | | Binary chromosomes randomly created |
| | | | · evtree [3] | 2014 | Error | Special | Special | Special | Numerical chromosomes with one test condition |
| | | | · BiLeGA [153] | 2019 | Accuracy | Roulette | Double-point | Special | Numerical chromosomes randomly created |
| | | | · *Ersoy* et al. [178] | 2020 | Accuracy | Roulette | Single-point, Double-point | Uniform | Numerical chromosomes randomly created, CART-based DTs |
| | | AF | · *Smith* [177] | 2008 | Error+Time | - | - | - | Integer chromosomes randomly created |
| | | | · EVO-Tree [157] | 2014 | Error+Size | Roulette | Single-point | Uniform | Numerical chromosomes randomly created |
| | OB | UF | · GDTI [167] | 2005 | Accuracy | Tournament | Special | Special | Integer chromosomes randomly created |
| | | AF | · EFTI [156] | 2015 | Accuracy+Size | N/A | N/A | Special | A hyperplane of a one-node DT |
| | NL | UF | · GALE [162,182] | 2001 | Accuracy | Special | Single-point | Uniform | Integer chromosomes randomly created |
| | SF | AF | · *Kym & Ryu* [188] | 2005 | Accuracy+Size | Roulette | Special | Special | Numerical chromosomes randomly created |
| SO | AP | AF | · *Chen* et al. [189] | 2009 | Error+Size | Roulette | Single-point | Flip-bit | Binary chromosomes randomly created |
| | | MF | · *Brunello* et al. [190] | 2017 | Accuracy,Size | Tournament | Special | Flip-bit | Binary chromosomes randomly created |
| | SF | UF | · *Crockett* et al. [185] | 1999 | Accuracy | | GENESIS genetic operators | | Numerical chromosomes randomly created |
| | | | · G-DT [159] | 2005 | Error | | GALib genetic operators | | Numerical chromosomes randomly created |
| | | | · IIVFDT [170] | 2012 | Accuracy | Tournament | Uniform | Uniform | Numerical chromosomes randomly created |

**Table 14**
Experimental analysis reported by LGA-based approaches for DTI.

| Strategy | DT | Studies | Datasets UCI | Datasets other | Sampling method | Performance measures | Statistical tests | Compared methods |
|---|---|---|---|---|---|---|---|---|
| RP | OB | · BTGA [154] | 1 | 3 | 3-f CV | Error, Size | - | C4.5 variant, Principal component analysis [194], Fisher's linear-ratio method [195] |
| | | · OC1-GA [11,137] | 10 | - | 5-f CV | Accuracy, Size, Time | *t*-test | CART, OC1, OC1-ES, OC1-SA |
| | | · *Krętowski* [33] | 4 | 4 | 10-f CV | Accuracy, Size, Time | - | OC1, OC1-GA |
| | | · RCGA-kDT [173] | - | 4 | 2-f CV | Accuracy, Time | - | C4.5, OC1, OC1-GA, OC1-ES, BTGA, Linear tree [146] |
| | | · *Pangilinan & Janssens* [180] | 6 | 1 | 5-f CV | Accuracy, Size, HV | - | C4.5 variant, OC1 |
| | | · HBDT [169] | 26 | - | 10-f CV | Accuracy, Size | ANOVA, Tukey | CART, OC1, OC1-AP, OC1-SA, OC1-GA, OC1-ES, GALE, GATree |
| | NL | · GA-QDT [161,183] | 2 | 2 | 10-f CV | Accuracy, Time | *t*-test | C4.5, C5.0, OC1, OC1-GA, OC1-ES, BTGA, Linear machine DT [196], Non-linear DT [197] |
| | | · NLDT [172] | 2 | 7 | Hold-out | Accuracy, Size | Wilcoxon | CART, SVM |
| | SF | · *Jakinow* [184] | - | 1 | Hold-out | Accuracy | - | - |
| | | · GA-FID3 [160] | 5 | - | - | Accuracy, Size | - | C4.5, Rank-based ID3 [198] |
| | | · GC-SDT [39] | 6 | - | 5-f CV | Error, Size | *t*-test | C4.5, Clustered-oriented fuzzy DT (C-fuzzy DT) method [199] |
| | | · FVBDT [158] | 8 | 2 | 10-f-CV | Accuracy | - | ID3, C-fuzzy DT, Merging branches method[200], *Yeung* et al. [201] |
| GS | AP | · Caltrop [7] | - | 1 | Complete dataset | Size | - | ID3 |
| | | · Bandar *et al.* [87] | 2 | - | Hold-out | Accuracy | - | - |
| | | · MEPDTI [171] | 11 | - | Hold-out | Accuracy | - | C4.5, CN2, BGP |
| | | · *Cha & Tappert* [175,176] | - | 1 | Complete dataset | Size | - | - |
| | | · *Smith* [177] | - | 3 | Hold-out | Time | - | - |
| | | · ECCO [155] | 4 | - | Hold-out | MC | - | Inexpensive classification with expensive tests method [202] |
| | | · EVO-Tree [157] | 7 | - | 5 × 2-CV | Accuracy, Size | Avg-R | C4.5, RTree, NB, MLP, SVM |
| | | · evtree [3] | 14 | 3 | Bootstrap | Accuracy, Size | - | CART, Conditional inference trees [45] |
| | | · BiLeGA [153] | 12 | - | Hold-out | Accuracy | *t*-test | C5.0, CART, SVM, ANN, LR |
| | | · *Ersoy* et al. [178] | 6 | - | 5-f CV | Accuracy, time | - | CART |
| | OB | · GDTI [167] | 11 | - | Hold-out | Accuracy | - | C4.5, CN2, BGP |
| | | · EFTI [156] | 26 | - | 10-f CV | Accuracy, Size | ANOVA, Tukey | CART, OC1, OC1-AP, OC1-SA, OC1-GA, OC1-ES, GALE, GATree, HDBT |
| | NL | · GALE [162,182] | 11 | - | 10-f CV | Accuracy | Avg, *t*-test | C4.5, CART, OC1 |
| | SF | · *Kym & Ryu* [188] | 7 | - | - | Accuracy, Size | Avg | *Yeung* et al. [201], *Abonyi* et al. [203]. |
| SO | AP | · *Chen* et al. [189] | 4 | - | Hold-out | Accuracy, Size | - | ID3 |
| | | · *Brunello* et al. [190] | 12 | - | Hold-out | Accuracy, Size | - | J48, C5.0 |
| | SF | · *Crockett* et al. [185] | - | 2 | Hold-out | Accuracy | - | ID3 |
| | | · G-DT [159] | 5 | - | 5-f CV | Error | - | C4.5 |
| | | · IIVFDT [170] | 17 | 3 | 5-f CV | Accuracy | Avg, Holm, Friedman, Wilcoxon | C4.5, GA-FID3, *Kym & Ryu* [188] |

**Table 15**
Genetic operators used by tree-based chromosomes.

| Acronym | Description |
|---|---|
| *Crossover operators:* | |
| Swap sub-trees | Swaps sub-trees randomly chosen from two DTs. |
| Swap nodes | Swaps nodes chosen at random from two DTs. Sub-trees of these nodes remain unchanged. |
| Swap branches | Swaps branches randomly selected from two DTs. |
| *Mutation operators:* | |
| Disturb node | Modifies some internal element of one node chosen at random. |
| Switch nodes | Swaps two nodes randomly chosen in the same DT. Sub-trees of these nodes remain unchanged. |
| Switch sub-trees | Swaps two sub-trees selected at random from the same DT. |
| Insert node | Adds a new node randomly created in the DT. |
| Replace node | Replaces one node randomly chosen in the DT. Sub-tree can be replaced with a leaf node, or a leaf node can be changed with a new sub-tree. |
| Replace sub-tree | Replaces one sub-tree randomly chosen in a DT with a new sub-tree. |

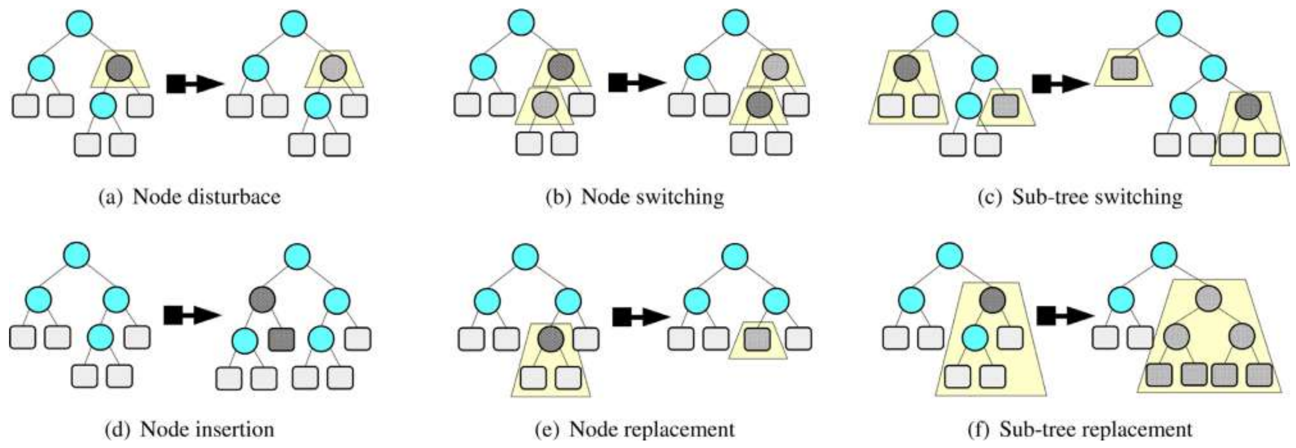

**Fig. 17.** Mutation operators used by tree-based chromosomes (Adapted from [163], [88], [204], [9], and [205]).

**Table 16**
Criteria used to handle categorical attributes with TGA-based methods.

| Criteria | Study |
|---|---|
| Multi-branching | GDT [9], GDT-MC [165], GDT-MA [164], genTrees [168], LEGAL-Tree [10], *Bosnjak* et al. [210], *Biedrzycki & Arabas* [215] |
| Binary-branching | GATree [163] |

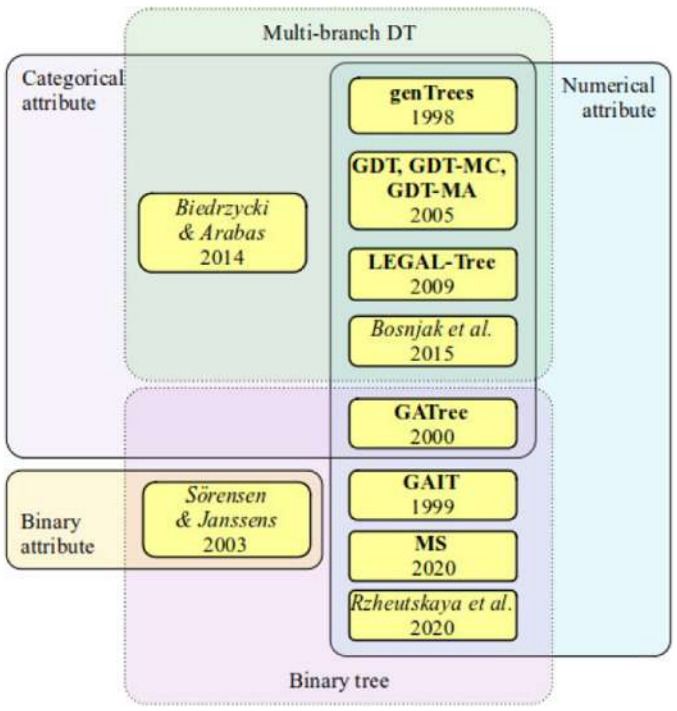

**Fig. 18.** DTs and attribute types used with TGA-based MHs inducing axis-parallel-DTs.

Components of TGA-based approaches for DTI are shown in Table 17. Single-objective and aggregating fitness functions are the evaluation schemes most used in these studies, and only one applies a multi-objective fitness function. Accuracy and size are the fitness measures most used in these works.

Tournament-based selection is used in three studies, and roulette-wheel and linear-ranking-based selection operators have each been implemented in four studies. All studies apply sub-tree-swapping-based crossover, but node-swapping and branch-swapping have also been used. However, *Biedrzycki & Arabas* do not apply one crossover operator. They only use a node-insertion-based mutation to configure initial DTs. Furthermore, DTs' random creation is the most common scheme to build the initial population. However, GAIT uses C4.5 to inducing DTs with instance subsets chosen at random. LEGAL-tree uses a random combination of several decision stumps previously created.

Table 18 shows the experimental studies reported in the existing literature. K-fold CV and hold-out and test accuracy and size are the sampling strategies and the performance measures most used in these methods, respectively. *Bosnjak* et al. and *Basgalupp* et al. adopt F-score as their performance measure, and only GDT-MC implements a cost-sensitive classification approach. Finally, eight studies describe applying a statistical test, and only LEGAL-Tree applies one post-hoc analysis.

The use of tree-like chromosomes is an option to deal with the variable size of DTs. It is guaranteed that there is no loss of information since no mapping scheme is necessary to convert a chromosome into a DT. However, their use in GAs implies the definition of several variation operators to generate feasible solutions. Furthermore, it is necessary to apply two or more genetic operators to avoid the evolutionary process's stagnation.

### 5.3. Genetic programming (GP)

Fig. 19 shows the timeline of GP-based approaches for DTI. Table 19 illustrates the acronyms used to identify various GP-based MHs for DTIs, as defined by their authors. Standard genetic programming has been applied in many studies, but two GP variants have also been used: Grammar-based genetic programming (GGP) and Strongly-typed genetic programming (TGP). In GGP, a Backus-Naur form (BNF) grammar allows formally defining GP structures and ensuring that the typing and syntax are maintained while manipulating the syntactic tree [71]. In the other case, TGP is an enhanced GP version applying data type constraints [73].

#### 5.3.1. Axis-Parallel DTs

*Koza* [4] proposes encoding DTs with Lisp S-Expressions (S-Exps), where internal nodes are built with a function set and leaf nodes with a terminal set. S-Exps are used by *Iba* et al. [247] and *Tür & Güvenir* [89]. Function and terminal sets are also used in other studies (Table 20), such as:

**Table 17**
Components of TGA-based approaches for DTI.

| Strategy | DT | FF | Studies | Year | Fitness measure | Genetic operators: Selection | Genetic operators: Crossover | Genetic operators: Mutation | Initial population |
|---|---|---|---|---|---|---|---|---|---|
| RP | AP | UF | · *Rzheutskaya* et al. [216] | 2020 | Accuracy | Special | Swap sub-trees | Disturb node | DTs randomly created |
| GS | AP | UF | · GAIT [102] | 1999 | Accuracy | - | Swap sub-trees | Switch nodes | DTs induced by C4.5 with a subset of training instances randomly chosen |
| | | | · GAIT [204,211–214] | 2000 | Accuracy | Roulette | Swap sub-trees | Switch sub-trees | DTs induced by C4.5 with a subset of training instances randomly chosen |
| | | | · *Sörensen & Janssens* [88] | 2003 | Accuracy | Roulette | Swap sub-trees, Swap nodes | Switch nodes, Switch sub-trees | Binary DTs randomly created |
| | | | · *Bosnjak* et al. [210] | 2015 | Accuracy, Size, F-Score | Tournament | Swap sub-trees | Disturb node | DTs randomly created |
| | | AF | · genTrees [168] | 1998 | Error+Size | Exponential ranking | Swap sub-trees | Disturb node | DTs randomly created |
| | | | · GATree [163,206] | 2000 | Accuracy+Size | - | Swap sub-trees | Disturb node | Decision stumps randomly created |
| | | | · GDT [9] *Jurczuk* et al. [207–209] | 2005 2017 2021 | Accuracy+Size | Linear ranking | Swap sub-trees, Swap nodes, Swap branches | Disturb node, Switch nodes, Replace node, Switch sub-trees | DTs randomly created |
| | | | · *Biedrzycki & Arabas* [215] | 2006 | Error+Size | Tournament | N/A | Insert node | Empty DTs |
| | | | · GDT-MC [165] | 2007 | MC+Size | Linear ranking | Swap sub-trees, Swap nodes, Swap branches | Disturb node, Switch nodes, Replace node, Switch sub-trees | DTs randomly created |
| | | | · GDT-MA [164] | 2008 | Accuracy+Size | Linear ranking | Swap sub-trees, Swap nodes, Swap branches | Disturb node, Switch nodes, Replace node, Switch sub-trees | DTs induced with 10% of training set using several splitting criteria (IG, GR, Gini, and Dipolar) |
| | | | · MS [113] | 2020 | Accuracy+Stability | Roulette | Swap sub-trees | Replace node | DTs with a single leaf only |
| | | MF | · LEGAL-Tree [10,205] | 2009 | Accuracy, Size | Tournament | Swap sub-trees | Replace node | DTs created with a random combination of several decision stumps previously induced with 10% of training set |
| | OB | UF | · TARGET [174] | 2008 | Error | Roulette | Swap sub-trees, Swap nodes | Disturb node, Switch nodes | DTs randomly created |
| | | AF | · GEA-ODT [217,218] | 2005 | Accuracy+Size | Linear ranking | Swap sub-trees, Swap nodes, Swap branches | Disturb node, Switch nodes, Replace node, Switch sub-trees | DTs randomly created |
| | | | · GDT-Mix [166] *Krętowski & Popczyński* [219] *Czajkowski* et al. [114] *Reska* et al. [220] | 2006 2008 2015 2018 | Accuracy+Size | Linear ranking | Swap sub-trees, Swap nodes | Disturb node, Switch nodes, Replace node, Switch sub-trees | DTs randomly created |

1. The Evolutionary Dynamic Data Investment Evaluator (EDDIE) system of *Tsang* et al. [230] where leaf nodes are encoded as terminals with indicators of financial forecasting.
2. The Evolutionary Multi-objective Optimization (EMO) of *Kim* [232,248].
3. The GPDTI method of *Estrada-Gil* et al. [239].
4. The work of *Niimi & Tazaki* [249,250] where the initial population is created using the Apriori algorithm [251].

Other GP-based methods for DTI build their trees without using the function set and the terminal set. *Shirasaka* et al. [263] define a DT as a list of nodes, each one with references to one predecessor node and two successor nodes. Each node is a 6-tuple $\{t, l, p, l, r, c\}$ where $t$ is the node id, $l$ is the class label, $p$, $l$ and $r$ are pointers to one parent and two children, and $c$ is a counter set defining internal nodes. This representation scheme is used in the following studies:

1. *Zhao & Shirasaka* [264] evaluate several alternatives for inducing more compact DTs, such as controlling DT size in the selection process and deleting redundant elements.
2. *Oka & Zhao* [103] apply C4.5 with segments of the training set to build the initial population.
3. *Tanigawa & Zhao* [265] use GP to induce small subtrees, which are combined to build a complete DT.
4. *Haruyama & Zhao* [266] implement three multi-objective GA-based methods for DTI.

Furthermore, a simple DT encoding scheme is applied in the Evolutionary Programming Tree (EPTree) of *DeLisle & Dixon* [233], as well as by *Buontempo* et al. [267] and *Wang* et al. [268].

**Table 18**
Experimental analysis reported by TGA-based approaches for DTI.

| Stra-tegy | DT | | Studies | Datasets UCI | Datasets other | Sampling method | Performance measures | Statistical tests | Compared methods |
|---|---|---|---|---|---|---|---|---|---|
| RP | AP | · | *Rzheutskaya* et al. [216] | 10 | - | - | Accuracy | - | - |
| GS | AP | · | genTrees [168] | - | 1 | Hold-out | Accuracy, Size | - | Traditional DTI method |
| | | · | GAIT [102] | - | 1 | Hold-out | Accuracy, Time | - | C4.5 variant [102] |
| | | · | GATree [163] | 13 | 5 | 5-f CV | Accuracy, Size | Avg | C4.5, One-rule method [221] |
| | | · | GAIT [211,212] | - | 1 | Hold-out | Accuracy, Time | - | C4.5 |
| | | · | *Sörensen & Janssens* [88] | - | 1 | - | Accuracy | - | - |
| | | · | GAIT [213] | - | 1 | Hold-out | Accuracy, Time | *t*-test | LR |
| | | · | GAIT [204] | 1 | 2 | Hold-out | Accuracy, Size, Time | *t*-test | C4.5 |
| | | · | GDT [9] | 8 | 3 | 10-f CV | Accuracy, Size | - | C4.5 |
| | | · | *Jurczuk* et al. [207,208] | 2 | 1 | Complete dataset | Speedup | - | - |
| | | · | *Jurczuk* et al. [209] | 2 | 1 | Complete dataset | Time | - | J48, GDT |
| | | · | GAIT [214] | 3 | 2 | Hold-out | Accuracy, Size | *t*-test | C4.5 |
| | | · | *Biedrzycki & Arabas* [215] | 4 | 2 | 10-f CV | Error, Size | - | ID3, J48 |
| | | · | GDT-MC [165] | 13 | - | 10-f CV | MC, Size | Avg | C5.0, Cost-sensitive J48 [52], MetaCost [222] |
| | | · | GDT-MA [164] | 15 | - | 10-f CV | Accuracy, Size | - | C4.5, GDT |
| | | · | LEGAL-Tree [10] | 6 | - | 10-f CV | Accuracy, Size | *t*-test | J48 |
| | | · | GATree [206] | 12 | - | 5-f CV | Accuracy, Size, Time | - | J48 |
| | | · | LEGAL-Tree [205] | 16 | - | 10-f CV | Accuracy, F-Score, Size, Time | Friedman, Nemenyi | J48, CART, GALE |
| | | · | *Bosnjak* et al. [210] | 20 | - | Hold-out | Accuracy, Size, F-Score | Kruskall | GATree |
| | | · | MS [113] | 4 | 6 | Hold-out | Accuracy, Stability | - | RF |
| | OB | · | GEA-ODT [217,218] | 10 | 5 | 10-f CV | Accuracy, Size | - | C4.5, OC1 |
| | | · | GDT-Mix [166] | 10 | 15 | 10-f CV | Accuracy, Size | - | C4.5, OC1, GEA-ODT, GDT |
| | | · | *Krętowski & Popczyński* [219] | 10 | 10 | 10-f CV | Accuracy, Size | - | C4.5, OC1, GDT-Mix |
| | | · | *Czajkowski* et al. [114] | - | 4 | Complete dataset | Speedup | - | - |
| | | · | *Reska* et al. [220] | 2 | 1 | Complete dataset | Speedup | - | - |
| | | · | TARGET [174] | 3 | 2 | 10-f CV | Error, Size | - | CART, QUEST, CRUISE, RF, Bayesian CART [223], Bootstrap bumping [224] |

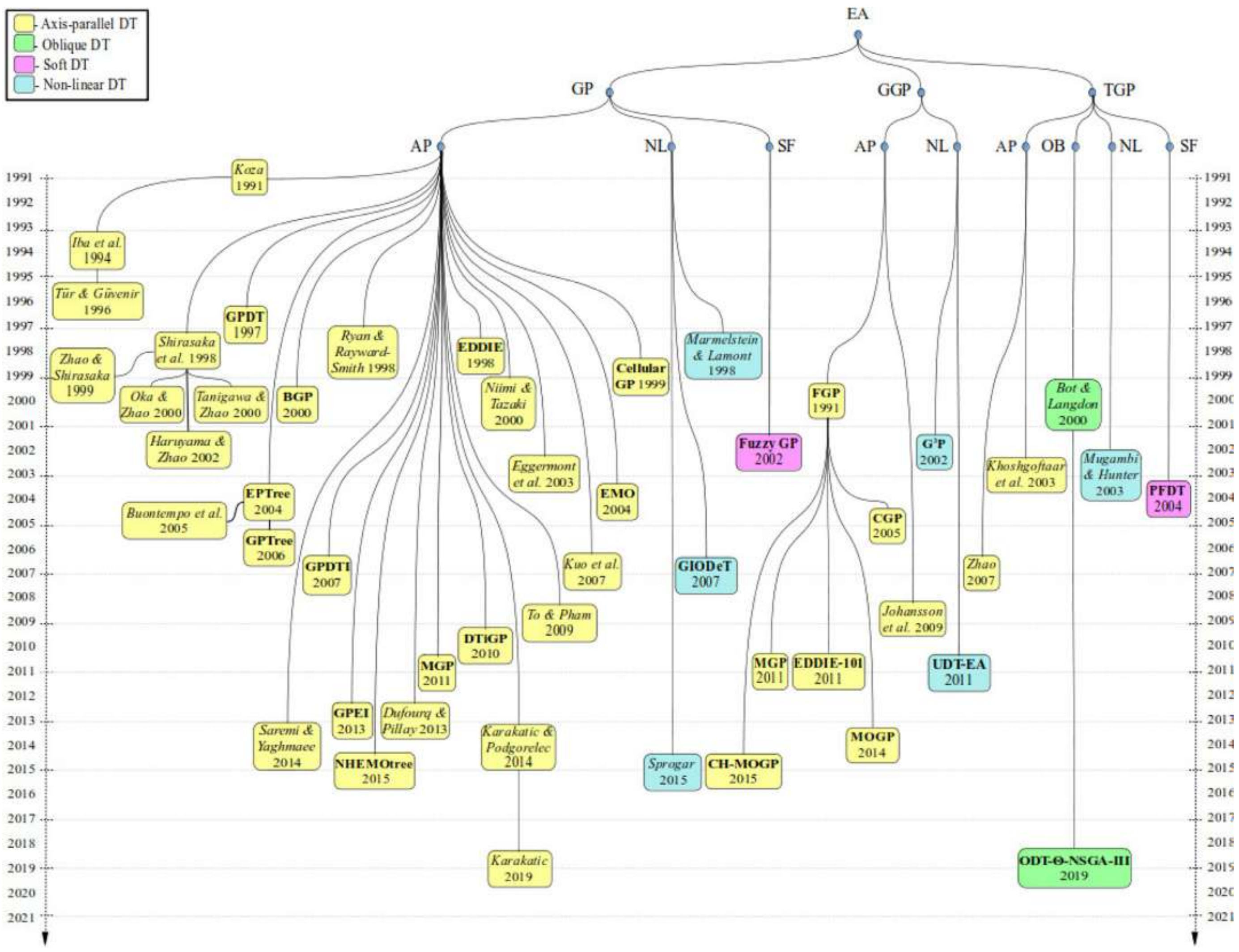

**Fig. 19.** Timeline of GP-based approaches for DTI.

**Table 19**
Acronyms used to identify several GP-based MH for DTI.

| Acronym | Description |
|---|---|
| BGP | Building Block approach for GP [225] |
| Cellular GP | Cellular GP algorithm [226] |
| CGP | Constrained GP [227] |
| CH-MOGP | Convex-Hull-based MOGP [228] |
| DTiGP | DT Injection GP [229] |
| EDDIE | Evolutionary Dynamic Data Investment Evaluator [230] |
| EDDIE-101 | EDDIE-101 [231] |
| EMO | Evolutionary Multi-objective Optimization [232] |
| EPTree | Evolutionary Programming Tree [233] |
| FGP | Financial GP [234] |
| fuzzy-GP | Fuzzy GP [235] |
| $G^3P$ | Grammar-guided GP [236] |
| GIODeT | GP for Induction of Oblique DTs [237] |
| GPDT | GP for DT [238] |
| GPDTI | GP for DTI [239] |
| GPEI | GP Evolved Intervals [240] |
| M-GP | Memetic GP [241] |
| MGP | Multiage GP [242] |
| MOGP | Multi-objective GP [243] |
| NHEMOtree | Nonhierarchical Evolutionary MO tree learner [244] |
| ODT-Θ-NSGA-III | ODT-based-Θ-Nondominated Sorting GA-III [245] |
| PFDT | Polynomial-Fuzzy DTs [246] |
| UDT-EA | Unconstrained DT-EA [231] |

*Rouwhorst & Engelbrecht* [225] and *Engelbrecht* et al. [269] implement the Building Block approach for GP (BGP) to encode DTs with test conditions as *building blocks* as Eqn. (5).

$$\left(x, op, \{x|t\}\right) \tag{5}$$

where $x$ is an attribute, $t$ is a threshold value, and $op$ can be $\{=, \neq, <, \leq, >, \geq\}$. By starting with a population of decision stumps, BGP adds new nodes in its evolutionary process. BGP also applies one pruning operator.

A similar encoding scheme called *full atomic representation* is described by *Eggermont* et al. [270,271] where DTs evolve through a multi-layered fitness function of two ranked fitness measures. They first use an information-theory-based splitting criterion [270] and then develop a refined atomic representation using bounded values for numerical attributes [271].

Some GP-based methods introduce specialized operators to build feasible and more accurate offspring.

1. GP for DT (GPDT) method of *Nikolaev & Slavov* [238] implement a depth-first search strategy for its mutation operator.
2. *Ryan & Rayward-Smith* [252] use the training set to refine the offspring generated in the crossover stage.
3. *Kuo* et al. [257] introduce operators to eliminate redundant sub-trees and to remove subsumed sub-trees.

**Table 20**
Structure of internal and leaf nodes used in GP-based approaches for DTI.

| Study | Function set | Terminal set |
|---|---|---|
| *Axis-parallel DT:* | | |
| · *Koza* [4] | Attributes as functions with arguments | Class labels |
| · *Iba* et al. [247], *Tür & Güvenir*[89], *Ryan & Rayward-Smith* [252] | Attributes as boolean functions | Class labels |
| · GPDT [238], Cellular GP [226,253], *To & Pham* [254] | Attributes | Class labels |
| · EDDIE [230], FGP [5,234] | $\{$*if-then-else*$, \wedge, \vee, \neg, >, <\}$ | Attributes, class labels, real-valued values |
| · EMO [232,248] | A comparison operator for an attribute and its threshold | Class labels |
| · GPDTI [239] | Functions with arguments | Class labels |
| · *Niimi & Tazaki* [250] | $\{$*if-less-than*, *if-equals*$, *, /, +, -\}$ | Attributes, class labels |
| · *Khoshgoftaar* et al. [255,256] | $\{$*if*$, <\}$, class labels | Attributes, real-valued values |
| · Kuo *et al.* [257] | $\{$*if-then*, *if-then-else*$, \wedge, \vee, \neg, >, <, \geq, \leq\}$. | Attributes, class labels |
| · *Zhao* [258] | A empty node. | Attributes, class labels, real-valued values |
| · *Johansson* et al. [259,260], DTiGP [229] | $\{$*if*$, >, <, =\}$ | Attributes, class labels, real-valued values |
| · MGP [241], EDDIE-101 [231], MOGP [243] | $\{$*if-then-else*$, \wedge, \vee, \neg, >, <, =\}$ | $\{0, 1\}$ |
| *Oblique DT:* | | |
| · *Bot & Langdom* [6,261] | A label indicating the number of attributes in the linear combination (1, 2, 3). | Attributes, class labels, real-valued values |
| *Non-linear DTs:* | | |
| · *Marmelstein & Lamont* [237] | $\{+, -, \times, \div, \leq\}$. | Attributes, a random value generator |
| · UDT-EA [231] | $\{$*if-then-else*$, +, -, *, \wedge, \vee, \neg, >, <, \geq, \leq, =, \leq\}$. | Attributes, class labels, real-valued constants |
| · *Šprogar* [262] | $\{$*if*$, +, -, *, /, 0, eph\}$, attributes. | $\{modus\}$ |

4. *Dufourq & Pillay* [272] use an encapsulation operator to preserve promising sub-trees.

Also, a migration operator is periodically used to swap DTs placed in a bi-dimensional grid with the Cellular GP proposed by *Folino* et al. [226,253]. *To & Pham* [254] use a similar operator to exchange chromosomes between sub-populations evolving in an island-based parallel GP.

More recently, two methods improve the crossing point determination of the recombination operator:

1. *Karakatič & Podgorelec* [273] select individuals based on the accuracy and usage of the internal node. The usage of one node is a percentage of training instances processed in it.
2. The Nonhierarchical Evolutionary Multi-Objective tree learner (NHEMOtree) of *Casjens* et al. [244] use the importance of an attribute. They argue that an attribute is more important than the others if it is near the root node and is more likely to be selected as a crossover point.

Several GP-based approaches for DTI have been implemented to avoid premature convergence and improve the model performance. In the DT Injection GP (DTiGP), *König* et al. [229] use an adaptive fitness function to include diversity in the population and control the size model. First, the fitness value of a DT induced using a subset of instances randomly chosen is designated as a reference value. Next, the evolutionary process starts. At even intervals, the fitness value of the best DT is compared with the reference value. If the first is less than the second, the fitness function is modified to create larger DTs. Furthermore, *Yi & Wanli* [242] implement the multiage GP (MGP) to evolve groups of DTs according to their number of leaf nodes (ages). MGP first divides the population using their ages and then evolves each group independently. These evolved groups are combined to build a new population. This grouping scheme reduces the selection pressure in a particular area (group) and tries preventing genetic operators destroy the evolutionary process continuity. Finally, *Karakatič* et al. [274] introduce a Lazy Evaluation Model of the fitness value considering only DTs used in the selection procedure to reduce the processing time. They also define a strategy to assign weights to the instances to be used in the DT evaluation.

Additionally, the discretization of real-valued attributes has been implemented in the following methods: The GP Evolved Intervals (GPEI) approach of *Dufourq & Pillay* [240] including an adaptive discretization with fixed and varying number of intervals. The work of *Saremi & Yaghmaee* [275,276] introduce specialized operators to modify threshold values.

Alternatively, GGP has been implemented in the following DTI approaches:

1. The Financial GP (FGP) described by *Li* [234] and *Tsang & Li* [5].
2. The Constrained GP (CGP) of *Li* et al. [227] for cost-sensitive classification.
3. The Memetic GP (MGP) and the EDDIE-101 system of *Wang* et al. [231,241] where class labels are replaced by instance counters. EDDIE-101 also uses a local search to improve the threshold values of each test condition.
4. The multi-objective GP (MOGP) [243] and the Convex-Hull-based MOGP (CH-MOGP) [228] where the ROC curve is used in the evolutionary process.
5. The methods of *Johansson & Niklasson* [259], and *Johansson* et al. [260] where one oracle data[6] is used to improve the DT performance.

Finally, TGP has been used: (1) by *Khoshgoftaar* et al. [255] and *Khoshgoftaar & Liu* [256] in a multi-objective method, and (2) by *Zhao*

[6] An oracle data is a set of test instances together with their predictions (class labels) obtained using another classifier method (ANN or RF).

**Table 21**
Criteria used to handle categorical attributes with GP-based methods.

| Criteria | Study |
|---|---|
| Multi-branching | *Koza* [4] |
| Binary mapping | *Niimi & Tazaki* [249], BGP [225], *Johansson* et al. [260], MOGP [243], *Ryan & Rayward-Smith* [252], *Eggermont* et al. [270], *Kuo* et al. [257] |

[258] in a MOGA-based application to select partial preferences on conflicting objectives.

### 5.3.2. Oblique DTs

*Bot & Langdon* [6,261] apply TGP to encode the hyperplanes of an oblique DT in a global-search strategy. The function set allows creating test conditions using either one attribute or a linear combination of two or three attributes. This representation is used in the ODT-Θ-NSGA-III (ODT-based-Θ-Nondominated Sorting GA-III) of *Chabbouth* et al. [245].

### 5.3.3. Non-linear DTs

Two similar GP-based approaches implementing a recursive-partitioning strategy to find a near-optimal hypersurface in each test condition of a non-linear DT are described by *Shali* et al. [277] in the GP for Induction of Oblique DTs (GIODeT) method, and by *Marmelstein & Lamont* [237].

Alternatively, a global search of non-linear DTs also has been implemented in four methods:

1. The Unconstrained DT-EA (UDT-EA) implemented by *Wang* et al. [231] uses hypersurfaces in test conditions, and that described by *Šprogar* [262] using the node information-gain-value in its prudent-crossover operator.
2. The Grammar-guided GP (G$^3$P) method of *Tsakonas* [236,278].
3. *Mugambi & Hunter* [279] implement TGP in a multi-objective approach evolving a population of DTs to find Pareto optimum values.

### 5.3.4. Soft DTs

*Eggermont* [235] applies his *full atomic representation* in the fuzzy-GP method to evolve a population of soft DTs with triangular membership functions. Moreover, *Mugambi* et al. [246] use TGP in the Polynomial-Fuzzy DTs (PFDT) method to find near-optimal hypersurfaces of a non-linear DT. PFDT evolves both polynomials representing the hypersurfaces and sigmoid and bell-shaped membership functions associated with numerical attributes.

### 5.3.5. Discussion

Fig. 20 shows the attribute and branch types of the axis-parallel DTs induced with the studies described previously. Two criteria have been used to handle categorical attributes: multi-branching and binary mapping (Table 21). In particular, although several methods manipulate numeric attributes, they are first discretized to be used as categorical attributes in GPDT, Cellular GP, and GPEI, and that of *Saremi & Yaghmaee*.

Table 22 shows the components of GP-based approaches for DTI. Twenty studies implement a single-objective fitness function, 17 use an aggregating fitness function, and the remaining studies evaluate a multi-objective fitness function. Accuracy and tree size are the fitness measure most commonly applied to these methods.

Furthermore, sub-tree-swapping crossover and sub-tree-replacement mutation operators are typically applied in these algorithms, although the node-perturbation is also used. The random creation of the initial population and the ramped half-and-half (RH&H) criterion [152] are used to generate the majority's initial candidate solutions of these studies. However, some apply alternative strategies, such as using C4.5 with a collection of instances randomly chosen from the training set, creating decision stumps, and mapping DTs from rules previously created.

Table 23 shows the experimental studies' elements conducted by these approaches. K-fold CV and hold-out sampling methods are used in 23 studies each. Three works evaluate the complete datasets to compute the algorithm performance.

Accuracy, size, and error-rate are adopted as performance measures in most of these methods. Misclassification cost, sensitivity, specificity, fidelity, AUC, and ROC curve analysis have been used as performance measures in several methods. Furthermore, 14 studies conducted a statistical test to compare their experimental results with those obtained from other methods, Finally, two studies implement some post-hoc analysis only.

Genetic programming is the EA most used to induce DTs since it can evolve DTs directly, and a mapping scheme to build a DT from a chromosome is not needed. In the previous paragraphs, diverse directions to improve genetic programming capacities to induce near-optimal DTs have been described, such as avoiding premature convergence, improving diversity, and reducing the bloat problem. The first two problems are common to all EAs. However, bloat is a particular GP problem, occurring when the evolved trees grow without improving the model performance.

## 5.4. Other EA-based approaches for DTI

Although genetic algorithms and genetic programming are the most used MHs to create DTI methods, other EAs such as coevolutionary algorithms, estimation of distribution algorithms, grammatical evolution, gene expression programming, differential evolution, and the evolution strategies have also been used for the same purpose. The first two encode their chromosomes as trees, and the others use linear chromosomes. Fig. 21 shows a timeline of the other EA-based approaches applied to induce DTs. Table 24 illustrates the acronyms used to identify various EA-based MHs for DTIs, as defined by their authors.

### 5.4.1. Axis-parallel-DTs

*Coevolutionary algorithms: Podgorelec & Kokol* [286,298], *Babič* et al. [299], and *Zorman* et al. [300] describe several versions of the Self-adapting Evolutionary DT (SEADT) method, a competitive-CEA-based approach to build DTs. SAEDT uses a tree-based chromosome with three independent populations competing to find a near-optimal DT (Fig. 22). One population contains only one DT induced by C4.5, and the others evolve DT populations. The best chromosome from the main population competes with the best one from the others. When it becomes dominant over the others, the fitness functions are adjusted in each population. A global fitness value is used to determine the dominance of one population over the others.

Furthermore, *Aitkenhead* [301] implements competition between one DT and the training set. This algorithm mutates the tree nodes while updating the training set size and the DT-depth. In the beginning, the training set contains only two sets randomly chosen from the dataset. Once DT evolves and its fitness value becomes higher than a threshold value, the sets are increased. This process continues until the entire dataset is used to evaluate the DT.

Alternatively, the multi-population GA for DTI (MPGA) of *Podgorelec & Karakatic* [284] and the multi-population genTrees (MPGT) method of *Podgorelec* et al. [285] are cooperative CEA using two DTs sub-populations. After a predefined number of generations, one DTs exchange between populations occurs according to one migration rate. The first sub-population uses accuracy, and the other applies one balanced single-class accuracy as their fitness values.

*Differential Evolution: Rivera-López & Canul-Reich* introduce the DE-based approach to build axis-parallel-DTs using the smallest-position-

**Table 22**
Components of GP-based approaches for DTI.

| Strategy | DT | FF | MH | | Studies | Year | Fitness measure | Genetic operators | | | Initial population |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | Selection | Crossover | Mutation | |
| RP | NL | AF | GP | · | *Marmelstein & Lamont* [237] | 1998 | Error+Size | Tournament | Swap sub-trees | Replace sub-tree | S-Exps randomly created |
| | | | | · | GIODeT [277] | 2007 | GR+Size | Tournament | Swap sub-trees | Replace sub-tree | ExpTs randomly created |
| GS | AP | UF | GP | · | *Koza* [4] | 1991 | Accuracy | Tournament | Swap sub-trees | Replace sub-tree | - |
| | | | | · | *Iba* et al. [247] | 1994 | MDL | Tournament | Swap sub-trees | Replace node, Replace sub-tree | S-Exps randomly created |
| | | | | · | GPDT [238] | 1997 | MDL | Roulette | Swap sub-trees | Special | DTs randomly created |
| | | | | · | EDDIE [230] | 1998 | Accuracy | Tournament | Swap branches | Replace sub-tree | - |
| | | | | · | *Shirasaka* et al. [263] | 1998 | Accuracy | Truncate | Swap sub-trees | Disturb node | - |
| | | | | · | Cellular GP [226,253] | 1999 | J-Measure | Special | Swap sub-trees | Replace sub-tree | DTs randomly created |
| | | | | · | *Zhao & Shirasaka* [264] | 1999 | Accuracy | Truncate | Swap sub-trees | Disturb node | DTs randomly created |
| | | | | · | *Oka & Zhao* [103] | 2000 | Accuracy | Truncate | Swap sub-trees | Disturb node | DTs created using C4.5 |
| | | | | · | *Tanigawa & Zhao* [265] | 2000 | Accuracy | Truncate | Swap sub-trees | Disturb node | DTs randomly created |
| | | | | · | BGP [225,269] | 2000 | Accuracy | Tournament | Swap sub-trees | Disturb node | Decision stumps randomly created |
| | | | | · | *Niimi & Tazaki* [250] | 2000 | Accuracy | Tournament | Swap sub-trees | Replace sub-tree | DTs created using Apriori |
| | | | | · | EPTree [233] | 2004 | MDL | Tournament | Swap sub-trees | Disturb node, Replace sub-tree | DTs randomly created |
| | | | | · | GPTree [267,268] | 2005 | Accuracy | Tournament | Swap sub-trees | Disturb node, Replace sub-tree | DTs randomly created |
| | | | | · | *To & Pham* [254] | 2009 | Accuracy | Truncate | Swap sub-trees | N/A | DTs randomly created |
| | | | | · | GPEI [240,272] | 2013 | Accuracy | Tournament | Swap sub-trees | Replace sub-tree | RH&H criterion |
| | | | | · | *Karakatič & Podgorelec* [273] | 2014 | Accuracy | Tournament | Special | Special | DTs randomly created |

(*continued on next page*)

**Table 22** (*continued*)

| Stra-tegy | DT | FF | MH | Studies | Year | Fitness measure | Genetic operators | | | Initial population |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | Selection | Crossover | Mutation | |
| | | | GGP | · FGP [5,234] | 2000 | Accuracy | Tournament | Swap sub-trees | Replace sub-tree | RH&H criterion |
| | | | | · *Johansson* et al. [260] | 2010 | Accuracy | Roulette | Swap sub-trees | Replace sub-tree | RH&H criterion |
| | | AF | GP | · *Tür & Güvenir* [89] | 1996 | Accuracy+Size | - | Swap sub-trees | Replace sub-tree | DTs randomly created |
| | | | | · *Ryan & Rayward-Smith* [252] | 1998 | Error+Size | - | Special | N/A | DTs created using C4.5 |
| | | | | · *Niimi & Tazaki* [249] | 1999 | Accuracy+Size | Tournament | Swap sub-trees | Replace sub-tree | DTs created using C4.5 |
| | | | | · GPDTI [239] | 2007 | Accuracy+Size | Tournament | Swap sub-trees | Replace sub-tree | DTs randomly created |
| | | | | · *Kuo* et al. [257] | 2007 | Accuracy+Size | Roulette | Swap sub-trees | Replace sub-tree | DTs randomly created |
| | | | | · DTiGP [229] | 2010 | Error+Size | Roulette | Swap sub-trees | Replace sub-tree | RH&H criterion |
| | | | | · MGP [242] | 2011 | Accuracy+Size | - | - | - | DTs randomly created |
| | | | | · *Saremi & Yaghmaee* [275,276] | 2014 | Accuracy+Size | Tournament | Swap sub-trees | Disturb node, Replace sub-tree | DTs randomly created |
| | | | | · *Karakatič* et al. [274] | 2019 | Accuracy+Size | Tournament | Swap sub-trees | Special | DTs randomly created |
| | | | GGP | · CGP [227] | 2005 | Accuracy+Fall-out | Tournament | Swap sub-trees | Replace sub-tree | RH&H criterion |
| | | | | · *Johansson* et al. [259] | 2009 | Accuracy+Size | Roulette | Swap sub-trees | Replace sub-tree | RH&H criterion |
| | | | | · MGP [241] | 2011 | IG+Size | Tournament | Swap sub-trees | Special | RH&H criterion |
| | | | | · EDDIE-101 [231] | 2011 | IG+Size | Tournament | Swap sub-trees | Replace sub-tree | RH&H criterion |
| | | MF | GP | · *Haruyama & Zhao* [266] | 2002 | Accuracy, Size | Tournament | Swap sub-trees | Disturb node | DTs randomly created |
| | | | | · *Eggermont* et al. [270,271] | 2003 | Error, Size | Tournament | Swap sub-trees | Replace sub-tree | RH&H criterion |
| | | | | · EMO [232,248] | 2004 | Error, Size | Tournament | Swap sub-trees | Disturb node, Replace sub-tree, Replace node | DTs randomly created |
| | | | | · NHEMOtree [244] | 2015 | Error, MC | Tournament | Special | Disturb node, Switch nodes, Replace node, Switch sub-trees, Replace sub-tree | RH&H criterion |
| | | | GGP | · MOGP [243] | 2014 | Sensitivity, Fall-out | Tournament | Swap sub-trees | Special | RH&H criterion |
| | | | | · CH-MOGP [228] | 2015 | Sensitivity, Fall-out | Tournament | Swap sub-trees | Special | RH&H criterion |
| | | | TGP | · *Khoshgoftaar* et al. [255,256] | 2003 | Error, Size | Tournament | Swap sub-trees | Replace sub-tree | RH&H criterion |
| | | | | · *Zhao* [258] | 2007 | Fall-out, Miss-rate, Precision, Sensitivity, Specificity | Tournament | Swap sub-trees | Replace sub-tree | DTs randomly created |
| | OB | UF | TGP | · *Bot & Langdon* [6] | 2000 | Error | Tournament | Swap sub-trees | Replace sub-tree | RH&H criterion |
| | | MF | TGP | · *Bot & Langdon* [261] | 2000 | Error, Size | Tournament | Swap sub-trees | Replace sub-tree | RH&H criterion |
| | | | | · ODT-Θ-NSGA-III [245] | 2019 | Precision, Sensitivity | Special | Swap sub-trees | Replace sub-tree | DTs randomly created |
| | NL | UF | GP | · *Šprogar* [262] | 2015 | Accuracy | Tournament | Special | - | RH&H criterion |
| | | AF | GGP | · $G^3P$ [236,278] | 2002 | Accuracy+Size | Tournament | Swap sub-trees | Replace sub-tree | DTs randomly created |
| | | | | · UDT-EA [231] | 2011 | IG+Size | Tournament | Swap sub-trees | Special | RH&H criterion |
| | | MF | TGP | · *Mugambi & Hunter* [279] | 2003 | Sensitivity, Specificity, Size | Tournament | Swap sub-trees | Replace sub-tree | DTs randomly created |
| | SF | MF | GP | · Fuzzy-GP [235] | 2002 | Accuracy, Size | Tournament | Swap sub-trees | Replace sub-tree | RH&H criterion |
| | | | TGP | · PFDT [246] | 2004 | Sensitivity, Specificity, Size | Tournament | Swap sub-trees | Replace sub-tree | DTs randomly created |

**Table 23**
Experimental evaluation reported by GP-based approaches for DTI.

| Stra-tegy | DT | MH | Studies | Datasets UCI | Datasets other | Sampling method | Performance measures | Statistical tests | Compared methods |
|---|---|---|---|---|---|---|---|---|---|
| RP | NL | GP | · *Marmelstein & Lamont* [237] | 2 | 1 | Hold-out | Error | - | RBF-NN, MLP, GP |
| | | | · GIODeT [277] | 19 | - | 5-f CV | Accuracy, Size | Avg | C4.5 |
| GS | AP | GP | · *Koza* [4] | - | - | - | - | - | - |
| | | | · *Iba* et al. [247] | - | 2 | Hold-out | Accuracy | - | ANN |
| | | | · *Tür & Güvenir* [89] | - | 1 | Hold-out | Accuracy, Time | - | - |
| | | | · GPDT [238] | 1 | 11 | Hold-out | Accuracy, Size | - | C4.5 |
| | | | · EDDIE [230] | - | 2 | Hold-out | Accuracy | - | - |
| | | | · *Shirasaka* et al. [263] | - | 1 | Complete dataset | Accuracy, Size | - | - |
| | | | · *Ryan & Rayward-Smith* [252] | 6 | - | Hold-out | Error, Size | - | C4.5 |
| | | | · Cellular GP [226,253] | 5 | - | Hold-out | Error | - | C4.5 |
| | | | · *Zhao & Shirasaka* [264] | - | 1 | Hold-out | Accuracy, Size | - | - |
| | | | · *Oka & Zhao* [103] | - | 1 | Hold-out | Accuracy, Size | - | C4.5, *Shirasaka* et al. [263] |
| | | | · *Tanigawa & Zhao* [265] | - | 1 | Hold-out | Accuracy, Size | - | - |
| | | | · BGP [225,269] | 4 | - | Hold-out | Accuracy | *t*-test | C4.5, CN2 |
| | | | · *Niimi & Tazaki* [249] | 1 | 1 | Hold-out | Accuracy, Size | - | C4.5 |
| | | | · *Niimi & Tazaki* [250] | 1 | - | Hold-out | Accuracy, Size | - | *Niimi & Tazaki* [249] |
| | | | · *Haruyama & Zhao* [266] | 3 | 1 | Hold-out | Accuracy, Size | - | *Shirasaka* et al. [263] |
| | | | · *Eggermont* et al. [270,271] | 6 | - | 10-f CV | Error | - | C4.5 |
| | | | · EPTree [233] | - | 2 | 10-f CV | Accuracy, Size | - | CART |
| | | | · EMO [232,248] | 8 | 1 | 10-f CV | Error | - | C4.5 |
| | | | · GPTree [267,268] | - | 2 | Hold-out | Accuracy, Size | - | C5.0 |
| | | | · GPDTI [239] | - | 4 | 10f-CV | Error | - | GP-based neural network [280] |
| | | | · *Kuo* et al. [257] | - | 1 | Hold-out | Accuracy | - | C5.0 |
| | | | · *To & Pham* [254] | - | 1 | Complete dataset | Sensitivity, Specificity | - | SVM, LR, LDA |
| | | | · DTiGP [229] | 18 | - | 4-f CV | Accuracy | Avg, WTL | J48, GP |
| | | | · MGP [242] | 12 | - | 10-f CV | Accuracy, Time | - | C4.5, DTiGP, GP |
| | | | · GPEI [240,272] | 5 | - | 10-f CV | Accuracy | - | C4.5, ID3-S [281] |
| | | | · *Saremi & Yaghmaee* [275] | 6 | - | Hold-out | Accuracy, Size | - | C4.5 |
| | | | · *Karakatič & Podgorelec* [273] | 6 | - | 10f-CV | Accuracy | Mann-W | GP with sub-tree switching |
| | | | · NHEMOtree [244] | - | 301 | 5-f CV | Error, MC | Friedman, Wilcoxon | CART, NHEMOtree with sub-tree switching |
| | | | · *Saremi & Yaghmaee* [276] | 6 | - | 5-f CV | Accuracy, Size, Time | - | CHAID, evtree |
| | | | · *Karakatič* et al. [274] | 10 | - | 5-f CV | Accuracy, F-Score, Time | Kruskall, Wilcoxon | GP with standard fitness evaluation |

(*continued on next page*)

**Table 23** (*continued*)

| Stra-tegy | DT | MH | Studies | Datasets | | Sampling method | Performance measures | Statistical tests | Compared methods |
|---|---|---|---|---|---|---|---|---|---|
| | | | | UCI | other | | | | |
| | | GGP | · FGP [5,234] | - | 2 | 3-f CV | Accuracy | - | C4.5 |
| | | | · CGP [227] | 3 | - | 10-f CV | MC | - | C4.5, kNN, NB, PART [282] |
| | | | · *Johansson* et al. [259,260] | 26 | - | 4-f CV | Accuracy, Fidelity | Friedman, Bonferroni, Nemenyi | J48 |
| | | | · MGP [241] | 10 | - | 5-f CV | AUC | WTL, Wilcoxon | C4.5, FGP, GGP, EGP |
| | | | · EDDIE-101 [231] | 5 | 3 | Hold-out | Accuracy | Wilcoxon | J48, REPTree, RF, UDT-EA |
| | | | · MOGP [243] | 27 | - | 5-f CV | AUC | WTL, Wilcoxon | C4.5, NB, FGP, GGP, EGP, PRIE [283], 4 MO-based EAs |
| | | | · CH-MOGP [228] | 27 | - | 5-fCV | AUC, Time | Wilcoxon | C4.5, NB, PRIE [283], 4 MO-based EAs |
| | | TGP | · *Khoshgoftaar* et al. [255,256] | - | 1 | Hold-out | Error | - | GP |
| | | | · *Zhao* [258] | 13 | - | Complete dataset | AUC | - | C4.5, MLP, SVM |
| | OB | TGP | · *Bot & Langdon* [6,261] | 4 | - | 10-f CV | Accuracy, Size | - | C5.0, OC1, M5′ [282] |
| | | | · ODT-Θ-NSGA-III [245] | 10 | - | 5-f CV | F-Score | Friedman, Iman-D, Shaffer | OC1-GA, C4.5, ten methods to create classification rules, and five ensemble methods |
| | NL | GP | · G$^3$P [236,278] | 6 | - | 10-f CV, Hold-out | Error, Size | - | C4.5, ID3, NB |
| | | | · *Šprogar* [262] | 21 | - | 5-f CV | Accuracy | Wilcoxon | GP with several crossover operators |
| | | GGP | · UDT-EA [231] | 5 | 3 | Hold-out | Accuracy | Wilcoxon | J48, REPTree, RF |
| | | TGP | · *Mugambi & Hunter* [279] | - | 1 | Hold-out | ROC | - | RBF-NN |
| | SF | GP | · Fuzzy-GP [235] | 5 | - | 10-f CV | Error | - | C4.5, and two methods to create classification rules |
| | | TGP | PFDT [246] | - | 2 | Hold-out | ROC | - | C4.5 |

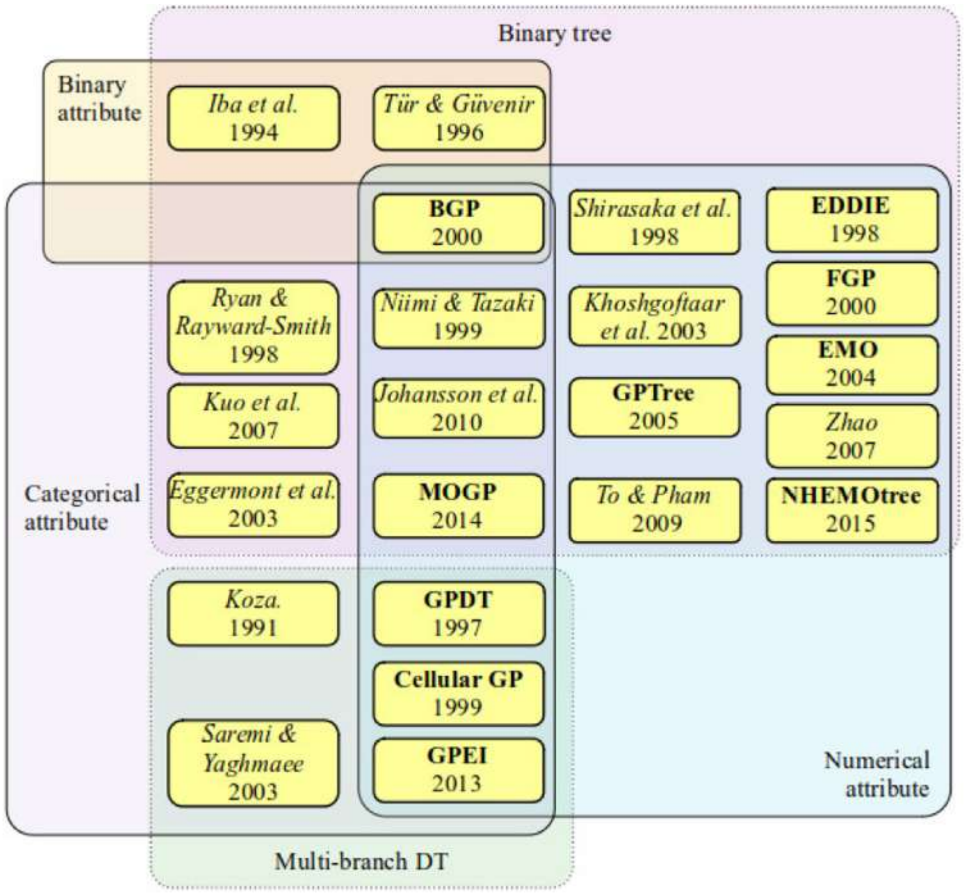

**Fig. 20.** DTs and attribute types used with GP-based MHs inducing axis-parallel-DTs.

**Table 24**
Acronyms used to identify several other EA-based MH for DTI.

| Acronym | Description |
|---|---|
| *Coevolutionary algorithms:* | |
| MPGA | Multi-population GA for DTI [284] |
| MPGT | Multi-population genTrees [285] |
| SEADT | Self-adapting Evolutionary DT [286] |
| *Differential Evolution:* | |
| AJADE-MDT | Adapted JADE with Multivariate DT [287] |
| DE-ADT$^{SPV}$ | DE-based method to build axis-parallel-DTs using SPV rule [8] |
| DE-ODT | DE algorithm to build oblique-DTs [288] |
| OC1-DE | OC1-based DE [289] |
| PA-DE | Parallel-Coordinates-based DE [290] |
| PDT | Perceptron DT [291] |
| *Evolution strategies:* | |
| MESODT | Multi-membered ES Oblique DT [292] |
| OC1-ES | OC1-based ES [11] |
| *Estimation of Distributions Algorithm:* | |
| Ardennes | Ardennes [293] |
| *Gene Expression Programming:* | |
| GEPDT | GEP decision tree [294] |
| *Grammatical Evolution:* | |
| MGEDT | MO approach to evolving DT using GE [295] |
| *Hybrid approaches:* | |
| CGP/SA | The Cellular-GP coupled with SA [296] |
| GP-MM | GP hybridized with Margin Maximisation [297] |

value rule (DE-ADT$^{SPV}$) [8]. This method evolves real-valued parameters encoding the internal nodes (attributes and threshold values) of a DT in a global search strategy. A binary-DT depth, whose number of internal nodes is not less than the number of attributes ($d$) in the training set, is used to define the vector size ($n$), as Eqn. (6).

$$n = 2^{\max\{\lceil \log_2 (d+1)+1\rceil, \lceil \log_2 (s)+1\rceil\}} - 1 + n_z, \qquad (6)$$

where $s$ is the number of class labels in the dataset, and $n_z$ is the number of threshold values associated with the numerical attributes used in the internal nodes. Also, a three-stage procedure to map one DT from this real-valued vector is defined (Fig. 23). On the other hand, *Dolotov & Zolotykh* [302] describe a similar approach. However, their coding scheme allows each attribute to be used only once as an internal node. Finally, *Mitrofanov & Semenkin* [303] describe a recursive-partitioning strategy to build an axis-parallel-DT. Here, the population evolves to find a near-optimal univariate test condition.

*Estimation of Distributions Algorithm:* The Ardennes method of *Cagnini* et al. [293] implement a global-search strategy to find axis-parallel-DTs with numerical attributes. Ardennes first constructs a Probabilistic Graphical Model (PGM) resembling a complete binary tree. Then, DTs' initial population is created using (1) PGM to place attributes and (2) information gain to compute the threshold values. Next, the evolutionary process is conducted, updating the PGM and replacing the worst individuals with new trees. Finally, the best DT in the last population is returned as the algorithm solution.

*Evolution Strategies: Dolotov & Zolotykh* [302] uses (1 + 1)-ES in a global-search strategy where one real-valued chromosome encodes the internal nodes of an axis-parallel-DT. They use the SPV rule to determine how the attributes are used to build a DT. In (1 + 1)-ES, a single offspring is created from one parent.

*Grammatical Evolution: Motsinger-Reif* et al. [304] define an appropriate grammar to map axis-parallel-DTs from binary strings used to model one gene-gene interaction [305]. The binary string is decoded into an integer string, and then grammar is used to construct a DT. Furthermore, *Ono & Kushida* [306] try controlling the search bias present on the evolutionary process by estimating the solution landscape using rank correlation. Finally, the MO approach to evolving DT using GE (MGEDT) of *Pereira* et al. [295] is used to induce binary DTs.

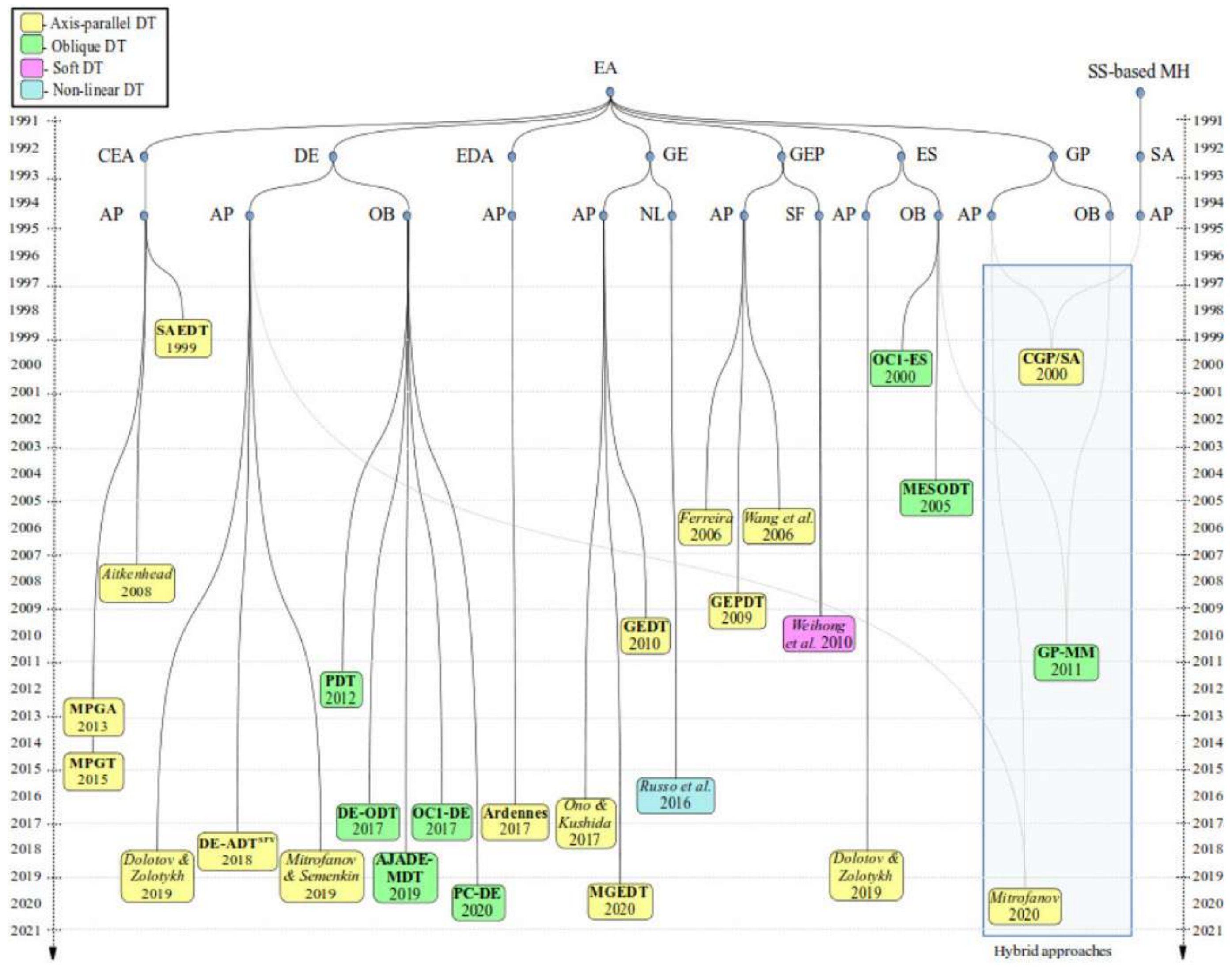

**Fig. 21.** Timeline of the other EA-based approaches for DTI.

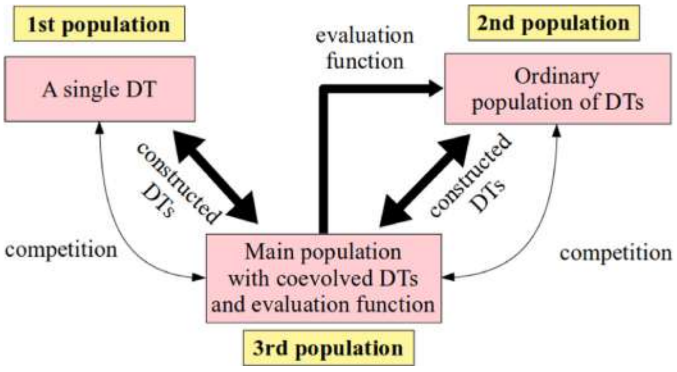

**Fig. 22.** Population competing to find a near-optimal DT (adapted to [286]).

*Gene Expression Programming: Ferreira* [307] and *Wang* et al. [308] conduct a global-search of near-optimal axis-parallel-DTs. A linear chromosome is composed of one or more genes, each one structurally divided into a head and a tail. The head encodes the dataset attributes, and the tail works as a buffer of class labels. An additional vector represents the threshold-values used with numerical attributes. Furthermore, the GEP decision tree (GEPDT) method of *Qu* et al. [294] considers the values-range of each numerical attribute to compute the threshold-values used in test conditions.

*Hybrid approaches:* Two studies combine GP with other MHs to implement a global-search strategy to find a near-optimal DT. The Cellular-GP coupled with SA (CGP/SA) of *Folino* et al. [296] uses a cellular automaton to evolve DTs placed in a grid. In each evolutionary step, the best neighbor of each DT is selected to be recombined and generate two offsprings. The best offspring replace the current DT applying the Boltzmann criterion. Furthermore, *Mitrofanov* [309] use DE to optimize the threshold-values of all internal nodes of the trees evolving in a GP-based DTI method.

### 5.4.2. Oblique-DTs

*Differential Evolution:* Both recursive-partitioning and global-search strategies to induce oblique-DTs have been implemented using DE-based approaches. In the first case, the following methods evolve a population of real-valued individuals to find near-optimal hyperplanes:

1. The OC1-DE algorithm of *Rivera-López* et al. [289].
2. The Adapted JADE with Multivariate DT (AJADE-MDT) method of *Jariyavajee* et al. [287].
3. The Parallel-Coordinates (PA-DE) algorithm of *Estivil-Castro* et al. [290].

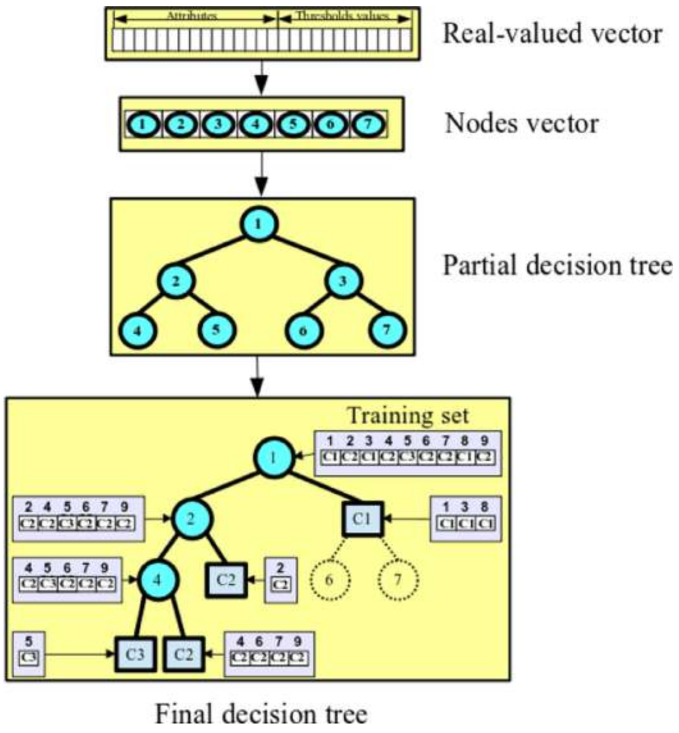

**Fig. 23.** Mapping scheme used in the DE-ADT$^{SPV}$ method.

OC1-DE and PA-DE apply the standard DE algorithm. AJADE-MDT uses a self-adaptive version adjusting the DE control-parameters into its evolutionary process.

On the other case, two methods implement a global search strategy to find a near-optimal oblique-DT:

1. The Perceptron DT (PDT) method of *Lopes* et al. [291] and *Freitas* et al. [310], where (1) the hyperplane coefficients of one DT are encoded with a real-valued individual, (2) the hyperplane independent terms, and the class label of leaf nodes, are stored in two additional vectors. In each DE iteration, mutation parameters are randomly altered. A group of new DTs randomly created replaces the worst individuals in the population.
2. The DE algorithm to build oblique-DTs (DE-ODT) of *Rivera-López & Canul-Reich* [288], where the size of the real-valued vector is computed as a factor of the number of internal nodes of an oblique-DT estimated using the number of dataset attributes.

*Evolution strategies:* Two ES-based methods for DTI use a recursive-partitioning scheme to create oblique-DTs:

1. The OC1-ES of *Cantú-Paz & Kamath* [11,137] implements a $(1+1)$-ES.
2. The Multi-membered ES Oblique DT (MESODT) induction method of *Zhang* et al. [292] applies $(\mu, \lambda)$-ES using $\mu$ parents to create $\lambda$ offsprings.

*Hybrid approaches:* The GP hybridized with Margin Maximisation (GP-MM) of *Agapitos* et al. [297] is a GGP-based method used to induce oblique-DTs. GP-MM implements a $(1+1)$-ES as a local search strategy to maximize the margins associated with each hyperplane.

### 5.4.3. Non-linear-DTs

*Grammatical Evolution: Russo* et al. [311] apply C4.5 and RTree to create DTs of the initial population of a GE-based approach to induce non-linear-DTs.

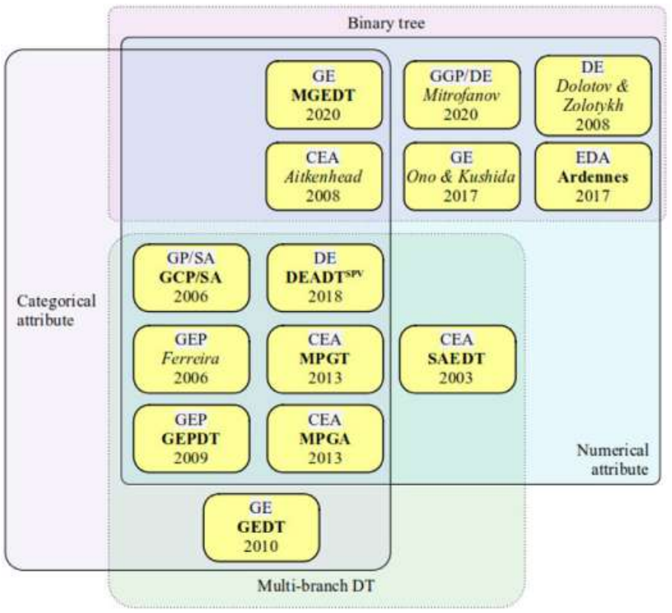

**Fig. 24.** DTs and attribute types used with other EA-based MHs inducing axis-parallel-DTs.

### 5.4.4. Soft-DTs

*Gene expression programming: Weihong* et al. [312] implement a GEP-based method including a fuzzification process of numerical attributes. Several symmetrical triangular membership functions are associated with each test condition of an axis-parallel DT previously induced with the GEPDT method.

### 5.4.5. Discussion

Fig. 24 shows the attribute and branch types of axis parallel DTs generated using these EA-based methods, and Table 25 lists the criteria used to handle categorical attributes. In particular, the discretization of numerical values is applied in SAEDT and GCP/SA.

Table 26 shows the components of these approaches. Most of these studies use a single-objective fitness function. However, four apply an aggregating fitness function, and only one uses a multi-objective fitness function in its evolutionary process. Furthermore, accuracy is the fitness measure most commonly used by these methods. Tournament-based selection is the prominent operator implemented in these studies, but linear-ranking and roulette-wheel-based selection have also been used. The sub-tree-swapping crossover and the node-disturbance mutation are applied in eight studies, although several studies implement specialized operators to evolve their candidate solutions. Finally, a random generation of candidate solutions to create the initial population is conducted with most of these approaches. In particular, MESODT creates a set of hyperplanes through an ANN-based approach. In the work of *Aitkenhead*, a randomly generated decision stump is used as the first candidate solution in his CEA-based method.

Table 27 resumes the experimental studies reported in the existing literature implementing other EA-based approaches for DTI. K-fold CV and hold-out and test accuracy and size are the sampling methods and the performance measures most used in these studies. Eight studies implement a statistical test of their experimental results. One post-hoc analysis is conducted in four methods.

Fig. 21 shows that more than half of these EA-based studies have been proposed in recent years, highlighting DE and GE-based approaches to induce DTs. In the first case, a mapping scheme to build DT from candidate solutions is needed, and in the second one, this scheme is avoided since grammar is used to build DTs. However, the number of studies using these EAs is far fewer than those using GA or GP, even

**Table 25**
Criteria used to handle categorical attributes with other EA-based methods.

| Criteria | Study |
|---|---|
| Multi-branching | MPGA [284], MPGT [285], DE-ADT$^{SPV}$ [8], GEDT [304], MGEDT [295], *Ferreira* [307], and GEPDT [294]. |
| Binary mapping | *Aitkenhead* [301]. |

though they effectively find near-optimum solutions to many complex problems.

## 6. Swarm-intelligence-based methods for DTI

Only three swarm-intelligence methods have been applied for DTI: ant colony optimization, particle swarm optimization, and bat algorithm. All approaches described in the existing literature induce axis-parallel DTs, only. Table 28 lists the acronyms used to name various SI-based MHs for DTIs, as defined by their authors. A timeline of these methods is shown in Fig. 25.

### 6.1. Ant colony optimization

*Bursa & Lhotska* [317,323] describe the ACO-DTree method as a global-search strategy in which each artificial ant construct an axis-parallel DT using a pheromone matrix. This matrix represents a fully connected graph where the nodes are associated with the attributes of the dataset, the edges indicate the transition between two nodes, and the pheromone values indicate the probability of visiting a successor node. First, a root node with one attribute randomly chosen is created, and for each possible successor node, the next attribute is probabilistically selected using the pheromone matrix. This process is repeated until a predefined DT depth is reached. In the ACO-DTree, only ants representing better solutions can deposit pheromone in the matrix. The induced DTs are pruned by applying a penalty value for unused nodes.

*Boryczka & Kozak* [12,324] describe a similar approach known as ACDT (Ant Colony algorithm for constructing DTs). ACDT uses a combination of splitting criterion and pheromone values to select the attributes used to build a DT. Also, *Boryczka & Kozak* [318] implement an adaptive discretization of numerical attributes in the continuous ACDT (cACDT) method.

Furthermore, the Ant-Tree-Miner (ATM) method of *Otero* et al. [13] implements an iterative process where each artificial ant creates a new DT based on a combination of splitting criterion and pheromone values. Each element in the pheromone matrix has three values $(edge, level, x)$, where $edge$ represents a univariate test condition, $level$ is the DT level of the edge, and $x$ is a successor attribute.

### 6.2. Particle swarm optimization

*Veenhuis* et al. [321] introduce the Tree Swarm Optimization (TSO) method to induce axis-parallel DTs. A DT is a particle moving in the solution space represented as a sequence of nodes (test conditions and leaf nodes). Each node has a *vector of symbols* grouping all possible test conditions and class labels (Fig. 26). The symbol used in a node is the one with the maximum value in the real-valued vector representing the swarm particle. If a numerical attribute is selected, the value of the vectors first element is taken as its threshold value.

Furthermore, two multi-objective PSO-based methods have been implemented:

1. The DT-MPSO method of *Santos & Naval* [319] encodes the nodes of a complete axis-parallel DT with a predefined depth as a particle. This depth is increased in the search procedure if no significant improvement exists in the best solution.
2. *Fieldsend* [325] uses a real-valued matrix, where each row is a vector of symbols (similar to TSO), and each column represents a tree node. An additional matrix column is used to store the threshold values of numerical attributes.

Alternatively, *Chan* et al. [326] perform a recursive-partitioning strategy to find the test conditions with numerical attributes used in an axis-parallel DT. Each particle represents one univariate test condition. Additionally, *Cho* et al. [327] implement a subsequent-optimization strategy to improve the threshold values of the test conditions of a DT previously induced by CART. Each particle in the swarm encodes the threshold values of all test conditions used in the induced DT. Finally, *Malik & Khan* [320] uses PSO to prune binary DTs previously induced. Both the single-objective optimized DT pruning (SO-DTP) and MO optimized DT pruning (MO-DTP) approaches are implemented. In these methods, a particle represents the internal nodes to the tree to be pruned.

### 6.3. Bat algorithm

In the Bat-Tree-Constructor (BTC) of *Bida & Aouat* [322], a bat population explores the DT space to find a near-optimal DT. DTs can use both nominal and numerical attributes.

### 6.4. Discussion

Fig. 27 shows the attribute and branch types of axis parallel DTs generated using these SI-based methods, and Table 29 lists the criteria used to handle categorical attributes. In particular, the discretization of numerical values is applied in ACDT and ATM. Fig. 28 shows the scheme used for the methods implementing a subsequent-optimization strategy.

In Table 30 are shown the components of the SI-based methods for DTI. Five studies use one single-objective fitness function and four other evaluate an aggregating fitness function. Only three methods apply a multi-objective fitness function. Test accuracy and error-rate have been used as fitness measures in three and four studies, respectively. Furthermore, matrix representation and sequences of values have been used to encode candidate solutions in three and five studies, respectively. Finally, the random generation of candidate solutions to create the initial population is the strategy applied in all studies implementing these MHs.

Table 31 resumes the experimental studies reported in the existing literature implementing SI-based methods for DTI. Six studies implement a hold-out sampling procedure. One k-fold CV is applied in ATM, APSO, and BTC, as well as in the procedure described by *Chan* et al. TSO and MOPSO use the complete datasets to obtain their performance values. Test accuracy, size, and error-rate have been adopted as performance measures in nine, eight, and three studies, respectively. Finally, two methods report the use of statistical tests to compare its results.

Unlike other data mining techniques such as attribute selection [328,329], or for classifier parameter tuning [330] where swarm-based methods are extensively applied, the use of SI-based approaches to induce DT has been poorly studied. Since the most prominent SI-based methods have been created to solve numerical-optimization problems, their use to disturb tree-like structures involves the definition of (1) a mapping scheme between particles and DTs, and (2) a correspondence between swarm operators and DTI elements.

Ant colony optimization can build DTs since it denotes a problem with a graph where artificial ants walk to find a near-optimal solution. This scheme allows expressing a DT using a pheromone matrix. In particular, a challenge for the use of particle swarm optimization and the bat algorithm is the definition of a scheme to map the DT structure from

**Table 26**
Components of other EA-based approaches for DTI.

| Strategy | DT | FF | MH | Studies | Year | Rep. scheme | Fitness measure | Genetic operators | | | Initial population |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | Selection | Crossover | Mutation | |
| RP | AP | UF | DE | · *Mitrofanov & Semenkin* [303] | 2019 | Linear | ClusterS | Tournament | Binomial | DE mutation | Numerical chromosomes randomly created |
| | OB | UF | DE | · OC1-DE [289] | 2017 | Linear | Twoing | Tournament | Binomial | DE mutation | Hyperplanes randomly created and several copies of the best axis-parallel hyperplane found by OC1-AP |
| | | | | · AJADE-MDT [287] | 2019 | Linear | IG | Tournament | Binomial | DE mutation | Hyperplanes randomly created |
| | | | | · PA-DE [290] | 2020 | Linear | GR | Tournament | Binomial | DE mutation | Hyperplanes randomly created |
| | | | ES | · OC1-ES [11,137] | 2000 | Linear | Twoing | N/A | N/A | Special | Axis-parallel hyperplanes found by OC1-AP |
| | | | | · MESODT [292] | 2005 | Linear | IG, LinearS | Special | Special | Special | Hyperplanes created by ANN |
| GS | AP | UF | CEA | · *Aitkenhead* [301] | 2008 | Tree | Accuracy | N/A | N/A | Disturb node | A decision stump randomly created |
| | | | DE | · DE-ADT$^{SPV}$ [8] | 2018 | Linear | Accuracy | Tournament | Binomial | DE mutation | Numerical chromosomes randomly created |
| | | | | · *Dolotov & Zolotykh* [302] | 2019 | Linear | Accuracy | Tournament | Binomial | DE mutation | Numerical chromosomes randomly created |
| | | | EDA | · Ardennes [293] | 2017 | Tree | Accuracy | TruS | N/A | N/A | DT randomly created |
| | | | ES | · *Dolotov & Zolotykh* [302] | 2019 | Linear | Accuracy | N/A | N/A | Special | Numerical chromosomes randomly created |
| | | | GE | · *Ono & Kushida* [306] | 2017 | Linear | Accuracy | Tournament | Single-point | Flip-bit | DTs randomly created |
| | | | GEP | · *Ferreira* [307], *Wang* et al. [308] | 2006 | Linear | Accuracy | Roulette | Single-point, Double-point, Special | Uniform, Special | String chromosomes randomly created |
| | | | | · GEPDT [294] | 2009 | Linear | Accuracy | Roulette | Single-point, Double-point, Special | Uniform, Special | String chromosomes randomly created |
| | | | GGP/SA | · GCP/SA [296] | 2000 | Tree | Error | Special | Swap sub-trees | Replace sub-tree | DTs randomly created |
| | | | GGP/DE | · *Mitrofanov* [309] | 2020 | Tree | Accuracy | Tournament | Swap sub-trees | Replace sub-tree | DTs randomly created |
| | | AF | CEA | · SAEDT [286,298–300] | 1999 | Tree | Accuracy+Size | Linear ranking | Swap sub-trees | Disturb node, Replace sub-tree | DTs randomly created |
| | | | | · MPGA [284] | 2013 | Tree | Accuracy+Size | Linear ranking | Swap sub-trees | Disturb node, Replace sub-tree | DTs randomly created |
| | | | | · MPGT [285] | 2015 | Tree | Accuracy+F-Score | Tournament | Swap sub-trees | Disturb node, Replace sub-tree | DTs randomly created |
| | | | GE | · GEDT [304] | 2010 | Linear | Sensitivity, Specificity | Tournament | Single-point | Flip-bit | DTs randomly created |
| | | MF | GE | · MGEDT [295] | 2020 | Linear | AUC, Size | Tournament | Swap sub-trees | Replace sub-tree | DTs randomly created |
| | OB | UF | DE | · PDT [291,310] | 2012 | Linear | Error | Tournament | Binomial | DE mutation | Numerical chromosomes randomly created |
| | | | | · DE-ODT [288] | 2017 | Linear | Accuracy | Tournament | Binomial | DE mutation | Numerical chromosomes randomly created |
| | | | GGP/ES | · GP-MM [297] | 2011 | Tree | Accuracy | Tournament | N/A | Sub-tree replacement | RH&H criterion |
| | NL | UF | GE | · *Russo* et al. [311] | 2016 | Linear | Error | Tournament | Single-point | Flip-bit | DTs created by C4.5 |
| | SF | UF | GEP | · *Weihong* et al. [312] | 2010 | Linear | Accuracy | Roulette | Single-point, Double-point, Special | Uniform, Special | String chromosomes randomly created |

**Table 27**
Experimental analysis reported using other EA-based approaches for DTI.

| Stra-tegy | DT | MH | Studies | Datasets UCI | Datasets other | Sampling method | Performance measures | Statistical tests | Compared methods |
|---|---|---|---|---|---|---|---|---|---|
| RP | AP | DE | *Mitrofanov & Semenkin* [303] | 4 | - | 10f-CV | Accuracy, Time | *t*-test | ID3, CART |
| | OB | DE | OC1-DE [289] | 16 | - | 5-fCV, 10f-CV | Accuracy, Size | Friedman, Nemenyi | OC1, OC1-SA, OC1-GA, OC1-ES, HBDT |
| | | | AJADE-MDT [287] | 5 | - | 5-fCV | Accuracy, Size | - | OC1, HHCART [313] |
| | | | PA-DE [290] | 15 | - | 10-fCV | F-Score, Size | Friedman, Bergmann | C4.5, J48, OC1, OC1-DE |
| | | ES | OC1-ES [11,137] | 10 | 3 | 5-f CV | Accuracy, Size, Time | *t*-test | CART, OC1, OC1-GA, OC1-SA |
| | | | MESODT [292] | 2 | 2 | Hold-out | Accuracy, Size | - | C5.0, OC1, OC1-ES, Alopex Perceptron DT [35] |
| GS | AP | CEA | SAEDT [286,298] | - | 1 | Hold-out | Accuracy, Sensitivity, Specificity, Size | - | C4.5, genTrees |
| | | | SAEDT [299] | - | 1 | Hold-out | Accuracy, Sensitivity, Specificity, Size | - | MtDeciT [314], C5.0 |
| | | | SAEDT [300] | - | 1 | Hold-out | Accuracy, Sensitivity, Specificity, Size | - | MtDeciT, Neuro generated DT [300], C5.0 |
| | | | *Aitkenhead* [301] | 2 | - | Complete dataset | Accuracy | - | C4.5, GALE, MLP |
| | | | MPGA [284] | - | 1 | 5-f CV | Accuracy | - | genTrees, J48, NB, SVM, MLP, AdaBoost [315] |
| | | | MPGT [285] | 10 | - | 5-f CV | Accuracy, F-Score, Size | Friedman, Nemenyi | C4.5, CART, genTrees |
| | | DE | DE-ADT$^{SPV}$ [8] | 20 | - | 10f-CV | Accuracy, Size | Friedman, Bergmann | J48, CART, NB, MLP, RBF-NN, RF |
| | | | *Dolotov & Zolotykh* [302] | 20 | - | ? | Accuracy | - | CART, MLP, ES |
| | | EDA | Ardennes [293] | 10 | - | 10f-CV | Accuracy, Size | Avg-R | J48, LEGAL-Tree |
| | | ES | *Dolotov & Zolotykh* [302] | 20 | - | ? | Accuracy | - | CART, MLP, DE |
| | | GEP | *Ferreira* [307] | 2 | - | Hold-out | Accuracy | - | - |
| | | | *Wang* et al. [308] | 2 | - | Complete dataset | Accuracy | - | - |
| | | | GEPDT [294] | 8 | - | 5-f CV | Accuracy | - | C4.5 |
| | | GE | GEDT [304] | - | 40 | 10-f CV | Error | - | C4.5 |
| | | | *Ono & Kushida* [306] | 5 | - | Complete dataset | Accuracy, F-Score, Size | - | C4.5 |
| | | | MGEDT [295] | - | 1 | Hold-out | AUC, Size | - | CART, RF, MLP |
| | | GP/SA | GCP/SA [296] | 12 | - | Hold-out | Error, Size | - | C4.5 |
| | | GGP/DE | *Mitrofanov* [309] | 8 | - | 10-f CV | Accuracy | - | C4.5 |
| | OB | DE | PDT [291,310] | 8 | - | Hold-out | Accuracy | Kruskall | J48, Best-first DT [316], RTree, RBF-NN, MLP, NB, kNN |
| | | | DE-ODT [288] | 13 | - | 10f-CV, Hold-out | Accuracy, Size | Friedman, Nemenyi | J48, OC1, PDT, EFTI |
| | | GGP/ES | GP-MM [297] | 5 | - | 10-f CV | Accuracy | - | C4.5, GP, SVM, NB |
| | NL | GE | *Russo* et al. [311] | - | 1 | Hold-out | Error | - | J48, RTree |
| | SF | GEP | *Weihong* et al. [312] | 2 | - | 5-f CV | Accuracy | - | GEPDT |

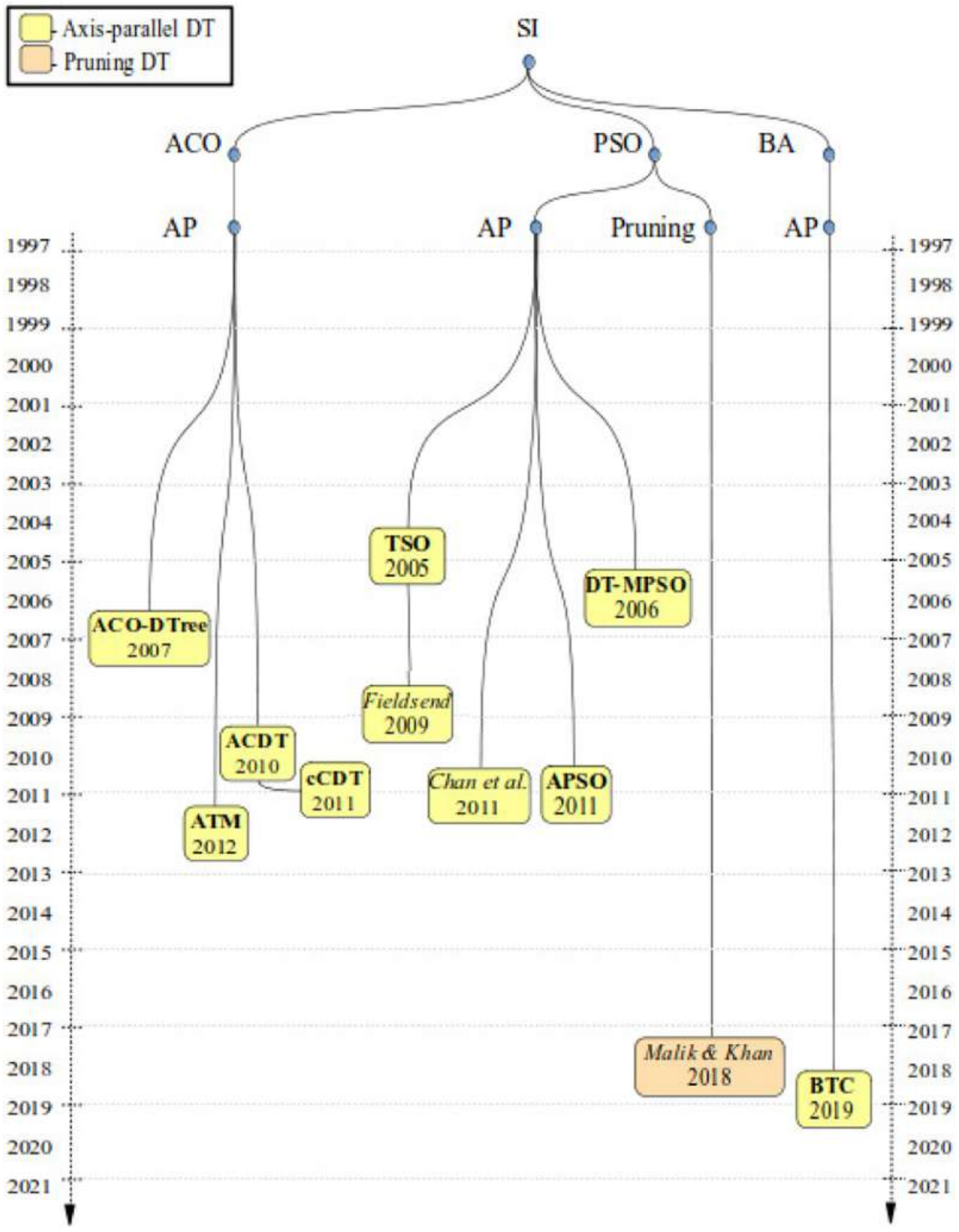

**Fig. 25.** Timeline of swarm-intelligence-based methods for DTI.

**Table 28**
Acronyms used to identify several SI-based MH for DTI.

| Acronym | Description |
|---|---|
| *Ant colony optimization:* | |
| ACDT | Ant Colony algorithm for constructing DTs [12] |
| ACO-DTree | ACO-based method for DT [317] |
| ATM | Ant-Tree-Miner method [13] |
| cACDT | Continuous ACDT method [318] |
| *Particle swarm optimization:* | |
| DT-MPSO | DT-based using multi-objective PSO [319] |
| MO-DTP | Multi-objective optimized DT pruning [320] |
| SO-DTP | Single-objective optimized DT pruning [320] |
| TSO | Tree Swarm Optimization method [321] |
| *Bat algorithm:* | |
| BTC | Bat-Tree-Constructor [322] |

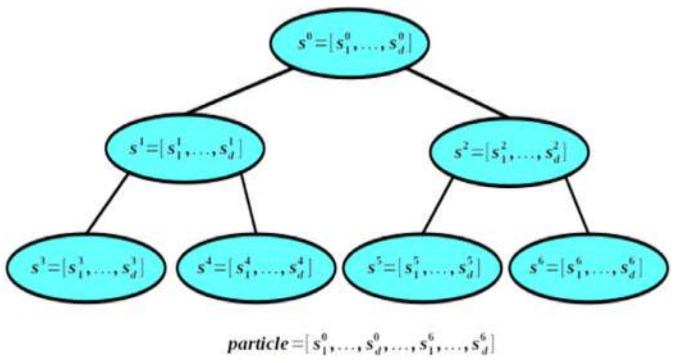
$s^0=[s^0_1,\dots,s^0_d]$
$s^1=[s^1_1,\dots,s^1_d]$
$s^2=[s^2_1,\dots,s^2_d]$
$s^3=[s^3_1,\dots,s^3_d]$
$s^4=[s^4_1,\dots,s^4_d]$
$s^5=[s^5_1,\dots,s^5_d]$
$s^6=[s^6_1,\dots,s^6_d]$
$particle=[s^0_1,\dots,s^0_d,\dots,s^6_1,\dots,s^6_d]$

**Fig. 26.** DTs representation using a TSO particle (Adapted of [321]).

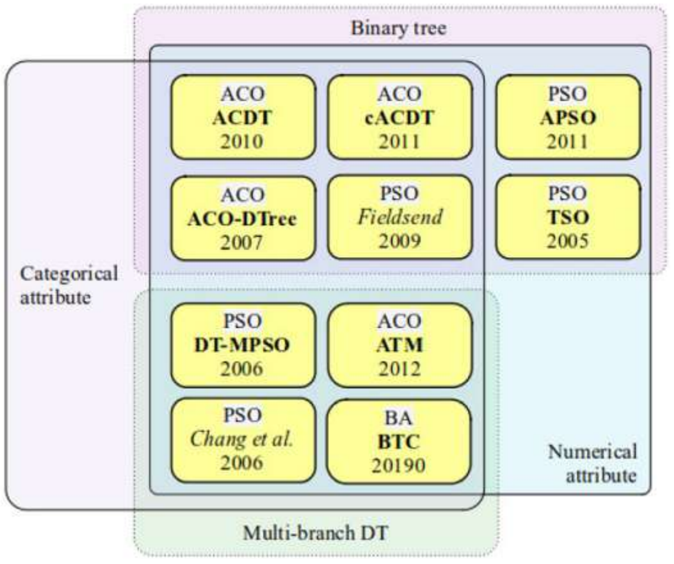

**Fig. 27.** DTs and attribute types used with other SI-based MHs inducing axis-parallel-DTs.

**Table 29**
Criteria used to handle categorical attributes with SI-based methods.

| Criteria | Study |
|---|---|
| Multi-branching | ATM [13], DT-MPSO [319], BTC [322] |
| Binary mapping | ACDT [12], cACDT [318] |
| Numerical mapping | ACO-DTree [317], Fieldsend [325] |

a real-valued vector representation. The lack of SI-based approaches for DTI can be since they are more recently introduced in the existing literature than EAs.

## 7. Hyper-heuristic-based approaches to build DTI methods

Hyper-heuristic-based approaches used to create DTI methods described in the existing literature are based on genetic algorithms or grammatical evolution. Table 32 lists the acronyms used to name various HH-based aproaches for DTIs, as defined by their authors. The timeline of these methods is shown in Fig. 29.

### 7.1. Genetic algorithms with linear chromosomes

*Vella* et al. [336] describe a GA-based hyper-heuristic to build DTI algorithms (HHDT). An individual represents a list of rules to select the most appropriate splitting criterion based on the entropy degree of

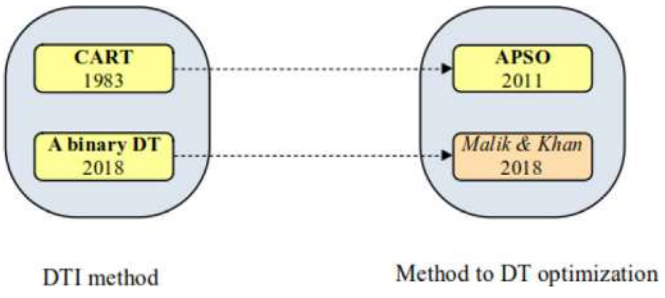

**Fig. 28.** Scheme used to subsequent-optimization with SI-based MHs.

dataset attributes. Each rule "*if* $(x > high$ *and* $y < low)$ *then apply criterion h*" is codified as a 5-tuple $(x, high, y, low, h)$, where $x$ and $y$ are entropy values, $high$ and $low$ are threshold values, and $h$ identifies one of 5 splitting criteria.

*Barros* et al. [334,340] describe the hyper-heuristic EA for designing DT algorithms (HEAD-DT). This method encodes four components of a DTI method in a linear chromosome: split criterion, stopping rule, procedure to deal with missing values, and pruning method. Moreover, *Basgalupp* et al. [337] introduce a multi-objective version of HEAD-DT (MOHEAD-DT) allowing to choose between the Pareto dominance approach and the lexicographic analysis. MOHEAD-DT proposes several modifications in fitness calculation, selection operator, and the procedure to select the best method in the population. Furthermore, *Jovanović* et al. [341] implement a similar approach encoding five components: split criterion, split evaluation method, stop criterion, pruning approach, and the feature subset selection procedure. These components are obtained from several DTI methods such as ID3, C4.5, CART, and CHAID.

Recently, *Nyath & Pillay* introduce AutoGA [333], a GA-based method to search for a GP configuration to lead optimized classifiers. AutoGA encodes 14 GP parameters: representation, population size, tree generation, initial tree depth, max offspring depth, selection method, selection size, reproduction rates, mutation type, max mutation depth, reproduction sequence, operator pool, fitness type, and number of generations. Finally, *Kumar* et al. implement the Hyper-heuristic Evolutionary Approach with Recursive and Partition Trees (HEARpart). This method uses a GA to evolve a population of chromosomes encoding four parameters of the Recursive and Partition Trees (RPART) package [342]: minimum split, complexity, DT maximum depth, and node-splitting criterion.

### 7.2. Grammatical evolution

*Basgalupp* et al. [339] implement a GE-based method known as Evolutionary Split Criteria with Grammatical Evolution (ESC-GE). This method applies a grammar that automatically generates the best split

**Table 30**
Components of SI-based approaches for DTI.

| Strategy | DT | FF | MH | | Studies | Year | Repr. scheme | Fitness measure | Initial solution |
|---|---|---|---|---|---|---|---|---|---|
| RP | AP | AF | PSO | · | *Chan* et al. [326] | 2011 | Linear | GR+Accuracy | Particles randomly created |
| GS | AP | UF | ACO | · | ACO-DTree [317,323] | 2007 | Matrix | Error | A pheromone matrix randomly created |
| | | | PSO | · | TSO [321] | 2005 | Linear | Error | Particles randomly created |
| | | | BA | · | BTC [322] | 2019 | Tree | Accuracy | DT randomly created |
| | | AF | ACO | · | ACDT [12,324] | 2010 | Tree | Twoing+Pheromone | A matrix with values based on the number of attributes |
| | | | | · | cACDT [318] | 2011 | Tree | Twoing+Pheromone | A matrix with values based on the number of attributes |
| | | | | · | ATM [13] | 2012 | Matrix | GR+Pheromone | A pheromone matrix randomly created |
| | | MF | PSO | · | DT-MPSO [319] | 2006 | Linear | Error, Size | Particles randomly created |
| | | | | · | *Fieldsend* [325] | 2009 | Matrix | Error, Size | Particles randomly created |
| SO | AP | UF | PSO | · | APSO [327] | 2011 | Linear | Accuracy | Particles randomly created (DT from CART) |
| | | UF,MF | PSO | · | *Malik & Khan* [320] | 2018 | Linear | Sensitivity, Fall-out | Particles randomly created (A binary DT) |

**Table 31**
Experimental analysis reported using SI-based approaches for DTI.

| Stra-tegy | DT | MH | Studies | Datasets UCI | Datasets other | Sampling method | Performance measures | Statistical tests | Compared methods |
|---|---|---|---|---|---|---|---|---|---|
| RP | AP | PSO | · *Chan* et al. [326] | 4 | 1 | 10-f CV | Accuracy, Size | - | C5.0, CART, QUEST, CHAID |
| GS | AP | ACO | · ACO-DTree [317,323] | 1 | 2 | Hold-out | Error | - | WEKA classifiers [52] |
| | | | · ATM [13] | 22 | - | 10-f CV | Accuracy, Size | Friedman, Hommel, Wilcoxon | C4.5, CART, aCDT |
| | | | · ACDT [12] | 7 | - | Hold-out | Accuracy, Size, Time | - | CART, Ant-Miner [331], cAnt-Miner [332] |
| | | | · ACDT [324] | 30 | - | Hold-out, 10f-CV | Accuracy, Size | Friedman | RTree, CART, ATM, C4.5 |
| | | | · cACDT [318] | 11 | - | Hold-out | Accuracy, Size | - | ACDT, Ant-Miner [331], cAnt-Miner [332] |
| | | PSO | · TSO [321] | 1 | - | Complete dataset | Accuracy | - | C4.5, GP |
| | | | · DT-MPSO [319] | 9 | - | Hold-out | Error, Size | - | C4.5 |
| | | | · *Fieldsend* [325] | 5 | - | Complete dataset | Error | - | C4.5, GP, TSO |
| | | BA | · BTC [322] | 29 | - | 10-f CV | Accuracy, Size | - | ATM, C4.5, CART, RTree |
| SO | AP | PSO | · APSO [327] | 1 | 1 | 5-f CV | Accuracy | - | CART |
| | | | · *Malik & Khan* [320] | - | 1 | Hold-out | Sensitivity, Fall-out, Accuracy, Precision, Size, Time, MC | - | CART, J48, REPTree, and four NB-based variants |

**Table 32**
Acronyms used to identify several HH-based methods for DTI.

| Acronym | Description |
|---|---|
| *Genetic algorithms:* | |
| AutoGA | GA-based method to search a GP configuration [333] |
| HEAD-DT | Hyper-heuristic EA for designing DT algorithms [334] |
| HEARpart | Hyper-heuristic Evolutionary Approach with Recursive and Partition Trees [335] |
| HHDT | Hyper-heuristic GA for DTs [336] |
| MOHEAD-DT | Multi-objective HEAD-DT [337] |
| *Grammatical evolution:* | |
| AutoGE | GE-based method to search a GP configuration [338] |
| ESC-GE | Evolutionary Split Criteria with Grammatical Evolution [339] |

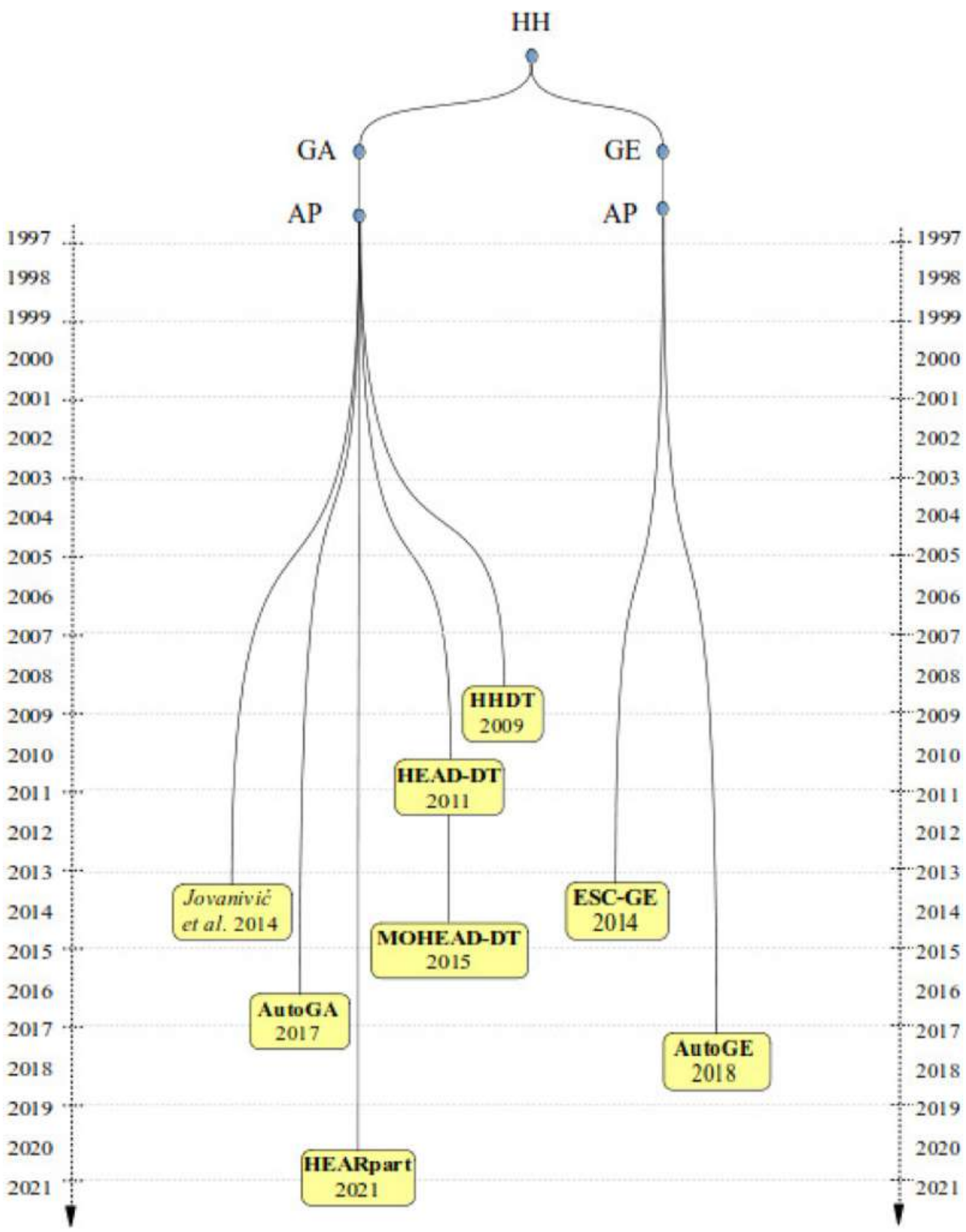

**Fig. 29.** Timeline of HH-based approaches to build DTI methods.

criterion for a DTI method. Each chromosome in the population represents a split criterion that is incorporated into a DTI algorithm. Furthermore, *Nyath & Pillay* introduce AutoGE [338], a version of AutoGA that uses GE to find the near-optimal GP parameters to induce DTI methods.

### 7.3. Discussion

Except for the method of *Jovanovic* et al. that groups the categorical values in two or more sets, the HH-based methods use the multi-branching criterion to manage these attributes. In particular, in the HHDT method, the values of numeric attributes are discretized to be handled as categorical attributes.

In Table 33 are shown the components of the HH-based approaches to building DTI methods. Both HHDT and HEAD-DT methods, and the work of *Jovanovic* et al. evaluate an single-objective fitness functions, the ESC-GE procedure implements an aggregating fitness function, and the MOHEAD-DT method applies a multi-objective fitness function. Test accuracy is the fitness measure most commonly evaluated by these approaches. On the other hand, both the sequence of values as representation scheme and the random generation of candidate solutions for the initial population are the components included in these procedures. Finally, tournament-based selection, single-point crossover, and uniform mutation are the genetic operators commonly used by these methods.

Finally, Table 34 resumes the experimental studies reported in the existing literature in which a hyper-heuristic is implemented to build DTI methods. In this Table is shown that seven methods use UCI datasets, and the ESC-GE method uses another type of dataset. Most of them implement the k-fold CV, and they calculate the test accuracy of DTI method as a performance measure. F-score and size are also evaluated each one in four studies. Six studies implement a statistical test to compare their results with those reported by other approaches.

Hyper-heuristics are novel approaches where classification methods are created to induce DTs. Although these schemes use some MHs, they select the best combination of elements to build a DT. Hyper-heuristics have been applied in two schemes: 1) to build DTI methods combining several components, and 2) to select the best splitting criterion used by one DTI method. However, the model creation process uses a recursive-partitioning strategy. They can have better performance than those models induced using traditional schemes. However, they are created using a greedy scheme, and the problems related to this induction scheme remain.

## 8. Summary analysis

In the following paragraphs, a summary analysis of the different MH-based approaches for DTI is developed. First, a general classification of the methods described in this paper is presented. Then, a summary analysis of their components and the elements considered in the experimental studies is completed. A histogram based on the published year of these methods is depicted in Fig. 30.

### 8.1. General classification

*SS-based methods:* Their classification is detailed in Table 35. These MHs have been implemented only as recursive-partitioning and subsequent-optimization strategies (Fig. 31(a)). They have been used to build oblique and axis-parallel DTs in similar proportion (Fig. 31(b)), and only five types of these MHs have been used for DTI (Fig. 31(c)), highlighting the implementation of those based on SA (50%).

*EA-based procedures:* Table 36 shows their classification. Global search (78.86%) is the strategy most commonly used to build DTs with these methods (Fig. 32(a)). In particular, EAs implementing a recursive-partitioning strategy are currently used to induce multivariate DTs. Fig. 32(b) shows that they are mainly used to generate either axis-parallel (66.67%) or oblique DTs (17.89%). Also, non-linear and soft DTs are induced to a lesser extent. Table 36 indicates that LGA and GP-based

**Table 33**
Components of HH-based approaches to build DTI methods.

| Stra- | DT | FF | MH | | Studies | Year | Rep. | Fitness | Genetic operators | | | Initial population |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| tegy | | | | | | | scheme | measure | Selection | Crossover | Mutation | |
| GS | AP | UF | LGA | · | HHDT [336] | 2009 | Linear | Accuracy | Tournament | Single-point | Uniform | Numerical chromosomes randomly created |
| | | | | · | HEAD-DT [334,340] | 2011 | Linear | Accuracy | Tournament | Single-point | Uniform | Numerical chromosomes randomly created |
| | | | | · | *Jovanović* et al. [341] | 2014 | Linear | Accuracy | Roullete | Uniform | Special | String chromosomes randomly created |
| | | | | · | AutoGA [333] | 2017 | Linear | Accuracy | Linear ranking | Uniform | Bit-string | Numerical chromosomes randomly created |
| | | | | · | HEARpart [335] | 2021 | Linear | Accuracy | Tournament | Single-point | Uniform | Numerical chromosomes randomly created |
| | | | GE | · | AutoGE [338] | 2018 | Linear | Accuracy | Tournament | Single-point | Uniform | Numerical chromosomes randomly created |
| | | AF | GE | · | ESC-GE [339] | 2014 | Linear | Precision+Sensitivity | Tournament | Single-point | Uniform | Binary chromosomes randomly created |
| | | MF | LGA | · | MOHEAD-DT [337] | 2015 | Linear | F-Score, Size | Tournament | Single-point | Uniform | Integer chromosomes randomly created |

**Table 34**
Experimental evaluation reported by hyper-heuristics.

| Stra-tegy | DT | MH | Studies | Datasets UCI | Datasets other | Sampling method | Performance measures | Statistical tests | Compared methods |
|---|---|---|---|---|---|---|---|---|---|
| GS | AP | LGA | · HHDT [336] | 12 | - | 9-f CV | Accuracy | - | ID3 |
| | | | · HEAD-DT [334,340] | 20 | - | 10-f CV | Accuracy, F-Score, Size | Friedman, Nemenyi | C4.5, CART |
| | | | · *Jovanović* et al. [341] | 16 | - | 10-f CV | Accuracy | - | C4.5, REPTree, Alternating DT [343], LR-based DTI method [344] |
| | | | · MOHEAD-DT [337] | 20 | - | 10-f CV | F-Score, Size | Friedman, Nemenyi | C4.5, CART, HEAD-DT |
| | | | · AutoGA [333] | 11 | - | Hold-out | Accuracy | Friedman, Nemenyi | GP, ANN, SVM |
| | | | · HEARpart [335] | 30 | - | Hold-out | F-Score, Error, Size | Friedman, Nemenyi | J48, CART, ATM |
| | | GE | · ESC-GE [339] | - | 20 | 5-f CV | Accuracy, F-Score, Size | Friedman, Nemenyi | J48 |
| | | | · AutoGE [338] | 22 | 6 | Hold-out | Accuracy,Time | Friedman, Bonferroni | AutoGA, GP-based approaches |

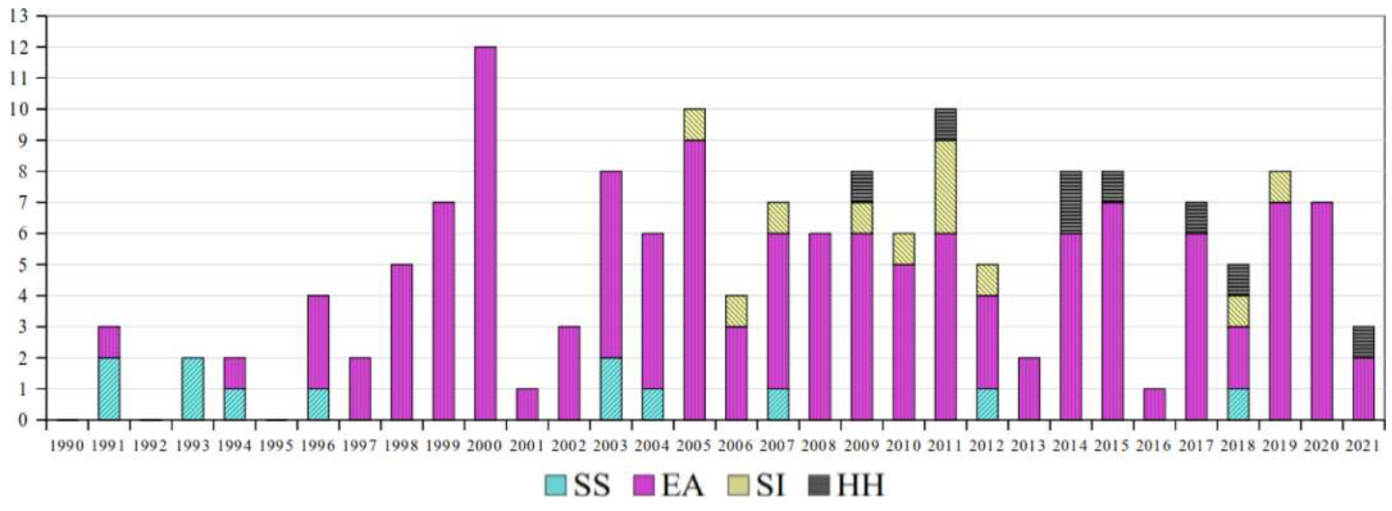

**Fig. 30.** Number of papers published from 1991 to 2021.

**Table 35**
SS-based MHs for DTI.

| Strategy | DT | MH | Num studies | Studies |
|---|---|---|---|---|
| RP | AP | GRASP | 1 | *Pacheco* et al. [142] |
| | OB | SLS | 1 | OC1 [134,135] |
| | | SA | 2 | SADT [37]; OC1-SA [137] |
| | | TS | 2 | LDTS [139]; LDSDT$_{TS}$ [36] |
| SO | AP | SA | 3 | *Sutton* [141]; *Bucy & Diesposti* [101,140]; SACS [136] |
| | | VNS | 1 | *Vila Boas* et al. [143] |
| | OB | TS | 1 | EPTS [138] |
| | SF | SA | 1 | *Dvořák & Savický* [145] |

methods construct non-linear DTs, too. Fig. 32(c) shows that GP is the MH most used to induce DTs (43.09%). Nevertheless, if LGA and TGA are combined, they represent 38.21% of these approaches. In contrast to these MHs, other EAs have been less applied for constructing DTs. They represent 18.7% of all these methods, in comparison with 81.3% of studies using GA or GP for DTI.

*SI-based methods:* The only DT type induced with SI methods is the axis-parallel DT (Table 37). Fig. 33(a) shows that a global search is typically conducted in these methods (72.73%), and Fig. 33(b) illustrates that PSO is the most used method to build DTs.

*Hyper-heuristics:* Table 38 shows that LGA is the MH most widely used to implement DTI algorithms. Axis-parallel DTs are the only DT type induced for the classifiers created with these methods.

*Summary:* Tables 39 and 40 show the classification of MHs-based approaches for DTI by type of strategy implemented and by the sort of induced DT, respectively. Fig. 34(a) shows that the global search is the prominent strategy implemented for MHs (68.18%), and in Fig. 34(b) is shown that axis-parallel DTs are the most common type of induced DT (58.44%). In Fig. 34(c) is observed that GA (34.41%) and GP (34.42%) are the most common types of MHs implemented for DTI.

### 8.2. Algorithm components

*Fitness functions:* Single-objective fitness function is the prominent fitness function evaluated in these studies (58.44%). Also, the aggregating fitness function is implemented in 45 studies (29.23%), and only 19 methods (12.54%) evaluate a multi-objective fitness function (Fig. 35 and Table 41).

*Representation schemes:* Fig. 36 and Table 42 show that the candidate solutions are represented as tree-like structures in 84 studies (54.55%), sequence of values in 67 studies (43.51%), and matrices in only three studies (1.95%).

*Fitness measures:* Fig. 37 and Table 43 show that accuracy (31.03%) and size (20.26%) are the fitness values more used in these EA-based methods. In the other MH types, the error-rate is more used (1.72% in both classes). Alternatively, information gain, twoing rule, and the Gini index are the most used splitting criteria to conduct the search procedure.

*Genetic operators:* The kinds of selection, crossover, and mutation operators used for EA-based approaches for DTI are shown in Tables 44, 45, and 46, respectively. Tournament and roulette-wheel-based selection operators are used in 67 and 23 studies, respectively. Single-point crossover in 18 studies and uniform mutation in 15 studies are the most common operators implemented when the chromosomes are sequences of values. Alternatively, sub-tree swapping (65 studies) and sub-tree replacement (40 studies) are most used with tree-like structures. Fig. 38 shows the distribution of the genetic operators implemented in these approaches.

### 8.3. Experimental studies

*Sampling methods:* Fig. 39 shows that hold-out (37.58%) and cross-validation (50.30%) are the more popular sampling methods. Table 47 shows that eleven methods use the complete dataset in their experimental analysis.

*Performance measures:* Fig. 40 shows that accuracy (40.28%) and size (28.82%) are the preferred measures to determine the quality of the induced DTs. Error-rate and the running time have also been used in these experimental studies (Table 48).

*Statistical tests:* Friedman test (20.93%), *t*-test (16.27%), Wilcoxon test (12.79%), and Nemenyi post-hoc test (11.63%) are the most adopted statistical tests in the reported studies (Fig. 41). Table 49 shows the complete distribution of each test used in these approaches.

**Table 36**
EA-based approaches for DTI.

| Strategy | DT | MH | Num studies | Studies |
|---|---|---|---|---|
| RP | AP | TGA | 1 | *Rzheutskaya* et al. [216] |
| | | DE | 1 | *Mitrofanov & Semenkin* [303] |
| | OB | LGA | 6 | BTGA [154]; OC1-GA [11,137]; *Krętowski* [33]; RCGA-kDT [173]; *Pangilinan & Janssens* [180]; HBDT [169] |
| | | DE | 3 | OC1-DE [289]; AJADE-MDT [287]; PA-DE [290] |
| | | ES | 2 | OC1-ES [11,137]; MESODT [292] |
| | NL | LGA | 2 | GA-QDT [161,183]; NLDT [172] |
| | | GP | 2 | *Marmelstein & Lamont* [237]; GIODeT [277] |
| | SF | LGA | 4 | *Janikow* [184]; GC-SDT [39]; GA-FID3 [160]; FVBDT [158] |
| GS | AP | LGA | 10 | Caltrop [7]; *Bandar* et al. [87]; MEPDTI [171]; *Cha & Tappert* [175,176]; ECCO [155]; evtree [3]; BiLeGA [153]; *Ersoy* et al. [178]; *Smith* [177]; EVO-Tree [157] |
| | | TGA | 15 | GAIT [102]; GAIT [204,211–214]; *Sorensen & Janssens* [88]; *Bosnjak* et al. [210]; genTrees [168]; GATree [163,206]; GDT [9]; *Biedrzycki & Arabas* [215]; GDT-MC [165]; *Jurczuk* et al. [207–209]; GDT-MA [164]; MS [113]; LEGAL-Tree [10,205] |
| | | GP | 42 | *Koza* [4]; *Iba* et al. [247]; GPDT [238]; EDDIE [230]; *Shirasaka* et al. [263]; Cellular GP [226,253]; *Zhao & Shirasaka* [264]; *Oka & Zhao* [103]; *Tanigawa & Zhao* [265]; BGP [225,269]; *Niimi & Tazaki* [250]; EPTree [233]; GPTree [267,268]; *To & Pham* [254]; GPEI [240,272]; *Karakatič & Podgorelec* [273]; FGP [5,234]; *Johansson* et al. [260]; *Tur & Guvenir* [89]; *Ryan & Rayward-Smith* [252]; *Niimi & Tazaki* [249]; GPDTI [239]; *Kuo* et al. [257]; DTiGP [229]; MGP [242]; *Saremi & Yaghmaee* [275,276]; *Karakatič* et al. [274]; CGP [227]; *Johansson* et al. [259]; MGP [241]; EDDIE-101 [231]; *Haruyama & Zhao* [266]; *Eggermont* et al. [270,271]; EMO [232,248]; NHEMOtree [244]; MOGP [243]; CH-MOGP [228]; *Khoshgoftaar* et al. [255,256]; *Zhao* [258]; GCP/SA [296]; *Mitrofanov* [309] |
| | | DE | 2 | DE-ADT$^{SPV}$ [8]; *Dolotov & Zolotykh* [302] |
| | | ES | 1 | *Dolotov & Zolotykh* [302] |
| | | CEA | 4 | *Aitkenhead* [301]; SAEDT [286,298–300]; MPGA [284]; MPGT [285] |
| | | GEP | 2 | *Ferreira* [307]; *Wang* et al. [308]; GEPDT [294] |
| | | GE | 3 | *Ono & Kushida* [306]; GEDT [304]; MGEDT [295] |
| | | EDA | 1 | Ardennes [293] |
| | OB | LGA | 2 | GDTI [167]; EFTI [156] |
| | | TGA | 6 | TARGET [174]; GEA-ODT [217,218]; GDT-Mix [166]; *Krętowski & Popczyński* [219]; *Czajkowski* et al. [114]; *Reska* et al. [220] |
| | | DE | 2 | PDT [291,310]; DE-ODT [288] |
| | NL | LGA | 1 | GALE [162,182] |
| | | GP | 3 | *Mugambi & Hunter* [279]; G$^3$P [236,278];*Šprogar* [262] |
| | | GE | 1 | *Russo* et al. [311] |
| | SF | LGA | 1 | *Kym & Ryu* [188] |
| | | GP | 2 | Fuzzy-GP [235]; PFDT [246] |
| | | GEP | 1 | *Weihong* et al. [312] |
| SO | AP | LGA | 2 | *Chen* et al. [189]; *Brunello* et al. [190] |
| | SF | LGA | 3 | *Crockett* et al. [185]; G-DT [159]; IIVFDT [170] |

**Table 37**
SI-based MHs for DTI.

| Strategy | DT | MH | Num studies | Studies |
|---|---|---|---|---|
| RP | AP | PSO | 1 | *Chan* et al. [326] |
| GS | AP | ACO | 4 | ACDT [12,324]; cACDT [318]; ACO-DTree [317,323]; ATM [13] |
| | | PSO | 3 | TSO [321]; DT-MPSO [319]; *Fieldsend* [325] |
| | | BA | 1 | BTC [322] |
| SO | AP | PSO | 2 | APSO [327]; *Malik & Khan* [320] |

**Table 38**
HH-based approaches to build DTI methods.

| DT | MH | Num studies | Studies |
|---|---|---|---|
| AP | LGA | 6 | HHDT [336]; HEAD-DT [334,340]; *Jovanović* et al. [341]; AutoGA [333]; HEARpart [335]; MOHEAD-DT [337] |
| | GE | 2 | AutoGE [338]; ESC-GE [339] |

**Table 39**
MH-based approaches by type of strategy applied.

| MH Type | MH | RP | GS | SO | Total |
|---|---|---|---|---|---|
| SS | GRASP | 1 | | | 1 |
| | SLS | 1 | | | 1 |
| | SA | 2 | | 4 | 6 |
| | TS | 2 | | 1 | 3 |
| | VNS | | | 1 | 1 |
| | **Subtotal** | 6 | 0 | 6 | 12 |
| EA | LGA | 12 | 20 | 5 | 37 |
| | TGA | 1 | 15 | | 16 |
| | GP | 2 | 51 | | 53 |
| | DE | 4 | 4 | | 8 |
| | ES | 2 | 1 | | 3 |
| | CEA | | 4 | | 4 |
| | GEP | | 3 | | 3 |
| | GE | | 6 | | 6 |
| | EDA | | 1 | | 1 |
| | **Subtotal** | 21 | 105 | 5 | 131 |
| SI | ACO | | 4 | | 4 |
| | PSO | 1 | 3 | 2 | 6 |
| | BA | | 1 | | 1 |
| | **Subtotal** | 1 | 8 | 2 | 11 |
| **Total** | | 28 | 113 | 13 | 154 |

**Table 41**
MH-based approaches by fitness function.

| MH Type | MH | UF | AF | MF | Total |
|---|---|---|---|---|---|
| SS | GRASP | 1 | | | 1 |
| | SLS | 1 | | | 1 |
| | SA | 5 | 1 | | 6 |
| | TS | 2 | 1 | | 3 |
| | VNS | | 1 | | 1 |
| | **Subtotal** | 9 | 3 | 0 | 12 |
| EA | LGA | 28 | 7 | 2 | 37 |
| | TGA | 6 | 9 | 1 | 16 |
| | GP | 23 | 17 | 13 | 53 |
| | DE | 8 | | | 8 |
| | ES | 3 | | | 3 |
| | CEA | 1 | 3 | | 4 |
| | GEP | 3 | | | 3 |
| | GE | 3 | 2 | 1 | 6 |
| | EDA | 1 | | | 1 |
| | **Subtotal** | 76 | 38 | 17 | 131 |
| SI | ACO | 1 | 3 | | 4 |
| | PSO | 3 | 1 | 2 | 6 |
| | BA | 1 | | | 1 |
| | **Subtotal** | 5 | 4 | 2 | 11 |
| | **Total** | 90 | 45 | 19 | 154 |

**Table 40**
MH-based approaches for type of induced DT.

| MH Type | MH | AP | OB | NL | SF | Total |
|---|---|---|---|---|---|---|
| SS | GRASP | 1 | | | | 1 |
| | SLS | | 1 | | | 1 |
| | SA | 3 | 2 | | 1 | 6 |
| | TS | | 3 | | | 3 |
| | VNS | 1 | | | | 1 |
| | **Subtotal** | 5 | 6 | 0 | 1 | 11 |
| EA | LGA | 18 | 8 | 3 | 8 | 37 |
| | TGA | 13 | 3 | | | 16 |
| | GP | 43 | 4 | 4 | 2 | 53 |
| | DE | 3 | 5 | | | 8 |
| | ES | 1 | 2 | | | 3 |
| | CEA | 4 | | | | 4 |
| | GEP | 2 | | | 1 | 3 |
| | GE | 5 | | 1 | | 6 |
| | EDA | 1 | | | | 1 |
| | **Subtotal** | 90 | 22 | 8 | 11 | 131 |
| SI | ACO | 4 | | | | 4 |
| | PSO | 6 | | | | 6 |
| | BA | 1 | | | | 1 |
| | **Subtotal** | 11 | 0 | 0 | 0 | 11 |
| **Total** | | 106 | 28 | 8 | 12 | 154 |

**Table 42**
MH-based approaches by representation scheme.

| MH Type | MH | Linear | Tree | Matrix | Total |
|---|---|---|---|---|---|
| SS | GRASP | 0 | 1 | | 1 |
| | SLS | 1 | | | 1 |
| | SA | 2 | 4 | | 6 |
| | TS | 2 | 1 | | 3 |
| | VNS | | 1 | | 1 |
| | **Subtotal** | 5 | 7 | 0 | 12 |
| EA | LGA | 37 | | | 37 |
| | TGA | | 16 | | 16 |
| | GP | | 53 | | 53 |
| | DE | 8 | | | 8 |
| | ES | 3 | | | 3 |
| | CEA | | 4 | | 4 |
| | GEP | 3 | | | 3 |
| | GE | 6 | | | 6 |
| | EDA | | 1 | | 1 |
| | **Subtotal** | 57 | 74 | 0 | 131 |
| SI | ACO | | 2 | 2 | 4 |
| | PSO | 5 | | 1 | 6 |
| | BA | 0 | 1 | | 1 |
| | **Subtotal** | 5 | 3 | 3 | 12 |
| | **Total** | 67 | 84 | 3 | 154 |

**Table 43**
MH-based approaches by fitness measure.

| MH type | MH | Splitting criteria | | | | Performance measures | | | | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| | | IG | Twoing | Gini | Other | Acc | Size | Error | Other | |
| SS | GRASP | 1 | | | | | | | | 1 |
| | SLS | 1 | 1 | 1 | 3 | | | | | 6 |
| | SA | | 1 | 1 | 2 | | 1 | 2 | | 7 |
| | TS | 1 | | | 1 | | | 2 | | 4 |
| | VNS | | | | | 1 | 1 | | | 2 |
| | **Subtotal** | 3 | 2 | 2 | 6 | 1 | 2 | 4 | 0 | 20 |
| EA | LGA | 3 | 2 | 3 | 2 | 17 | 10 | 6 | 3 | 46 |
| | TGA | | | | | 12 | 10 | 3 | 3 | 28 |
| | GP | 3 | | | 4 | 29 | 24 | 10 | 18 | 88 |
| | DE | 1 | 1 | | 2 | 3 | | 1 | | 8 |
| | ES | 1 | 1 | | 1 | 1 | | | | 4 |
| | CEA | | | | | 4 | 2 | | 1 | 7 |
| | GEP | | | | | 3 | | | | 3 |
| | GE | | | | | 2 | 1 | 1 | 5 | 9 |
| | EDA | | | | | 1 | | | | 1 |
| | **Subtotal** | 8 | 4 | 3 | 9 | 72 | 47 | 21 | 30 | 194 |
| SI | ACO | | 2 | | 1 | | | 1 | 3 | 7 |
| | PSO | | | | 1 | 2 | 2 | 3 | 2 | 10 |
| | BA | | | | | 1 | | | | 1 |
| | **Subtotal** | 0 | 2 | 0 | 2 | 3 | 2 | 4 | 5 | 18 |
| | **Total** | 11 | 8 | 5 | 17 | 76 | 51 | 29 | 35 | 232 |

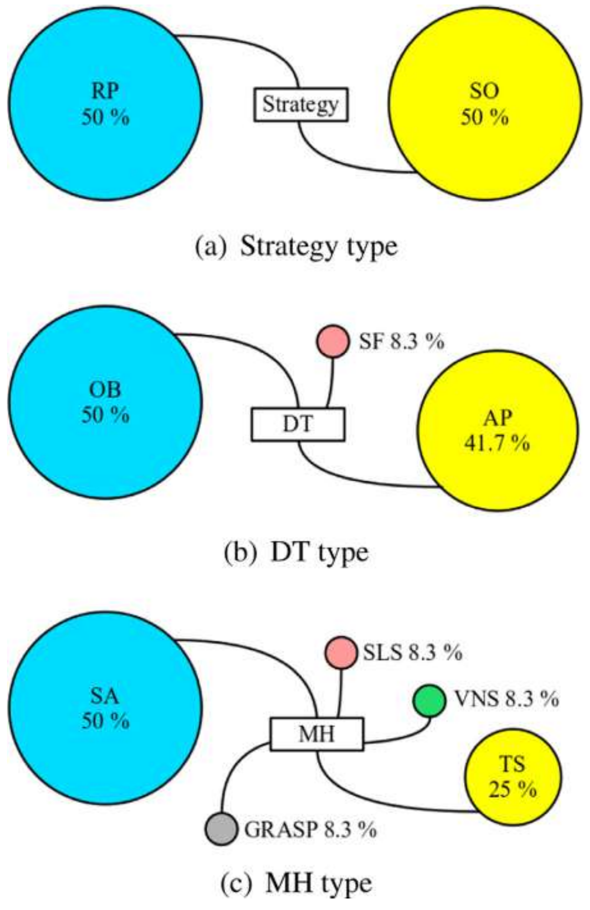

**Fig. 31.** Distribution of SS-based MHs for DTI.

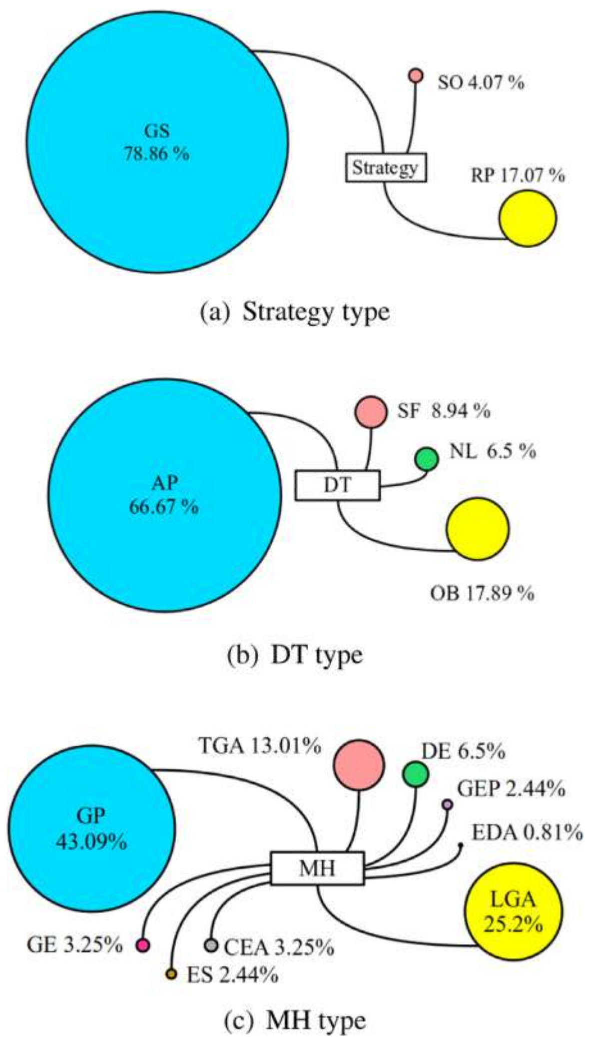

**Fig. 32.** Distribution of EA-based MHs for DTI.

## 9. Open questions and conclusions

DTI algorithms stand out from other machine learning techniques since they are simple procedures creating more accurate models with a high interpretability level. An essential aspect of increasing the model

**Table 44**
EA-based approaches by selection operator.

| MH | Tournament | Roulette | Linear Ranking | Special | Truncate | Other | Total |
|---|---|---|---|---|---|---|---|
| LGA | 12 | 11 | 1 | 2 | | 1 | 27 |
| TGA | 3 | 4 | 4 | 1 | | 1 | 13 |
| GP | 37 | 5 | | 3 | 5 | | 50 |
| DE | 8 | | | | | | 8 |
| ES | | | | 1 | | | 1 |
| CEA | 1 | | 2 | | | | 3 |
| GEP | | 3 | | | | | 3 |
| GE | 6 | | | | | | 6 |
| EDA | | | | | 1 | | 1 |
| **Total** | 67 | 23 | 7 | 7 | 6 | 2 | 112 |

**Table 45**
EA-based approaches by crossover operator.

| MH | Linear chromosome | | | | Tree-based chromosome | | | | Total |
|---|---|---|---|---|---|---|---|---|---|
| | Single-point | Double-point | Special | Other | Switch sub-trees | Switch nodes | Switch branches | Other | |
| LGA | 10 | 8 | 5 | 6 | | | | | 29 |
| TGA | | | | | 15 | 7 | 4 | | 26 |
| GP | | | | | 46 | | 1 | 4 | 51 |
| DE | | | | 8 | | | | | 8 |
| ES | | | 1 | | | | | | 1 |
| CEA | | | | | 3 | | | | 3 |
| GEP | 3 | 3 | 3 | | | | | | 9 |
| GE | 5 | | | | 1 | | | | 6 |
| **Total** | 18 | 11 | 9 | 14 | 65 | 7 | 5 | 4 | 133 |

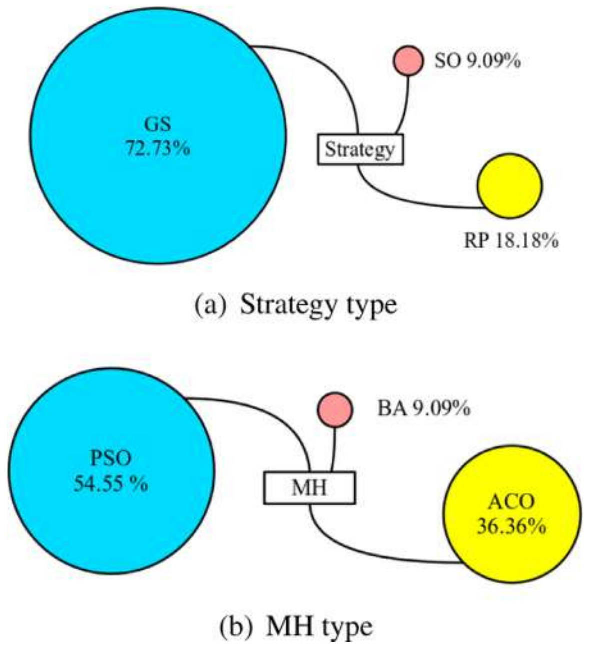

**Fig. 33.** Distribution of SI-based MHs for DTI.

utility is reducing this complexity level. In DTs, it is associated with its size, then it is critical to generate trees with the least number of nodes.

### 9.1. Implementation strategies

For avoiding the existing problems of greedy-search-based approaches, several MHs have been used to build DTs. This review identifies three strategies to use MHs as DTI algorithms: recursive partitioning, global search, and subsequent optimization.

Swarm and evolutionary computation methods successfully perform a global search in complex spaces, and EAs have been extensively applied to induce DTs. Although SI methods are remarkably effective in solving numerical optimization problems, their use in this field is limited. In both cases, several challenges must be faced, such as the solution representation, the creation of operators to guarantee the induction of feasible DTs and the issues associated with the particular characteristics of each MH.

Two drawbacks of these approaches are (1) their dependence on the definition and setting of a set of control parameters and (2) the time required for a population of DTs to converge to near-optimal solutions. In particular, GP-based approaches for DTI involve developing strategies to preserve population diversity and avoid the bloat phenomenon.

However, the main challenge in these approaches is to determine the fitness value of each candidate solution. Since each one is evaluated using the training instances, the processing time increases depending on the dataset size. A recently adopted option to deal with this problem is through parallel induction approaches to spread the processing charge into several computing units.

Alternatively, MHs have also been used to replace the traditional splitting criteria in the induction process. They can (1) solve the selection bias problem in axis-parallel DTs by adjusting their fitness function and (2) improve multivariate test conditions used on oblique and non-linear DTs. However, overfitting and instability of small changes in the training set persist since these problems are inherent to a greedy strategy. Here, SS-based MHs are well placed to implement a recursive partitioning strategy, although they can search in any solution space using suitable representation schemes. One advantage of these MHs is that they consume less time in their searching process than that used for population-based methods.

Finally, the use of MHs to improve a DT generated using another technique allows preserving the model's general structure, and by refining its elements, it can improve its predictive power, but always limited to the neighborhood of the original model.

### 9.2. Types of decision trees

MHs can induce any DT type, but axis-parallel DTs are the most common model built with them. Although multivariate and soft DTs can be more compact and better than the first, their induction through an MH is a little-studied approach. Non-linear DTs with few nodes can be con-

**Table 46**
EA-based approaches by mutation operator.

| MH | Linear chromosome | | | | Tree-based chromosome | | | | Total |
|---|---|---|---|---|---|---|---|---|---|
| | Bit-String | Uniform | Special | Other | Switch sub-trees | Distub node | Replace Node | Other | |
| LGA | 7 | 10 | 10 | 2 | | | | | 29 |
| TGA | | | | | | 10 | 7 | 16 | 33 |
| GP | | | | | 36 | 11 | 3 | 9 | 59 |
| DE | | | | 8 | | | | | 8 |
| ES | | | 3 | | | | | | 3 |
| CEA | | | | | 3 | 4 | | | 7 |
| GEP | | 3 | 3 | | | | | | 6 |
| GE | 3 | 2 | | | 1 | | | | 6 |
| **Total** | 10 | 15 | 16 | 10 | 40 | 25 | 10 | 25 | 151 |

**Table 47**
MH-based approaches by sampling method.

| MH Type | MH | CV | Hold-out | Complete | Other | Total |
|---|---|---|---|---|---|---|
| SS | GRASP | 1 | | | | 1 |
| | SLS | 1 | | | | 1 |
| | SA | 2 | 2 | 1 | 1 | 6 |
| | TS | 3 | | | | 3 |
| | VNS | 1 | | | | 1 |
| | **Subtotal** | 8 | 2 | 1 | 1 | 12 |
| EA | LGA | 13 | 4 | | 1 | 18 |
| | TGA | 17 | 17 | 2 | 4 | 40 |
| | GP | 25 | 23 | 3 | 1 | 52 |
| | DE | 7 | 2 | | 1 | 10 |
| | ES | 1 | 1 | | 1 | 3 |
| | CEA | 2 | 3 | 1 | | 6 |
| | GEP | 2 | 1 | 1 | | 4 |
| | GE | 2 | 3 | 1 | | 6 |
| | EDA | 1 | | | | 1 |
| | **Subtotal** | 70 | 54 | 8 | 8 | 140 |
| SI | ACO | 2 | 4 | | | 6 |
| | PSO | 2 | 2 | 2 | | 6 |
| | BA | 1 | | | | 1 |
| | **Subtotal** | 5 | 6 | 2 | 0 | 13 |
| | **Total** | 83 | 62 | 11 | 9 | 165 |

**Table 48**
MH-based approaches by performance measure.

| MH Type | MH | Accuracy | Size | Error | Time | F-Score | Other | Total |
|---|---|---|---|---|---|---|---|---|
| SS | SLS | 1 | 1 | | | | | 2 |
| | SA | 2 | 4 | 3 | 1 | | | 10 |
| | TS | 1 | 2 | 2 | 2 | | | 7 |
| | VNS | 1 | 1 | | 1 | | | 3 |
| | **Subtotal** | 6 | 9 | 5 | 4 | 0 | 0 | 24 |
| EA | LGA | 14 | 11 | 3 | 4 | 3 | 1 | 36 |
| | TGA | 32 | 23 | 3 | 8 | 2 | 3 | 71 |
| | GP | 31 | 17 | 11 | 5 | 2 | 10 | 76 |
| | DE | 7 | 5 | | 1 | 1 | | 14 |
| | ES | 3 | 2 | | 1 | | | 6 |
| | CEA | 6 | 4 | | | 1 | 6 | 17 |
| | GEP | 4 | | | | | | 4 |
| | GE | 3 | 3 | 2 | 1 | 2 | 1 | 12 |
| | EDA | 1 | 1 | | | | | 2 |
| | **Subtotal** | 101 | 66 | 19 | 20 | 11 | 21 | 238 |
| SI | ACO | 4 | 4 | 1 | 1 | | | 10 |
| | PSO | 4 | 3 | 2 | 1 | | 4 | 14 |
| | BA | 1 | 1 | | | | | 2 |
| | **Subtotal** | 9 | 8 | 3 | 2 | 0 | 4 | 26 |
| | **Total** | 116 | 83 | 27 | 26 | 11 | 25 | 288 |

structed using an MH. However, they have a less expressive power than that of axis-parallel DTs. A possible research line is to find a minimal combination of attributes in each internal node providing the same interpretation level as one axis-parallel DT.

## *9.3. Experimental studies*

Similarly to other studies where some DTI method are analyzed, most of the studies detailed in this review focus on describing how an MH is used to build a DT, and statistical analysis is not conducted. A large percentage only present comparisons of the experimental results and the complexity of the induced DT or use the average results and then apply a paired *t*-test. Just 13 of the 154 studies in the existing literature use a post-hoc test to compare the performance of their methods with other classifiers.

**Table 49**
MH-based approaches by statistical test.

| MH Type | MH | Friedman | *t*-test | Wilcoxon | Nemenyi | Avg | WTL | Other | Total |
|---|---|---|---|---|---|---|---|---|---|
| SS | GRASP | | 1 | | | 1 | 1 | | 3 |
| | SA | | 1 | | | | | | 1 |
| | TS | | | | | 1 | | 2 | 3 |
| | **Subtotal** | 0 | 2 | 0 | 0 | 2 | 1 | 2 | 7 |
| EA | LGA | 4 | 3 | 1 | 4 | | | 2 | 14 |
| | TGA | 2 | 6 | 1 | 1 | 5 | | 5 | 20 |
| | GP | 3 | 1 | 8 | 1 | 2 | 3 | 5 | 23 |
| | DE | 4 | 1 | | 2 | | | 3 | 10 |
| | ES | | 1 | | | | | | 1 |
| | CEA | 1 | | | 1 | | | | 2 |
| | GE | 2 | | | 1 | | | 1 | 4 |
| | EDA | | | | | | | 1 | 1 |
| | **Subtotal** | 16 | 12 | 10 | 10 | 7 | 3 | 17 | 75 |
| SI | ACO | 2 | | 1 | | | | 1 | 4 |
| | **Subtotal** | 2 | 0 | 1 | 0 | 0 | 0 | 1 | 4 |
| | **Total** | 18 | 14 | 11 | 10 | 9 | 4 | 20 | 86 |

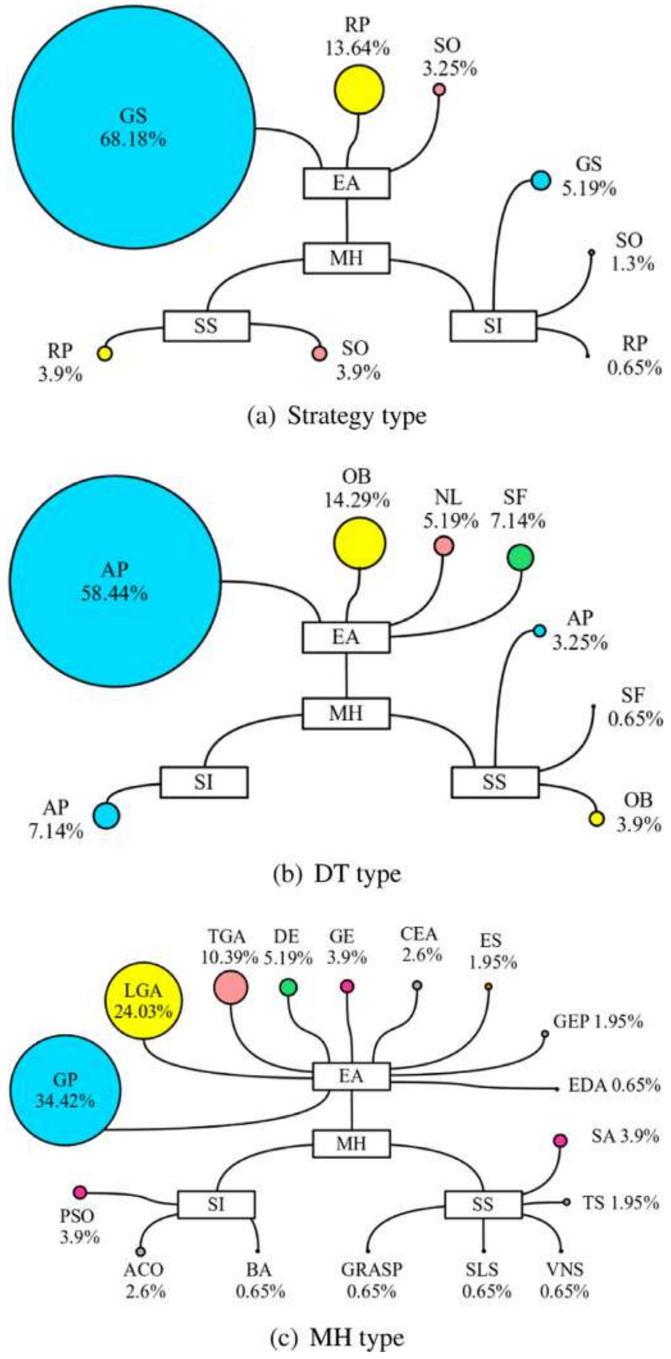

**Fig. 34.** Distribution of MH-based approaches for DTI.

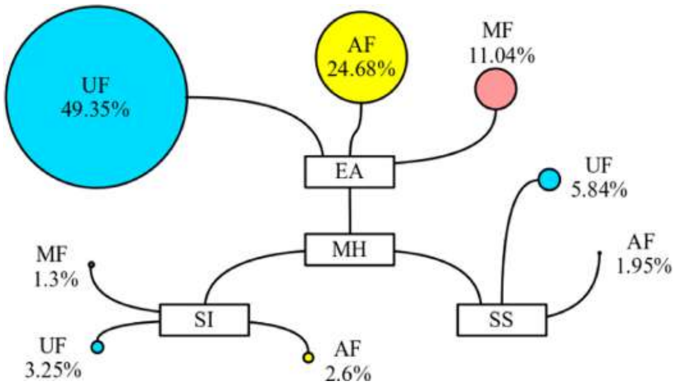

**Fig. 35.** Distribution of fitness functions.

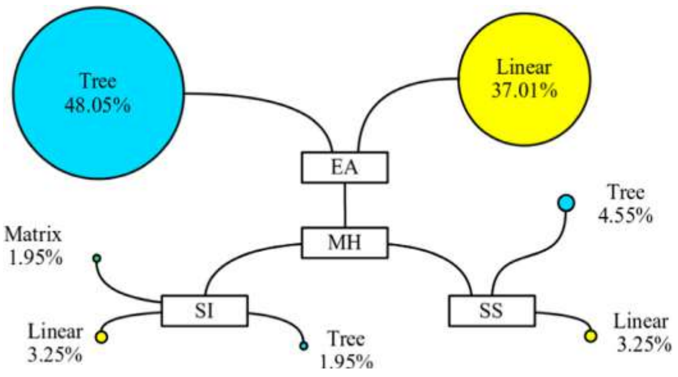

**Fig. 36.** Distribution of representation schemes.

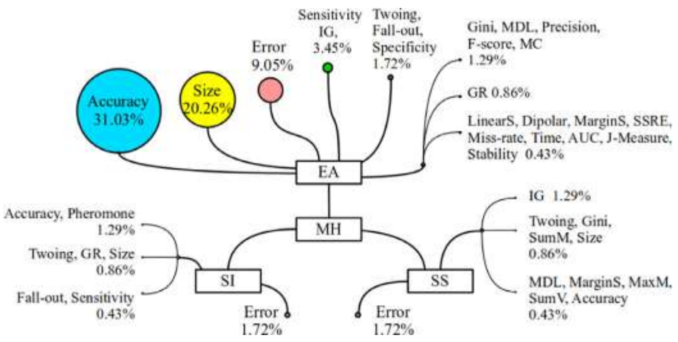

**Fig. 37.** Distribution of fitness measures.

### 9.4. Final remarks

This state-of-the-art review is developed to lump together all MH-based methods to build DTs under the same analytical perspective. This work shows the close relationship between machine learning and swarm and evolutionary computation methods, intending to provide better solutions to real problems in science and engineering. DTs are necessary when it is required that a predictive model be sufficiently understandable, and a decision-maker can reproduce a sequence of decisions taken to discriminate one case of study from another. As an extension of the review described in this manuscript, our next step is to conduct a study on the use of MHs to implement other types of DT-based classifiers such as random forests and multi-trees.

The use of MH for DTI has provided a new approach to building classifiers with better performance, especially by performing a global search in the solution space. Even though its application to induce DTs began three decades ago, currently presents many opportunities to study and new challenges since there are characteristics of MHs that must be analyzed in the machine learning context.

Since DTs are classifiers distinguished by their simplicity and high level of interpretability, the implementation of new approaches to induce near-optimal DTs is highly important in emerging research areas such as biotechnology, neuroscience, genomics, proteomics, and medical decision making. The use of MH-based approaches for DTI is a promising trend to build classifiers with higher performance. These schemes can combine their exploration and exploitation skills, providing a better way to discover the relationships between the attributes used in the training set. Another potential development area is applying SI-based MHs for DTI by defining schemes allowing an efficient search in the tree space using SI procedures to update the particle position in the swarm. Furthermore, the use of grammar-based MHs can guaran-

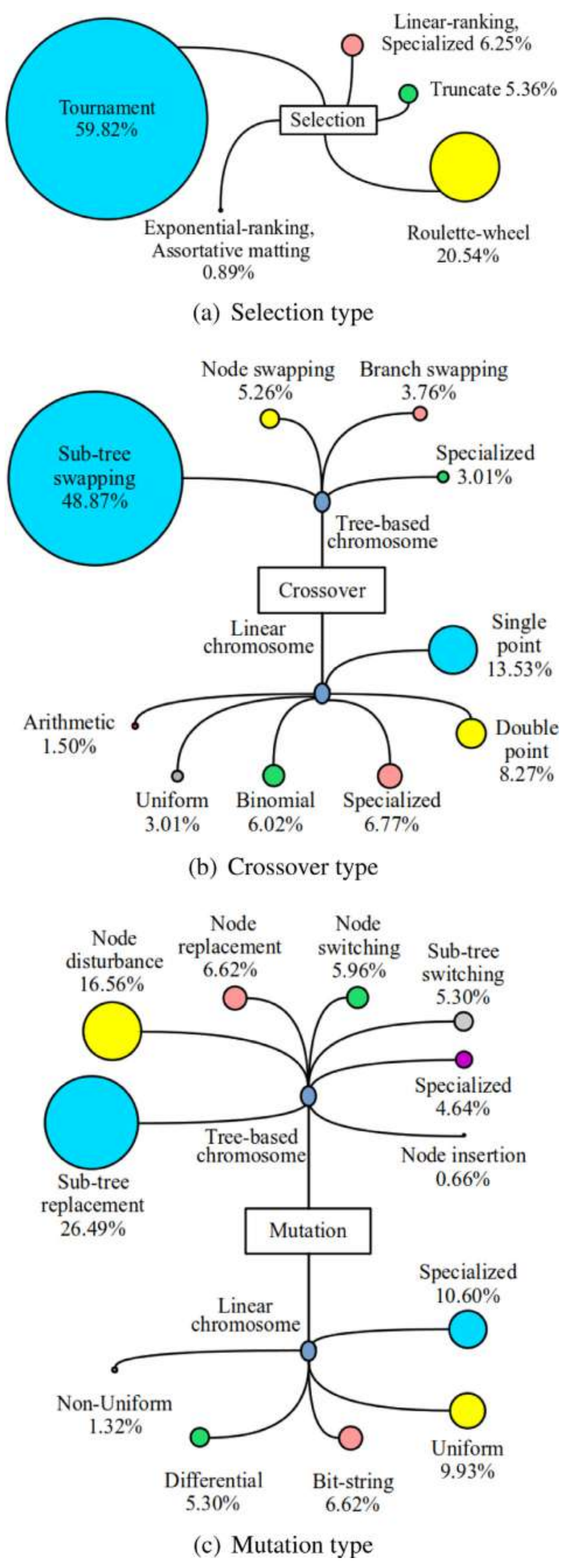

**Fig. 38.** Distribution of genetic operators.

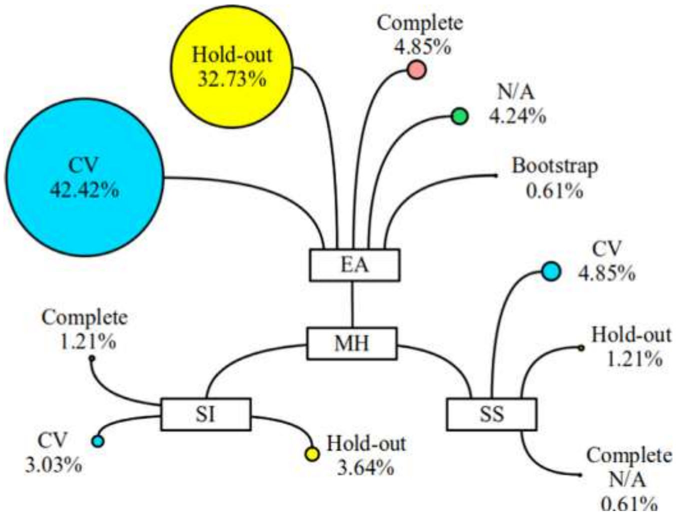

**Fig. 39.** Distribution of sampling methods.

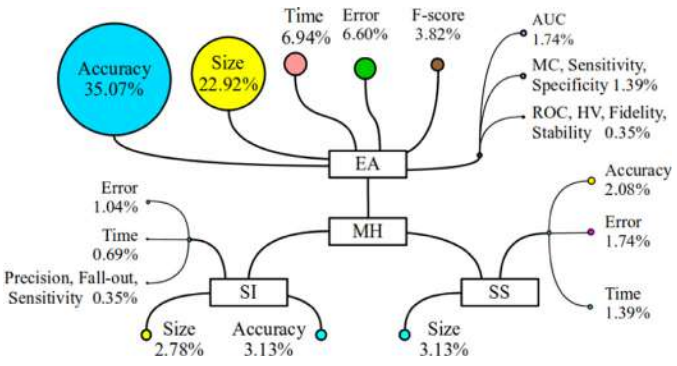

**Fig. 40.** Distribution of performance measures.

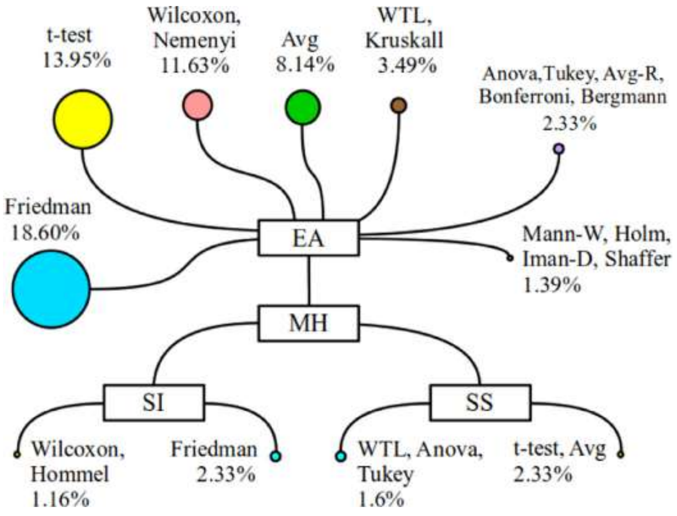

**Fig. 41.** Distribution of statistical tests.

tee the construction of only feasible candidate solutions. Finally, hyper-heuristics to build algorithms that induce DTs are a strategy that has been gaining interest recently.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper

## CRediT authorship contribution statement

**Rafael Rivera-Lopez:** Conceptualization, Methodology, Formal analysis, Investigation, Writing – original draft. **Juana Canul-Reich:** Conceptualization, Methodology, Formal analysis, Investigation, Supervision, Writing – review & editing. **Efrén Mezura-Montes:** Methodology, Formal analysis, Writing – review & editing, Supervision. **Marco Antonio Cruz-Chávez:** Methodology, Formal analysis, Writing – review & editing.

## References

[1] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, 3Rd Edition, Morgan Kaufmann, 2012, doi:10.1016/C2009-0-61819-5. Boston

[2] G.K. Gupta, Introduction to Data Mining with Case Studies, 3Rd Edition, PHI Learning, 2014. Delhi

[3] T. Grubinger, A. Zeileis, K.P. Pfeiffer, Evtree: evolutionary learning of globally optimal classification and regression trees in r, J. Stat. Software 61 (1) (2014) 1–29, doi:10.18637/jss.v061.i01.

[4] J.R. Koza, Concept Formation and Decision Tree Induction Using the Genetic Programming Paradigm, in: H.P. Schwefel (Ed.), PPSN 1990, volume 496, Springer, Berlin, 1991, pp. 124–128, doi:10.1007/BFb0029742. Of LNCS

[5] E.P.K. Tsang, J. Li, Combining Ordinal Financial Predictions with Genetic Programming, in: K.S. Leung (Ed.), IDEAL 2000, 1983 of LNCS, Springer, Berlin, 2000, pp. 532–537, doi:10.1007/3-540-44491-2_77.

[6] M.C.J. Bot, W.B. Langdon, Application of Genetic Programming to Induction of Linear Classification Trees, in: R. Poli (Ed.), EuroGP 2000, Vol. 1802 of LNCS, Springer, Berlin, 2000, pp. 247–258, doi:10.1007/978-3-540-46239-2_18.

[7] H. Kennedy, C. Chinniah, P.V.G. Bradbeer, L. Morss, The Construction and Evaluation of Decision Trees: A Comparison of Evolutionary and Concept Learning Methods, in: D. Corne (Ed.), AISB EC 1997, Vol. 1305 of LNCS, Springer, Manchester, 1997, pp. 147–162, doi:10.1007/BFb0027172.

[8] R. Rivera-Lopez, J. Canul-Reich, Construction of near-optimal axis-parallel decision trees using a differential-evolution-based approach, IEEE Access 6 (2018) 5548–5563, doi:10.1109/ACCESS.2017.2788700.

[9] M.K. etowski, M. Grześ, Global Learning of Decision Trees by an Evolutionary Algorithm, in: K. Saeed (Ed.), Information Processing and Security Systems, Springer, Boston, 2005, pp. 401–410, doi:10.1007/0-387-26325-X_36.

[10] M.P. Basgalupp, A.C.P.L.F. Carvalho, R.C. Barros, A.A. D. D. Ruiz, Freitas, lexicographic multi-objective evolutionary induction of decision trees, Int. J. of Bio-Inspired Computation 1 (1–2) (2009) 105–117, doi:10.1504/IJBIC.2009.022779.

[11] E. Cantú-Paz, C. Kamath,et al. Using evolutionary algorithms to induce oblique decision trees, D. Whitley, 2000, GECCO-00, Morgan Kaufmann. 1053–1060,10.5555/2933718.2933916

[12] U. Boryczka, J. Kozak, Ant Colony Decision Trees - a New Method for Constructing Decision Trees Based on Ant Colony Optimization, in: J.S. Pan (Ed.), ICCCI 2010, Part I, Vol. 6421 of LNAI, Springer, 2010, pp. 373–382, doi:10.1007/978-3-642-16693-8_39.

[13] F.E.B. Otero, A.A. Freitas, C.G. Johnson, Inducing decision trees with an ant colony optimization algorithm, Appl Soft Comput (2012), doi:10.1016/j.asoc.2012.05.028.

[14] T. Lal, O. Chapelle, J. Weston, A. Elisseeff, Embedded Methods, in: I. Guyon (Ed.), Feature Extraction: Foundations and Applications, Vol. 207 of Studies in Fuzziness and Soft Computing, Springer, 2006, pp. 137–165, doi:10.1007/978-3-540-35488-8_6.

[15] J. Mingers, An empirical comparison of pruning methods for decision tree induction, Mach Learn 4 (2) (1989) 227–243, doi:10.1023/A:1022604100933.

[16] S.R. Safavian, D. Landgrebe, A survey of decision tree classifier methodology, IEEE Trans. on Systems, Man and Cybernetics 21 (3) (1991) 660–674, doi:10.1109/21.97458.

[17] C. Brodley, P. Utgoff, Multivariate decision trees, Mach Learn 19 (1) (1995) 45–77, doi:10.1023/A:1022607123649.

[18] F. Esposito, D. Malerba, G. Semeraro, J. Kay, A comparative analysis of methods for pruning decision trees, IEEE Trans. on Pattern Analysis and Machine Intelligence 19 (5) (1997) 476–491, doi:10.1109/34.589207.

[19] L. Breslow, D. Aha, Simplifying decision trees: a survey, Knowl Eng Rev 12 (01) (1997) 1–40, doi:10.1017/S0269888997000015.

[20] S.K. Murthy, Automatic construction of decision trees from data: amulti-disciplinary survey, Data Min Knowl Discov 2 (4) (1998) 345–389, doi:10.1023/A:1009744630224.

[21] L. Rokach, O. Maimon, Top-down induction of decision trees classifiers – a survey, IEEE Trans. on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 35 (4) (2005) 476–487, doi:10.1109/TSMCC.2004.843247.

[22] S.B. Kotsiantis, Decision trees: a recent overview, Artif Intell Rev 39 (4) (2013) 261–283, doi:10.1007/s10462-011-9272-4.

[23] S. Lomax, S. Vadera, A survey of cost-sensitive decision tree induction algorithms, ACM Comput Surv 45 (2) (2013) 16:1–16:35, doi:10.1145/2431211.2431215.

[24] W.Y. Loh, Fifty years of classification and regression trees, Int. Statistical Review 82 (3) (2014) 329–348, doi:10.1111/insr.12016.

[25] R.C. Barros, A.C.P.L.F. Carvalho, A.A. Freitas, Automatic Design of Decision-tree Induction Algorithms, in: Ch. Decision-Tree Induction, Springer, 2015, pp. 7–45, doi:10.1007/978-3-319-14231-9_2.

[26] M.K. etowski, Evolutionary decision trees in large-Scale data mining, Springer, 2019, doi:10.1007/978-3-030-21851-5.

[27] Y. Peng, P.A. Flach, Soft discretization to enhance the continuous decision tree induction, Integrating Aspects of Data Mining, Decision Support and Meta-Learning 1 (109–118) (2001) 34.

[28] R.C. Barros, M.P. Basgalupp, A.C.P.L.F. Carvalho, A.A. Freitas, A survey of evolutionary algorithms for decision-tree induction, IEEE Trans. on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42 (3) (2012) 291–312, doi:10.1109/TSMCC.2011.2157494.

[29] X.Z. Wang, D.S. Yeung, E.C.C. Tsang, A comparative study on heuristic algorithms for generating fuzzy decision trees, IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics 31 (2) (2001) 215–226, doi:10.1109/3477.915344.

[30] J.R. Quinlan, C4.5: Programs for machine learning, Morgan Kaufmann (1993).

[31] J.R. Quinlan, Induction of decision trees, Mach Learn 1 (1) (1986) 81–106, doi:10.1007/BF00116251.

[32] D.L. Davies, D.W. Bouldin, A cluster separation measure, IEEE Trans. on Pattern Analysis and Machine Intelligence PAMI- 1 (2) (1979) 224–227, doi:10.1109/TPAMI.1979.4766909.

[33] M.K. etowski, 2004, L. Rutkowski An evolutionary algorithm for oblique decision tree inductionICAISC 2004, Vol. 3070 of LNAI, Springer, Zakopane, Poland, 432–437, 10.1007/978-3-540-24844-6_63,

[34] L. Breiman, J. Friedman, R. Olshen, C. Stone, Classification and regression trees, 1984, (????). Chapman and Hall.

[35] S. Shah, P.S. Sastry, New algorithms for learning and pruning oblique decision trees, IEEE Trans. on Systems, Man and Cybernetics, Part C: Applications and Reviews 29 (4) (1999) 494–505, doi:10.1109/5326.798764.

[36] C. Orsenigo, C. Vercellis, Discrete support vector decision trees via tabu search, Computational statistics & data analysis 47 (2) (2004) 311–322, doi:10.1016/j.csda.2003.11.005.

[37] D.G. Heath, S. Kasif, S. Salzberg, Induction of oblique decision trees, 1993, R. Bajcsy, IJCAI-93, Chambéry, France. 1002–1007

[38] J.R. Quinlan, R.L. Rivest, Inferring decision trees using the minimum description lenght principle, Information and computation 80 (3) (1989) 227–248, doi:10.1016/0890-5401(89)90010-2.

[39] S.K. Shukla, M.K. Tiwari, Soft decision trees: a genetically optimized cluster oriented approach, Expert Syst Appl 36 (1) (2009) 551–563, doi:10.1016/j.eswa.2007.09.065.

[40] V.E. Lee, L. Liu, R. Jin, Decision Trees: Theory and Algorithms, in: Data Classification: Algorithms and Applications, CRC Press, 2014, pp. 87–120.

[41] J.R. Quinlan, Simplifying decision trees, Int. Journal of Human-Computer Studies 27 (3) (1987) 221–234, doi:10.1006/ijhc.1987.0321.

[42] R. Reed, Pruning algorithms-a survey, IEEE Trans. on Neural Networks 4 (5) (1993) 740–747, doi:10.1109/72.248452.

[43] S. Mitra, T. Acharya, Data mining: multimedia, soft computing, and bioinformatics, Wiley, 2005.

[44] P. Geurts, Contributions to decision tree induction: bias/variance tradeoff and time series classification, Ph.d. thesis, University of Liége, Belgium

[45] T. Hothorn, K. Hornik, A. Zeileis, Unbiased recursive partitioning: a conditional inference framework, Journal of Computational and Graphical Statistics 15 (3) (2006) 651–674, doi:10.1198/106186006X133933.

[46] A.P. White, W.Z. Liu, Bias in information-based measures in decision tree induction, Mach Learn 15 (3) (1994) 321–329, doi:10.1023/A:1022694010754.

[47] C. Strobl, J. Malley, G. Tutz, An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests, Psychol Methods 14 (4) (2009) 323–348, doi:10.1037/a0016973.

[48] P.E. Utgoff, Incremental induction of decision trees, Mach Learn 4 (2) (1989) 161–186, doi:10.1023/A:1022699900025.

[49] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5–32, doi:10.1023/A:1010933404324.

[50] D. Bertsimas, J. Dunn, Optimal classification trees, Mach Learn 106 (7) (2017) 1039–1082, doi:10.1007/s10994-017-5633-9.

[51] L. Hyafil, R.L. Rivest, Constructing optimal binary decision trees is NP-complete, Inf Process Lett 5 (1) (1976) 15–17, doi:10.1016/0020-0190(76)90095-8.

[52] I.H. Witten, E. Frank, Data mining: practical machine learning tools and techniques, Morgan Kaufmann (2005).

[53] M. Birattari, Tuning Metaheuristics: A Machine Learning Perspective, Vol. 197 of Studies in Computational Intelligence, Springer, 2009, doi:10.1007/978-3-642-00483-4.

[54] K. Du, M. Swamy, Search and optimization by metaheuristics, Springer, 2016, doi:10.1007/978-3-319-41192-7.

[55] E.G. Talbi, Metaheuristics: from design to implementation, Wiley, 2006.

[56] T. Vicsek, A. Zafeiris, Collective motion, Phys Rep 517 (3–4) (2012) 71–140, doi:10.1016/j.physrep.2012.03.004.

[57] T.A. Feo, M.G.C. Resende, A probabilistic heuristic for a computationally difficult set covering problem, Operations Research Letters 8 (2) (1989) 67–71, doi:10.1016/0167-6377(89)90002-3.

[58] S. Kirkpatrick, D.C. Gelatt, M.P. Vecchi, Optimization by simmulated annealing, Science 220 (4598) (1983) 671–680, doi:10.1126/science.220.4598.671.

[59] H.H. Hoos, Stochastic local search - methods, models, applications, 1998 Ph.d. thesis. Darmstadt University of Technology
[60] F. Glover, Tabu search - part i, ORSA Journal on Computing 1 (3) (1989) 190–206, doi:10.1287/ijoc.1.3.190.
[61] N. Mladenović, P. Hansen, Variable neighborhood search, Computers & Operations Research 24 (11) (1997) 1097–1100, doi:10.1016/S0305-0548(97)00031-2.
[62] W.D. Hillis, Co-evolving parasites improve simulated evolution as an optimization procedure, Physica D 42 (1–3) (1990) 228–234, doi:10.1016/0167-2789(90)90076-2.
[63] M.A. Potter, K.A. DeJong, Y. Davidor, A Cooperative Coevolutionary Approach to Function Optimization (1994) 249–257. 10.1007/3-540-58484-6_269PPSN III, Vol. 866 of LNCS, Springer,
[64] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359, doi:10.1023/A:1008202821328.
[65] H. Mühlenbein, G. Paaß, From recombination of genes to the estimation of distributions i. binary parameters, Springer, 1996, pp. 178–187, doi:10.1007/3-540-61723-X_982. PPSN IV, Vol. 1141 of LNCS
[66] I. Rechenberg, Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution, Problemata, 15, Frommann-Holzboog (1973).
[67] H.P. Schwefel, Evolutionsstrategie und numerische optimierung, 1975, (????). Technische Universität Berlin.
[68] J.H. Holland, U. Michigan, Adaptation in natural and artificial systems, 1975, (????). Press.
[69] C. Ryan, J.J. Collins, M.O. Neill, Rammatical evolution: Evolving programs for an arbitrary language, Springer, 1998, pp. 83–96, doi:10.1007/BFb0055930. EuroGP'98, Vol. 1291 of LNCS
[70] C. Ferreira, Gene expression programming: a new adaptive algorithm for solving problems, Complex Systems 13 (2) (2001) 87–129.
[71] P. Whigham, Grammatically-based Genetic Programming, in: Workshop on Genetic Programming: from theory to real-world applications, 1995, pp. 33–41.
[72] J.R. Koza, N. Sridharan Hierarchical genetic algorithms operating on populations of computer programs, 1989, IJCAI'89, Morgan Kauffman. 768–774,
[73] D.J. Montana, Strongly typed genetic programming, Evol Comput 3 (2) (1995) 199–230, doi:10.1162/evco.1995.3.2.199.
[74] M. Dorigo, Optimization, learning and natural algorithms, 1992 Ph.d. thesis. Politecnico di Milano
[75] X.S. Yang, A New Metaheuristic Bat-inspired Algorithm, in: J.R. González (Ed.), NICSO 2010, SCI, Springer, 2010, pp. 65–74, doi:10.1007/978-3-642-12538-6_6.
[76] R.C. Eberhart, J. Kennedy, A New Optimizer Using Particle Swarm Theory, in: MHS'95, volume 1, IEEE, 1995, pp. 39–43, doi:10.1109/MHS.1995.494215.
[77] M. Galea, Q. Shen, J. Levine, Evolutionary approaches to fuzzy modelling for classification, Knowl Eng Rev 19 (1) (2004) 27–59, doi:10.1017/S0269888904000189.
[78] P.G. Espejo, S. Ventura, F. Herrera, A survey on the application of genetic programming to classification, IEEE Trans. on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 40 (2) (2010) 121–144, doi:10.1109/TSMCC.2009.2033566.
[79] H. Jabeen, A. Baig, Review of classification using genetic programming, Int. Journal of Engineering Science and Technology 2 (2) (2010) 94–103.
[80] P. Kokol, S. Pohorec, G. Štiglic, V. Podgorelec, Evolutionary design of decision trees for medical application, Data Min Knowl Discov 2 (3) (2012) 237–254, doi:10.1002/widm.1056.
[81] E. Kolçe, N. Frasheri, The use of heuristics in decision tree learning optimization, Int. Journal of Computer Engineering in Research Trends 1 (3) (2014) 127–130. https://www.ijcert.org/issue_des.php?id=520
[82] J. Kozak, Evolutionary computing techniques in data mining, Springer, 2019, pp. 29–44, doi:10.1007/978-3-319-93752-6_2.
[83] I. Bida, S. Aouat, Swarm Intelligence-based Decision Trees Induction for Classification – a Brief Analysis, in: (IHSH 2020), 2021, pp. 165–170, doi:10.1109/IHSH51661.2021.9378746.
[84] D.P. Muni, N.R. Pal, J. Das, A novel approach to design classifiers using genetic programming, IEEE Trans. on Evolutionary Computation 8 (2) (2004) 183–196, doi:10.1109/TEVC.2004.825567.
[85] M. Czajkowski, M. Grześ, M.K. etowski, Multi-test decision tree and its application to microarray data classification, Artif Intell Med 61 (1) (2014) 35–44, doi:10.1016/j.artmed.2014.01.005.
[86] B. Hemmateenejad, M. Shamsipur, V. Zare-Shahabadi, M. Akhond, Building optimal regression tree by ant colony system-genetic algorithm: Application to modeling of melting points, 2011, Anal. Chim. Acta 704, 1, 57–62, 10.1016/j.aca.2011.08.010
[87] Z. Bandar, H. Al-Attar, D. McLean, Genetic Algorithm Based Multiple Decision Tree Induction, in: ICONIP'99, volume 2, IEEE, Perth, Australia, 1999, pp. 429–434, doi:10.1109/ICONIP.1999.845633.
[88] K. Sörensen, G.K. Janssens, Data mining with genetic algorithms on binary trees, Eur J Oper Res 151 (2) (2003) 253–264, doi:10.1016/S0377-2217(02)00824-X.
[89] G. Tür, H.A. Güvenir, Decision Tree Induction Using Genetic Programming, in: E. Alpaydin (Ed.), TAINN'96, Bogazici University Press, 1996, pp. 187–196.
[90] G.V. Kass, An exploratory technique for investigating large quantities of categorical data, Journal of the Royal Statistical Society. Series C (Applied Statistics) 29 (2) (1980) 119–127, doi:10.2307/2986296.
[91] H. Kim, W.Y. Loh, Classification trees with unbiased multiway splits, J Am Stat Assoc 96 (454) (2001), doi:10.1198/016214501753168271.
[92] W.Y. Loh, Y.S. Shih, Split selection methods for classification trees, Stat Sin 7 (4) (1997) 815–840.
[93] P. Clark, T. Niblett, The CN2 induction algorithm, Mach Learn 3 (4) (1989) 261–283, doi:10.1007/BF00116835.
[94] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. on Information Theory 13 (1) (1967) 21–27, doi:10.1109/TIT.1967.1053964.
[95] D.R. Cox, The regression analysis of binary sequences, Journal of the Royal Statistical Society. Series B (Methodological) (1958) 215–242.
[96] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning Internal Representations by Error Propagation, in: J.A. Anderson (Ed.), Neurocomputing: Foundations of Research, MIT Press, 1988, pp. 673–695.
[97] D.S. Broomhead, D. Lowe, Radial basis functions, multi-variable functional interpolation and adaptive networks, Tech. rep., HMSO (1988).
[98] V. Vapnik, Estimation of Dependences Based on Empirical Data, Vol. 41 of Information Science and Statistics, Springer, 1982, doi:10.1007/0-387-34239-7.
[99] K. Deb, D. Kalyanmoy, Multi-Objective optimization using evolutionary algorithms, John Wiley & Sons, Inc., New York, NY, USA, 2001.
[100] C.A. Coello-Coello, G.B. Lamont, D.A.V. Veldhuizen, Evolutionary algorithms for solving multi-Objective problems (genetic and evolutionary computation), Springer, 2006, doi:10.1007/978-0-387-36797-2.
[101] R.S. Bucy, R.S. Diesposti, Classification tree optimization by simulated annealing, Summary report, The Aerospace Corporation (1991).
[102] Z. Fu, An innovative GA-based decision tree classifier in large scale data mining, Springer, 1999, pp. 348–353, doi:10.1007/978-3-540-48247-5_41. PKDD'99, Vol. 1704 of 1704
[103] S. Oka, Q. Zhao, Design of Decision Trees through Integration of C4.5 and GP, in: A. Namatame (Ed.), JAWIES 2000, 2000, pp. 128–135.
[104] D. Dua, C. Graff, UCI Machine learning repository, University of California, Irvine, School of Information and Computer Sciences (2019). http://archive.ics.uci.edu/ml
[105] C. Ferri, J. Hernández-Orallo, R. Modroiu, An experimental comparison of performance measures for classification, Pattern Recognit Lett 30 (1) (2009) 27–38, doi:10.1016/j.patrec.2008.08.010.
[106] M. Sokolova, G. Lapalme, A systematic analysis of performance measures for classification tasks, Information Processing & Management 45 (4) (2009) 427–437, doi:10.1016/j.ipm.2009.03.002.
[107] P. Cichosz, Assessing the quality of classification models: performance measures and evaluation procedures, Open Engineering 1 (2) (2011) 132–158, doi:10.2478/s13531-011-0022-9.
[108] J. Hanley, B. McNeil, A method of comparing the areas under receiver operating characteristic curves derived from the same cases, Radiology 148 (3) (1983) 839–843, doi:10.1148/radiology.148.3.6878708.
[109] M. Craven, J. Shavlik, Rule extraction: Where do we go from here?, 1999, (????). University of Wisconsin, Machine Learning Research Group, working Paper 99.
[110] E. Zitzler, L. Thiele, Multiobjective optimization using evolutionary algorithms — A comparative case study, Springer, 1998, pp. 292–301, doi:10.1007/BFb0056872. PPSN V
[111] P. Smyth, R.M. Goodman, An information theoretic approach to rule induction from databases, IEEE Trans. on Knowledge and Data Engineering 4 (4) (1992) 301–316, doi:10.1109/69.149926.
[112] K.A. Spackman, Signal detection theory: Valuable tools for evaluating inductive learning, 1989,. 6th Int. Workshop on Machine Learning, Morgan Kaufmann. 160–163
[113] F. Ranzato, M. Zanella, Genetic adversarial training of decision trees, 2020,. arXiv:2012.11352.
[114] M. Czajkowski, K. Jurczuk, M.K. etowski, A Parallel Approach for Evolutionary Induced Decision Trees. MPI + openMP Implementation, in: L. Rutkowski (Ed.), (ICAISC 2015), Vol. 9119 of LNCS, Springer, Zakopane, Poland, 2015, pp. 340–349, doi:10.1007/978-3-319-19324-3_31.
[115] S. Arlot, A. Celisse, A survey of cross-validation procedures for model selection, Stat Surv 4 (2010) 40–79, doi:10.1214/09-SS054.
[116] Y. Zhang, Y. Yang, Cross-validation for selecting a model selection procedure, J Econom 187 (1) (2015) 95–112, doi:10.1016/j.jeconom.2015.02.006.
[117] M. Stone, Cross-validatory choice and assessment of statistical predictions, journal of the royal statistical society, Series B (Methodological) (1974) 111–147.
[118] T.G. Dietterich, Approximate statistical tests for comparing supervised classification learning algorithms, Neural Comput 10 (7) (1998) 1895–1923.
[119] B. Efron, Bootstrap methods: another look at the jackknife, Ann. Statist. 7 (1) (1979) 1–26, doi:10.1214/aos/1176344552.
[120] R. Fisher, Statistical methods and scientific inference, Hafner Publishing Co., 1956.
[121] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, J Am Stat Assoc 32 (200) (1937) 675–701, doi:10.1080/01621459.1937.10503522.
[122] J. Luengo, S. García, F. Herrera, A study on the use of statistical tests for experimentation with neural networks: analysis of parametric test conditions and non-parametric tests, Expert Syst Appl 36 (4) (2009) 7798–7808, doi:10.1016/j.eswa.2008.11.041.
[123] B. Bergmann, G. Hommel, Improvements of General Multiple Test Procedures for Redundant Systems of Hypotheses, in: Multiple Hypotheses Testing, Springer, 1988, pp. 100–115, doi:10.1007/978-3-642-52307-6_8.
[124] O. Dunn, Multiple comparisons among means, J Am Stat Assoc 56 (293) (1961) 52–64, doi:10.1080/01621459.1961.10482090.
[125] S. Holm, A simple sequentially rejective multiple test procedure, Scand. J. Stat. (1979) 65–70. https://www.jstor.org/stable/4615733
[126] G. Hommel, A stagewise rejective multiple test procedure based on a modified bonferroni test, Biometrika 75 (2) (1988) 383–386, doi:10.1093/biomet/75.2.383.
[127] R. Iman, J. Davenport, Approximations of the critical region of the friedman statistic, Communications in Statistics-Theory and Methods 9 (6) (1980) 571–595, doi:10.1080/03610928008827904.

[128] W. Kruskal, W. Wallis, Use of ranks in one-criterion variance analysis, J Am Stat Assoc 47 (260) (1952) 583–621, doi:10.1080/01621459.1952.10483441.
[129] H.B. Mann, D.R. Whitney, On a test of whether one of two random variables is stochastically larger than the other, The Annals of Mathematical Statistics 18 (1) (1947) 50–60, doi:10.1214/aoms/1177730491.
[130] P. Nemenyi, Distribution-free multiple comparisons, Biometrics 18 (2) (1962) 263.
[131] J. Tukey, Comparing individual means in the analysis of variance, Biometrics (1949) 99–114, doi:10.2307/3001913.
[132] J. Shaffer, Modified sequentially rejective multiple test procedures, J Am Stat Assoc 81 (395) (1986) 826–831, doi:10.1080/01621459.1986.10478341.
[133] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics Bulletin 1 (6) (1945) 80–83, doi:10.2307/3001968.
[134] S.K. Murthy, S. Kasif, S. Salzberg, R. Beigel, OC1: A Randomized Algorithm for Building Oblique Decision Trees, in: AAAI'93, volume 93, AAAI press, 1993, pp. 322–327.
[135] S.K. Murthy, S. Kasif, S. Salzberg, A system for induction of oblique decision trees, Journal of Artificial Intelligence Research 2 (1) (1994) 1–32, doi:10.1613/jair.63.
[136] J.F. Lutsko, B. Kuijpers, Simulated Annealing in the Construction of Near-optimal Decision Trees, in: P. Cheeseman (Ed.), Selecting Models from Data, Vol. 89 of LNCS, Springer, 1994, pp. 453–462, doi:10.1007/978-1-4612-2660-4_46.
[137] E. Cantú-Paz, C. Kamath, Inducing oblique decision trees with evolutionary algorithms, IEEE Trans. on Evolutionary Computation 7 (1) (2003) 54–68, doi:10.1109/TEVC.2002.806857.
[138] K.P. Bennett, J.A. Blue, N.Y. Troy, An Extreme Point Tabu Search Method for Data Mining, Tech. rep., Rensselaer Polytechnic Institute, 1996.
[139] X.B. Li, J.R. Sweigart, J.T.C. Teng, J.M. Donohue, L. Thombs, S.M. Wang, Multivariate decision trees using linear discriminants and tabu search, IEEE Trans. on Systems, Man, and Cybernetics – Part A: Systems and Humans 33 (2) (2003) 194–205, doi:10.1109/TSMCA.2002.806499.
[140] R.S. Bucy, R.S. Diesposti, Decision tree design by simulated annealing, ESAIM: Mathematical Modelling and Numerical Analysis 27 (5) (1993) 515–534.
[141] C. Sutton, E. Keramidas, et al., Improving classification trees with simulated annealing, 1991,. 23th Interface Symp.: Computing Science and Statistics, Interface Fundation of North America. 396–402
[142] J. Pacheco, E. Alfaro, S. Casado, M. Gámez, N. García, A GRASP method for building classification trees, Expert Syst Appl 39 (3) (2012) 3241–3248, doi:10.1016/j.eswa.2011.09.011.
[143] M.G.V. Boas, H.G. Santos, L.H. de Campos Merschmann, Optimal decision trees for feature based parameter tuning: integer programming model and VNS heuristic, Electronic Notes in Discrete Mathematics 66 (2018) 223–230, doi:10.1016/j.endm.2018.03.029.
[144] K.P. Bennett, Decision Tree Construction via Linear Programming, Tech. Rep., Center for Parallel Optimization, Computer Sciences Department, 1992. University of Wisconsin
[145] J. Dvořák, P. Savický, Softening splits in decision trees using simulated annealing, Springer, 2007, pp. 721–729, doi:10.1007/978-3-540-71618-1_80. ICANNGA 2007, Part I, Vol. 4431 of LNCS
[146] J. Gama, P. Brazdil, Linear tree, Intell. Data Anal. 3 (1) (1999) 1–22, doi:10.1016/S1088-467X(99)00002-5.
[147] M.R. Garey, Optimal Binary Decision Trees for Diagnostic Identification Problems, 1970 Ph.d. thesis. The University of Wisconsin
[148] S.B. Gelfand, C.S. Ravishankar, E.I. Delp, An Iterative Growing and Pruning Algorithm for Classification Tree Design, in: Int. Conf. on Systems, Man and Cybernetics, IEEE, 1989, pp. 818–823, doi:10.1109/ICSMC.1989.71407.
[149] C. Atkeson, A. Moore, S. Schaal, Locally weighted learning, 1996,. Artif Intell Rev, 11, 11–7310.1023/A:1006559212014.
[150] S.K. Shevade, S.S. Keerthi, C. Bhattacharyya, K.R.K. Murthy, Improvements to the SMO algorithm for SVM regression, IEEE Trans. on Neural Networks 11 (5) (2000) 1188–1193, doi:10.1109/72.870050.
[151] M. Frank, P. Wolfe, An algorithm for quadratic programming, Naval Research Logistics Quarterly 3 (1–2) (1956) 95–110, doi:10.1002/nav.3800030109.
[152] J. Koza, Genetic programming: on the programming of computers by means of natural selection, MIT Press, 1992.
[153] M.A. Adibi, Single and multiple outputs decision tree classification using bi-level discrete-continues genetic algorithm, Pattern Recognit. Letters 128 (2019) 190–196, doi:10.1016/j.patrec.2019.09.001.
[154] B.B. Chai, T. Huang, X. Zhuang, Y. Zhao, J. Sklansky, Piecewise linear classifiers using binary tree structure and genetic algorithm, Pattern Recognit 29 (11) (1996) 1905–1917, doi:10.1016/0031-3203(96)00019-2.
[155] A. Omielan, S. Vadera, ECCO: A New Evolutionary Classifier with Cost Optimisation, in: Z. Shi (Ed.), IIP 2012, Vol. 385 of IFIP ICT, Springer, Guilin, China, 2012, pp. 97–105, doi:10.1007/978-3-642-32891-6_14.
[156] B. Vukobratovic, R. Struharik, Evolving Full Oblique Decision Trees, in: CINTI 2015, IEEE, 2015, pp. 95–100, doi:10.1109/CINTI.2015.7382901.
[157] D. Jankowski, K. Jackowski, Evolutionary Algorithm for Decision Tree Induction, in: K. Saeed (Ed.), CISIM 2014, Vol. 8838 of LNCS, Springer, Ho Chi Minh City, Vietnam, 2014, pp. 23–32, doi:10.1007/978-3-662-45237-0_4.
[158] S.B. Yang, Fuzzy variable-branch decision tree, J Electron Imaging 19 (4) (2010), doi:10.1117/1.3504357. 043012–043012–9
[159] W. Pedrycz, Z.A. Sosnowski, Genetically optimized fuzzy decision trees, IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics 35 (3) (2005) 633–641, doi:10.1109/TSMCB.2005.843975.
[160] J.Y. Chang, C.W. Cho, S.H. Hsieh, S.T. Chen, Genetic algorithm based fuzzy ID3 algorithm, Springer, 2004, doi:10.1007/978-3-540-30499-9_153. ICONIP 2004, Vol. 3316 of LNCS
[161] S.C. Ng, K.S. Leung, Induction of Quadratic Decision Trees Using Genetic Algorithms, in: 2003 Intelligent Automation Conf., 2003, pp. 979–984.
[162] X. Llorà, J.M. Garrell, Evolution of Decision Trees, in: CCIA'2001, ACIA, Press, 2001, pp. 115–122.
[163] A. Papagelis, D. Kalles, GA Tree: Genetically Evolved Decision Trees, in: ICTAI 2000, IEEE, 2000, pp. 203–2006, doi:10.1109/TAI.2000.889871.
[164] M.K. etowski, A memetic algorithm for global induction of decision trees, Springer, 2008, pp. 531–540, doi:10.1007/978-3-540-77566-9_46. SOFSEM 2008, Vol. 4910 of LNCS
[165] M.K. etowski, M. Grześ, Evolutionary induction of decision trees for misclassification cost minimization, Springer, 2007, pp. 1–10, doi:10.1007/978-3-540-71618-1_1. ICANNGA 2007, Part I, Vol. 4431 of LNCS
[166] M.K. etowski, M. Grześ, Mixed decision trees: An evolutionary approach, Springer, Krakow, Poland, 2006, pp. 260–269, doi:10.1007/11823728_25. DaWaK 2006, Vol. 4081 of LNCS
[167] D. Dumitrescu, J. András, Generalized decision trees built with evolutionary techniques, Studies in Informatics and Control 14 (1) (2005) 15–22.
[168] V. Podgorelec, P. Kokol, Evolutionary Construction of Medical Decision Trees, in: H.K. Chang (Ed.), IEEE-EMBS, volume 3, IEEE, 1998, pp. 1202–1205, doi:10.1109/IEMBS.1998.747088.
[169] R. Struharik, V. Vranjkovic, S. Dautovic, L. Novak, Inducing Oblique Decision Trees, in: SISY–2014, IEEE, Subotica, Serbia, 2014, pp. 257–262, doi:10.1109/SISY.2014.6923596.
[170] J. Sanz, H. Bustince, A. Fernández, F. Herrera, IIVFDT: Ignorance functions based interval-valued fuzzy decision tree with genetic tuning, Int. Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 20 (2012) 1–30, doi:10.1142/S0218488512400132. Supp02
[171] J. András, D. Dumitrescu, Evolving orthogonal decision trees, Studia Universitatis Babes-Bolyai. Series Informatica 48 (2) (2003) 33–44.
[172] Y. Dhebar, K. Deb, Interpretable rule discovery through bilevel optimization of split-rules of nonlinear decision trees for classification problems, IEEE Trans Cybern (2020) 1–12, doi:10.1109/TCYB.2020.3033003.
[173] S.C. Ng, K.S. Leung, Induction of Linear Decision Trees with Real-coded Genetic Algorithms and K-d Trees, in: M. Gallagher (Ed.), IDEAL, Springer, Berlin, Heidelberg, 2005, pp. 264–271, doi:10.1007/11508069_35.
[174] J.B. Gray, G. Fan, Classification tree analysis using TARGET, Computational Statistics & Data Analysis 52 (3) (2008) 1362–1372, doi:10.1016/j.csda.2007.03.014.
[175] S.H. Cha, C. Tappert, H.R. Arabnia, Constructing binary decision trees using genetic algorithms, 2008,. GEM 2008, CSREA, Las Vegas, Nevada, USA. 49–54
[176] S.H. Cha, C. Tappert, A genetic algorithm for constructing compact binary decision trees, Journal of Pattern Recognition Research 4 (1) (2009) 1–13, doi:10.13176/11.44.
[177] S.F. Smith, RNA Search Acceleration with Genetic Algorithm Generated Decision Trees, in: ICMLA'08, IEEE, San Diego, CA, USA, 2008, pp. 565–570, doi:10.1109/ICMLA.2008.77.
[178] E. Ersoy, E. Albey, E. Kayiş, S. Hammoudi, A CART-based genetic algorithm for constructing higher accuracy decision trees, 2020,. DATA 2020, SCITEPRESS. 328–338, 10.5220/0009893903280338
[179] M. Oltean, D. Dumitrescu, Multi expression programming, tech. rep. UBB-01-2002, 2002,. Babes-Bolyai University, Cluj-Napoca, Romania.
[180] J.M. Pangilinan, G.K. Janssens, Pareto-optimality of oblique decision trees from evolutionary algorithms, J. Global Optim. 51 (2) (2011) 301–311, doi:10.1007/s10898-010-9614-9.
[181] D. Levi, Hereboy: A Fast Evolutionary Algorithm, in: J. Lohn (Ed.), EH'00, IEEE, 2000, pp. 17–24, doi:10.1109/EH.2000.869338.
[182] X. Llorà, S.W. Wilson, K. Deb, et al., Mixed Decision Trees: Minimizing Knowledge Representation Bias in LCS, in: GECCO'04, Vol. 3103 of LNCS, Springer, 2004, pp. 797–809, doi:10.1007/978-3-540-24855-2_94.
[183] S.C. Ng, K.S. Leung, Induction of quadratic decision trees using genetic algorithms and k-d trees, WSEAS Trans. on Computers 3 (3) (2004) 839–845.
[184] C.Z. Janikow, A genetic algorithm method for optimizing fuzzy decision trees, Inf Sci (Ny) 89 (3) (1996) 275–296, doi:10.1016/0020-0255(95)00239-1.
[185] K.A. Crockett, Z. Bandar, A. Al-Attar, Optimising Decision Classifications Using Genetic Algorithms, in: A. Dobnikar (Ed.), Artificial Neural Nets and Genetic Algorithms, Springer, 1999, pp. 191–195, doi:10.1007/978-3-7091-6384-9_33.
[186] H. Bustince, M. Pagola, E. Barrenechea, J. Fernandez, P. Melo-Pinto, P. Couto, H.R. Tizhoosh, J. Montero, Ignorance functions. an application to the calculation of the threshold in prostate ultrasound images, Fuzzy Sets Syst. 161 (1) (2010) 20–36, doi:10.1016/j.fss.2009.03.005.
[187] Y. Yuan, M.J. Shaw, Induction of fuzzy decision trees, Fuzzy Sets Syst. 69 (2) (1995) 125–139, doi:10.1016/0165-0114(94)00229-Z.
[188] M.W. Kim, J.W. Ryu, L. Wang, et al., Optimized fuzzy classification using genetic algorithm, Springer, 2005, pp. 392–401, doi:10.1007/11539506_51. FSKD 2005, Vol. 3613 of LNAI
[189] J. Chen, X. Wang, J. Zhai, Pruning Decision Tree Using Genetic Algorithms, in: AICI'09, volume 3, IEEE, 2009, pp. 244–248, doi:10.1109/AICI.2009.351.
[190] A. Brunello, E. Marzano, A. Montanari, G. Sciavicco, Decision tree pruning via multi-objective evolutionary computation, Int. J. Mach. Learn. Comput. 7 (6) (2017) 167–175, doi:10.18178/ijmlc.2017.7.6.641.
[191] J. Grefenstette, GENESIS: A system for using genetic search procedures, Proc. of a Conf. on Intelligent Systems and Machines (1984) 161–165.
[192] Z. Michalewicz, C.Z. Janikow, GENOCOP: A genetic algorithm for numerical optimization problems with linear constraints, Commun ACM 39 (1996) 175, doi:10.1145/272682.272711. 12es
[193] M. Wall, GAlib: A C++ Library of Genetic Algorithm Components, Mas-

sachusetts Institute of Technology, Mechanical Engineering Department, 2016. http://lancet.mit.edu/ga
[194] H. Hotelling, Relations between two sets of variates, Biometrika 28 (3/4) (1936) 321–377.
[195] R. Fisher, The use of multiple measurements in taxonomic problems, Ann Eugen 7 (2) (1936) 179–188, doi:10.1111/j.1469-1809.1936.tb02137.x.
[196] P.E. Utgoff, C.E. Brodley, Linear machine decision trees, tech. rep, 1991,. University of Massachusetts,Amherst, MA, USA.
[197] A. Ittner, M. Schlosser, Non-linear Decision Trees-NDT, in: L. Saitta (Ed.), ICML'96, Morgan Kaufmann, 1996, pp. 252–257.
[198] N.R. Pal, S. Chakraborty, A. Bagchi, RID3: An ID3-like algorithm for real data, Inf Sci (Ny) 96 (3–4) (1997) 271–290, doi:10.1016/S0020-0255(96)00162-4.
[199] W. Pedrycz, Z.A. Sosnowski, C-Fuzzy decision trees, IEEE trans. on systems, man, and cybernetics, Part C: Applications and Reviews 35 (4) (2005) 498–511, doi:10.1109/TSMCC.2004.843205.
[200] X. Wang, B. Chen, G. Qian, F. Ye, On the optimization of fuzzy decision trees, Fuzzy Sets Syst. 112 (1) (2000) 117–125, doi:10.1016/S0165-0114(97)00386-2.
[201] D.S. Yeung, X.Z. Wang, E.C.C. Tsang, Learning Weighted Fuzzy Rules from Examples with Mixed Attributes by Fuzzy Decision Trees, in: SMC'99, volume 3, IEEE, 1999, pp. 349–354, doi:10.1109/ICSMC.1999.823229.
[202] P.D. Turney, Cost-sensitive classification: empirical evaluation of a hybrid genetic decision tree induction algorithm, Journal of Artificial Intelligence 2 (1) (1995) 369–409, doi:10.1613/jair.120.
[203] J. Abonyi, J.A. Roubos, F. Szeifert, Data-driven generation of compact, accurate, and linguistically sound fuzzy classifiers based on a decision-tree initialization, Int. Journal of Approximate Reasoning 32 (1) (2003) 1–21, doi:10.1016/S0888-613X(02)00076-2.
[204] Z. Fu, B.L. Golden, S. Lele, S. Raghavan, E.A. Wasil, Genetically engineered decision trees: population diversity produces smarter trees, Oper Res 51 (6) (2003) 894–907, doi:10.1287/opre.51.6.894.24919.
[205] M.P. Basgalupp, R.C. Barros, A.C.P.L.F. de Carvalho, A.A. Freitas, Evolving decision trees with beam search-based initialization and lexicographic multi-objective evaluation, Inf Sci (Ny) 258 (2014) 160–181, doi:10.1016/j.ins.2013.07.025.
[206] D. Kalles, A. Papagelis, Lossless fitness inheritance in genetic algorithms for decision trees, Soft comput 14 (9) (2010) 973–993, doi:10.1007/s00500-009-0489-y.
[207] K. Jurczuk, M. Czajkowski, M.K. etowski, Evolutionary induction of a decision tree for large-scale data: a GPU-based approach, Soft comput 21 (24) (2017) 7363–7379, doi:10.1007/s00500-016-2280-1.
[208] K. Jurczuk, M. Czajkowski, M.K. etowski, Fitness evaluation reuse for accelerating GPU-based evolutionary induction of decision trees, The Int. Journal of High Performance Computing Applications 35 (1) (2021) 20–32, doi:10.1177/1094342020957393.
[209] K. Jurczuk, M. Czajkowski, M.K. etowski, Multi-GPU approach to global induction of classification trees for large-scale data mining, 2021b, Applied Intelligence, 1–18, 10.1007/s10489-020-01952-5
[210] L. Bosnjak, S. Karakatic, V. Podgorelec, Using Similarity-based Selection in Evolutionary Design of Decision Trees, in: MIPRO 2015, IEEE, 2015, pp. 1206–1211, doi:10.1109/MIPRO.2015.7160459.
[211] Z. Fu, F. Mae, A Computational Study of Using Genetic Algorithms to Develop Intelligent Decision Trees, in: CEC-2001, volume 2, IEEE, 2001, pp. 1382–1387, doi:10.1109/CEC.2001.934352.
[212] Z. Fu, B.L. Golden, S. Lele, S. Raghavan, E.A. Wasil, Building a High-quality Decision Tree with a Genetic Algorithm, in: M. Laguna (Ed.), Computing Tools for Modeling, Optimization and Simulation, Springer, 2000, pp. 25–38, doi:10.1007/978-1-4615-4567-5_2. Vol. 12 of OR/CS Interfaces
[213] Z. Fu, B.L. Golden, S. Lele, S. Raghavan, E.A. Wasil, A genetic algorithm-based approach for building accurate decision trees, INFORMS J Comput 15 (1) (2003) 3–22.
[214] Z. Fu, B.L. Golden, S. Lele, S. Raghavan, E.A. Wasil, Diversification for better classification trees, Computers & Operations Research 33 (11) (2006) 3185–3202, doi:10.1016/j.cor.2005.02.035.
[215] R. Biedrzycki, J. Arabas, Evolutionary and greedy exploration of the space of decision trees, Evolutionary Computation and Global Optimization (2006) 479–489.
[216] S. Rzheutskaya, A. Rzheutskiy, R.R. Salikhova, Applying a Genetic Algorithm to Build a Classification Tree, in: DEFIN'20, ACM, NY, 2020, pp. 1–4, doi:10.1145/3388984.3390878. USA
[217] M.K. etowski, M. Grześ, Global Induction of Oblique Decision Trees: An Evolutionary Approach, in: M.A. Kłopotek (Ed.), IIPWM'05, volume 31, Springer, 2005, pp. 309–318, doi:10.1007/3-540-32392-9_32. Of ASC
[218] M.K. etowski, M. Grześ, Evolutionary Learning of Linear Trees with Embedded Feature Selection, in: L. Rutkowski (Ed.), ICAISC 2006, Vol. 4029 of LNAI, Springer, 2006, pp. 400–409, doi:10.1007/11785231_43.
[219] M.K. etowski, P. Popczyński, Global Induction of Decision Trees: From Parallel Implementation to Distributed Evolution, in: L. Rutkowski (Ed.), (ICAISC 2008), Vol. 5097 of LNCS, Springer, Zakopane, Poland, 2008, pp. 426–437, doi:10.1007/978-3-540-69731-2_42.
[220] D. Reska, K. Jurczuk, M.K. etowski, Evolutionary induction of classification trees on spark, Springer, Zakopane, Poland, 2018, pp. 514–523, doi:10.1007/978-3-319-91253-0_48. (ICAISC 2018), Vol. 10841 of LNCS
[221] R.C. Holte, Very simple classification rules perform well on most commonly used datasets, Mach Learn 11 (1) (1993) 63–90, doi:10.1023/A:1022631118932.
[222] P. Domingos, Metacost: A General Method for Making Classifiers Cost-sensitive, in: KDD'99, ACM, 1999, pp. 155–164, doi:10.1145/312129.312220.
[223] H.A. Chipman, E.I. George, R.E. McCulloch, Bayesian CART model search, J Am Stat Assoc 93 (443) (1998) 935–948. .1080/01621459.1998.10473750
[224] R. Tibshirani, K. Knight, Model search by bootstrap "bumping", Journal of Computational and Graphical Statistics 8 (4) (1999) 671–686, doi:10.1080/10618600.1999.10474842.
[225] S.E. Rouwhorst, A.P. Engelbrecht, Searching the Forest: Using Decision Trees as Building Blocks for Evolutionary Search in Classification Databases, in: CEC-2000, 1, IEEE, 2000, pp. 633–638, doi:10.1109/TSMCC.2004.843247.
[226] G. Folino, C. Pizzuti, G. Spezzano, A cellular genetic programming approach to classification, 1999,. GECCO-99, Morgan Kaufmann. W. Banzhaf, 1015–1020, 10.5555/2934046.2934058
[227] J. Li, X. Li, X. Yao, Cost-sensitive Classification with Genetic Programming, in: CEC-2005, volume 3, IEEE, 2005, pp. 2114–2121, doi:10.1109/CEC.2005.1554956.
[228] P. Wang, M. Emmerich, R. Li, K. Tang, T. Bäck, X. Yao, Convex hull-based multiobjective genetic programming for maximizing receiver operating characteristic performance, IEEE Trans. on Evolutionary Computation 19 (2) (2014) 188–200, doi:10.1109/TEVC.2014.2305671.
[229] R. König, U. Johansson, T. Löfström, L. Niklasson, Improving GP Classification Performance by Injection of Decision Trees, in: CEC-2010, IEEE, 2010, pp. 1–8, doi:10.1109/CEC.2010.5585988.
[230] E.P.K. Tsang, J. Li, J.M. Butler, EDDIE Beats the bookies, Software: Practice and Experience 28 (10) (1998) 1033–1043, doi:10.1002/(SICI)1097-024X(199808)28:10<1033::AID-SPE198>3.0.CO;2-1.
[231] P. Wang, T. Weise, R. Chiong, Novel evolutionary algorithms for supervised classification problems: an experimental study, Evol Intell 4 (1) (2011) 3–16, doi:10.1007/s12065-010-0047-7.
[232] D.E. Kim, Structural Risk Minimization on Decision Trees Using an Evolutionary Multiobjective Optimization, in: M. Keijzer (Ed.), EuroGP 2004, Springer, 2004, pp. 338–348, doi:10.1007/978-3-540-24650-3_32. Vol. 3003 of LNCS
[233] R.K. DeLisle, S.L. Dixon, Induction of decision trees via evolutionary programming, J Chem Inf Comput Sci 44 (3) (2004) 862–870, doi:10.1021/ci034188s.
[234] J. Li, FGP: A Genetic Programming Based Financial Forecasting Tool, 2000 Ph.d. thesis. University of Essex
[235] J. Eggermont, Evolving fuzzy decision trees with genetic programming and clustering, Springer, 2002, pp. 71–82, doi:10.1007/3-540-45984-7_7. EuroGP 2002, Vol. 2278 of LNCS
[236] A. Tsakonas, G. Dounias, Hierarchical Classification Trees Using Type-constrained Genetic Programming, in: 1st Int. IEEE Symposium Intelligent Systems, volume 2, IEEE, 2002, pp. 50–54, doi:10.1109/IS.2002.1042573.
[237] R.E. Marmelstein, G.B. Lamont, J.Y. Koza, Pattern classification using a hybrid genetic program decision tree approach, 1998,. GP-98, Morgan Kaufmann. 223–231
[238] N.I. Nikolaev, V. Slavov, Inductive Genetic Programming with Decision Trees, in: M. van Someren (Ed.), ECML-97 Part II, Vol. 1224 of LNCS, Springer, 1997, pp. 183–190, doi:10.1007/3-540-62858-4_83.
[239] J.K. Estrada-Gil, J.C. Fernández-López, E. Hernández-Lemus, I. Silva-Zolezzi, A. Hidalgo-Miranda, G. Jiménez-Sánchez, E. Vallejo-Clemente, GPDTI: A genetic programming decision tree induction method to find epistatic effects in common complex diseases, Bioinformatics 23 (13) (2007), doi:10.1093/bioinformatics/btm205. I167–i174
[240] E. Dufourq, N. Pillay, Incorporating Adaptive Discretization into Genetic Programming for Data Classification, in: WICT-2013, IEEE, 2013, pp. 127–133, doi:10.1109/WICT.2013.7113123.
[241] P. Wang, K. Tang, E.P.K. Tsang, X. Yao, A Memetic Genetic Programming with Decision Tree-based Local Search for Classification Problems, in: CEC-2011, IEEE, 2011, pp. 917–924, doi:10.1109/CEC.2011.5949716.
[242] L. Yi, K. Wanli, A new genetic programming algorithm for building decision tree, Procedia Eng 15 (2011) 3658–3662, doi:10.1016/j.proeng.2011.08.685.
[243] P. Wang, K. Tang, T. Weise, E.P.K. Tsang, X. Yao, Multiobjective genetic programming for maximizing ROC performance, Neurocomputing 125 (2014) 102–118, doi:10.1016/j.neucom.2012.06.054.
[244] S. Casjens, H. Schwender, T. Brüning, K. Ickstadt, A novel crossover operator based on variable importance for evolutionary multi-objective optimization with tree representation, J. Heuristics 21 (1) (2015) 1–24, doi:10.1007/s10732-014-9269-7.
[245] M. Chabbouh, S. Bechikh, C.C. Hung, L.B. Said, Multi-objective evolution of oblique decision trees for imbalanced data binary classification, Swarm Evol Comput 49 (2019) 1–22, doi:10.1016/j.swevo.2019.05.005.
[246] E.M. Mugambi, A. Hunter, G. Oatley, L. Kennedy, Polynomial-fuzzy decision tree structures for classifying medical data, Knowl Based Syst 17 (2–4) (2004) 81–87, doi:10.1016/j.knosys.2004.03.003.
[247] H. Iba, H. de Garis, T. Sato, Genetic Programming Using a Minimum Description Length Principle, in: K.E. Kinnear (Ed.), Advances in Genetic Programming, MIT Press, 1994, pp. 265–284.
[248] D.E. Kim, Minimizing Structural Risk on Decision Tree Classification, in: J. Yaouchu (Ed.), Multi-Objective Machine Learning, Vol. 16 of SCI, Springer, 2006, pp. 241–260, doi:10.1007/3-540-33019-4_11.
[249] A. Niimi, E. Tazaki, Object Oriented Approach to Combined Learning of Decision Tree and Adf Gp, in: IJCNN'99, volume 6, IEEE, 1999, pp. 4166–4170, doi:10.1109/IJCNN.1999.830832.
[250] A. Niimi, E. Tazaki, Genetic Programming Combined with Association Rule Algorithm for Decision Tree Construction, in: KES 2000, volume 2, IEEE, 2000, pp. 746–749, doi:10.1109/KES.2000.884154.
[251] R. Agrawal, R. Srikant, Fast Algorithms for Mining Association Rules, in: J.B. Bocca (Ed.), VLDB 1994, volume 1215, 1994, pp. 487–499. Morgan Kaufmann
[252] M.D. Ryan, V.J. Rayward-Smith, J.R. Koza, The evolution of decision trees, 1998, GP-98, Morgan Kaufmann. 350–358
[253] G. Folino, C. Pizzuti, G. Spezzano, Scalable Classification of Large Data Sets by Parallel Genetic Programming, in: P. Kacsuk (Ed.), Distributed and parallel systems, Springer, Boston, 2000, pp. 87–90, doi:10.1007/978-1-4615-4489-0_11.

[254] C. To, T.D. Pham, Analysis of Cardiac Imaging Data Using Decision Tree Based Parallel Genetic Programming, in: ISPA 2009, IEEE, 2009, pp. 317–320, doi:10.1109/ISPA.2009.5297730.
[255] T.M. Khoshgoftaar, N. Seliya, Y. Liu, Genetic Programming-based Decision Trees for Software Quality Classification, in: ICTAI 2003, IEEE, 2003, pp. 374–383, doi:10.1109/TAI.2003.1250214.
[256] T.M. Khoshgoftaar, Y. Liu, A multi-objective software quality classification model using genetic programming, IEEE Trans. on Reliability 56 (2) (2007) 237–245, doi:10.1109/TR.2007.896763.
[257] C.S. Kuo, T.P. Hong, C.L. Chen, Applying genetic programming technique in classification trees, Soft comput 11 (12) (2007) 1165–1172, doi:10.1007/s00500-007-0159-x.
[258] H. Zhao, A multi-objective genetic programming approach to developing pareto optimal decision trees, Decis Support Syst 43 (3) (2007) 809–826, doi:10.1016/j.dss.2006.12.011.
[259] U. Johansson, L. Niklasson, Evolving Decision Trees Using Oracle Guides, in: CIDM'09, IEEE, 2009, pp. 238–244, doi:10.1109/CIDM.2009.4938655.
[260] U. Johansson, R. König, T. Löfström, L. Niklasson, Using Imaginary Ensembles to Select GP Classifiers, in: A.I. Esparcia-Alcázar (Ed.), EuroGP 2010, Springer, 2010, pp. 278–288, doi:10.1007/978-3-642-12148-7_24.
[261] M.C.J. Bot, W.B. Langdon, Improving Induction of Linear Classification Trees with Genetic Programming, in: L.D. Whitley (Ed.), GECCO-00, 2000, pp. 403–410, doi:10.5555/2933718.2933796. Morgan Kaufmann
[262] M. Šprogar, Prudent alignment and crossover of decision trees in genetic programming, Genetic Programming and Evolvable Machines 16 (4) (2015) 499–530, doi:10.1007/s10710-015-9243-7.
[263] M. Shirasaka, Q. Zhao, O. Hammami, K. Kuroda, K. Saito, Automatic Design of Binary Decision Trees Based on Genetic Programming, in: SEAL'98, 1998, pp. 1–8.
[264] Q. Zhao, M. Shirasaka, A Study on Evolutionary Design of Binary Decision Trees, in: CEC-1999, volume 3, IEEE, 1999, pp. 1988–1993, doi:10.1109/CEC.1999.785518.
[265] T. Tanigawa, Q. Zhao, A study on efficient generation of decision trees using genetic programming, 2000, GECCO-00, Morgan Kaufmann. D. Whitley, 1047–1052, 10.5555/2933718.2933915
[266] S. Haruyama, Q. Zhao, A. El-Kamel, Designing smaller decision trees using multiple objective optimization based GPs, Int. Conf. on Systems, Man and Cybernetics 6 (2002), doi:10.1109/ICSMC.2002.1175597. 5–pp
[267] F.V. Buontempo, X.Z. Wang, M. Mwense, N. Horan, A. Young, D. Osborn, Genetic programming for the induction of decision trees to model ecotoxicity data, J Chem Inf Model 45 (4) (2005) 904–912, doi:10.1021/ci049652n.
[268] X.Z. Wang, F.V. Buontempo, A. Young, D. Osborn, Induction of decision trees using genetic programming for modelling ecotoxicity data: adaptive discretization of real-valued endpoints, SAR QSAR Environ Res 17 (5) (2006) 451–471, doi:10.1080/10629360600933723.
[269] A.P. Engelbrecht, S. Rouwhorst, L. Schoeman, A building block approach to genetic programming for rule discovery, Data Mining: A Heuristic Approach (2001) 174–189, doi:10.4018/978-1-930708-25-9.ch009.
[270] J. Eggermont, J.N. Kok, W.A. Kosters, T. Heskes, Genetic programming for data classification: Refining the search space, 2003,. BNAIC'03, University of Nijmegen. 123–130
[271] J. Eggermont, J.N. Kok, W.A. Kosters, Genetic Programming for Data Classification: Partitioning the Search Space, in: SAC'04, ACM, 2004, pp. 1001–1005, doi:10.1145/967900.968104.
[272] E. Dufourq, N. Pillay, A Preliminary Study on the Reuse of Subtrees within Decision Trees in a Genetic Programming Context for Data Classification, in: WICT-2013, IEEE, 2013, pp. 285–290, doi:10.1109/WICT.2013.7113150.
[273] S. Karakatič, V. Podgorelec, Heuristic Crossover Operator for Evolutionary Induced Decision Trees, in: Int. Conf. on Evolutionary Computation Theory and Applications, volume 2, SciTePress, 2014, pp. 289–293, doi:10.5220/0005137102890293.
[274] S. Karakatič, M. Heričko, V. Podgorelec, Improving Genetic Programming for Classification with Lazy Evaluation and Dynamic Weighting, in: C. Sabourin (Ed.), IJCCI 2017, Springer, Cham, 2019, pp. 63–75, doi:10.1007/978-3-030-16469-0_4.
[275] M. Saremi, F. Yaghmaee, Evolutionary Decision Tree Induction with Multi-interval Discretization, in: ICIS 2014, IEEE, 2014, pp. 1–6, doi:10.1109/Iranian-CIS.2014.6802543.
[276] M. Saremi, F. Yaghmaee, Improving evolutionary decision tree induction with multi-interval discretization, Comput Intell 34 (2) (2018) 495–514, doi:10.1111/coin.12153.
[277] A. Shali, M.R. Kangavari, B. Bina, M. Arif-Wani, Using genetic programming for the induction of oblique decision trees, IEEE, 2007, pp. 38–43, doi:10.1109/ICMLA.2007.66.
[278] A. Tsakonas, A comparison of classification accuracy of four genetic programming-evolved intelligent structures, Inf Sci (Ny) 176 (6) (2006) 691–724, doi:10.1016/j.ins.2005.03.012.
[279] E.M. Mugambi, A. Hunter, Multi-objective Genetic Programming Optimization of Decision Trees for Classifying Medical Data, in: V. Palade (Ed.), KES 2003, Vol. 2773 of LNCS, Springer, 2003, pp. 293–299, doi:10.1007/978-3-540-45224-9_42.
[280] M. Ritchie, B. White, J. Parker, L. Hahn, J. Moore, Optimizationof neural network architecture using genetic programming improvesdetection and modeling of gene-gene interactions in studies of humandiseases, BMC Bioinformatics 4 (1) (2003) 28, doi:10.1186/1471-2105-4-28.
[281] U.M. Fayyad, K.B. Irani, On the handling of continuous-valued attributes in decision tree generation, Mach Learn 8 (1) (1992) 87–102, doi:10.1007/BF00994007.
[282] E. Frank, I.H. Witten, J.W. Shavlik, Generating accurate rule sets without global optimization, 1998, ICML'98, Morgan Kaufmann. 144–151,
[283] T. Fawcett, PRIE: A system for generating rulelists to maximize ROC performance, Data Min Knowl Discov 17 (2) (2008) 207–224, doi:10.1007/s10618-008-0089-y.
[284] V. Podgorelec, S. Karakatic, A multi-population genetic algorithm for inducing balanced decision trees on telecommunications churn data, Elektronika ir Elektrotechnika 19 (6) (2013) 121–124, doi:10.5755/j01.eee.19.6.4578.
[285] V. Podgorelec, S. Karakatic, R.C. Barros, M.P. Basgalupp, Evolving balanced decision trees with a multi-population genetic algorithm, IEEE, 2015, pp. 54–61, doi:10.1109/CEC.2015.7256874.
[286] V. Podgorelec, P. Kokol, Self-adaptation of Evolutionary Constructed Decision Trees by Information Spreading, in: A. Dobnikar (Ed.), Artificial Neural Nets and Genetic Algorithms, Springer, Vienna, 1999, pp. 294–301, doi:10.1007/978-3-7091-6384-9_49.
[287] C. Jariyavajee, J. Polvichai, B. Sirinaovakul, Searching for Splitting Criteria in Multivariate Decision Tree Using Adapted JADE Optimization Algorithm, in: SSCI 2019, 2019, pp. 2534–2540, doi:10.1109/SSCI44817.2019.9003063.
[288] R. Rivera-Lopez, J. Canul-Reich, A global search approach for inducing oblique decision trees using differential evolution, Springer, Edmonton, Canada, 2017, pp. 27–38, doi:10.1007/978-3-319-57351-9_3. AI 2017, Vol. 10233 of LNCS
[289] R. Rivera-Lopez, J. Canul-Reich, J.A. Gámez, J.M. Puerta, OC1-DE: A differential evolution based approach for inducing oblique decision trees, Springer, Zakopane, Poland, 2017, pp. 427–438, doi:10.1007/978-3-319-59063-9_38. ICAISC 2017, Vol. 10245 of LNCS
[290] V. Estivill-Castro, E. Gilmore, R. Hexel, Constructing Interpretable Decision Trees Using Parallel Coordinates, in: L. Rutkowski (Ed.), ICAISC 2020, Vol. 12416 of LNCS, Springer, Zakopane, Poland, 2020, pp. 152–164, doi:10.1007/978-3-030-61534-5_14.
[291] R.A. Lopes, A.R.R. Freitas, R.C.P. Silva, F.G. Guimarães, Differential Evolution and Perceptron Decision Trees for Classification Tasks, in: H. Yin, J.A.F. Costa, G. Barreto (Eds.), IDEAL 2012, Vol. 7435 of LNCS, Springer, Natal, Brazil, 2012, pp. 550–557, doi:10.1007/978-3-642-32639-4_67.
[292] K. Zhang, Z. Xu, B.P. Buckles, Oblique Decision Tree Induction Using Multimembered Evolution Strategies, in: B.V. Dasarathy (Ed.), Proceeding of Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security, SPIE 2005, volume 5812, SPIE, Orlando, Florida, 2005, pp. 263–270, doi:10.1117/12.596766.
[293] H.E.L. Cagnini, R.C. Barros, M.P. Basgalupp, Estimation of Distribution Algorithms for Decision-tree Induction, in: CEC-2017, IEEE, 2017, pp. 2022–2029, doi:10.1109/CEC.2017.7969549.
[294] L. Qu, Y. Min, W. Weihong, C. Xiaohong, Dynamic Split-point Selection Method for Decision Tree Evolved by Gene Expression Programming, in: CEC-2009, IEEE, Trondheim, Norway, 2009, pp. 736–740, doi:10.1109/CEC.2009.4983018.
[295] P.J. Pereira, P. Cortez, R. Mendes, Multi-objective grammatical evolution of decision trees for mobile marketing user conversion prediction, Expert Systems with Applications, 2020, 114287, 10.1016/j.eswa.2020.114287,
[296] G. Folino, C. Pizzuti, G. Spezzano, Genetic Programming and Simulated Annealing: A Hybrid Method to Evolve Decision Trees, in: R. Poli (Ed.), EuroGP 2000, Springer, Berlin, 2000, pp. 294–303, doi:10.1007/978-3-540-46239-2_22.
[297] A. Agapitos, M. O'Neill, A. Brabazon, T. Theodoridis, Maximum Margin Decision Surfaces for Increased Generalisation in Evolutionary Decision Tree Learning, in: S. Silva (Ed.), EuroGP 2011, Vol. 6621 of LNCS, Springer, 2011, pp. 61–72, doi:10.1007/978-3-642-20407-4_6.
[298] V. Podgorelec, P. Kokol, Towards more optimal medical diagnosing with evolutionary algorithms, J Med Syst 25 (3) (2001) 195–219, doi:10.1023/A:1010733016906.
[299] v.H. Babič, P. Kokol, V. Podgorelec, M. Zorman, .M. Šprogar, M.M. Štiglic, The art of building decision trees, J Med Syst 24 (1) (2000) 43–52, doi:10.1023/A:1005437213215.
[300] M. Zorman, V. Podgorelec, P. Kokol, M. Peterson, M. Šprogar, M. Ojsteršek, Finding the right decision tree's induction strategy for a hard real world problem, Int J Med Inform 63 (1) (2001) 109–121, doi:10.1016/S1386-5056(01)00176-9.
[301] M.J. Aitkenhead, A co-evolving decision tree classification method, Expert Syst Appl 34 (1) (2008) 18–25, doi:10.1016/j.eswa.2006.08.008.
[302] E. Dolotov, N. Zolotykh, Evolutionary Algorithms for Constructing an Ensemble of Decision Trees, in: W.M.P. van der Aalst (Ed.), AIST 2019, Vol. 1086 of CCIS, Springer, Cham, 2019, pp. 9–15, doi:10.1007/978-3-030-39575-9_2.
[303] S. Mitrofanov, E. Semenkin, Differential evolution in the decision tree learning algorithm, Siberian Journal of Science and Technology 20 (3) (2019), doi:10.31772/2587-6066-2019-20-3-312-319.
[304] A.A. Motsinger-Reif, S. Deodhar, S.J. Winham, N.E. Hardison, Grammatical evolution decision trees for detecting gene-gene interactions, BioData Min 3 (1) (2010) 1, doi:10.1186/1756-0381-3-8.
[305] R. Jiang, Gene-gene Interaction, in: M.D. Gellman (Ed.), Encyclopedia of Behavioral Medicine, Springer, 2013, pp. 841–842, doi:10.1007/978-1-4419-1005-9_690.
[306] K. Ono, J.I. Kushida, Landscape Estimation of Decision-tree Induction Based on Grammatical Evolution Using Rank Correlation, in: CEC-2017, IEEE, 2017, pp. 781–788, doi:10.1109/CEC.2017.7969389.
[307] C. Ferreira, Gene expression programming: Mathematical modeling by an artificial intelligence, Springer, 2006, pp. 337–380, doi:10.1007/3-540-32849-1_9. Ch. Decision Tree Induction
[308] W. Wang, Q. Li, S. Han, H. Lin, A preliminary study on constructing decision tree with gene expression programming, IEEE, Beijing, China, 2006, pp. 222–225, doi:10.1109/ICICIC.2006.22. ICICIC'06, Vol. I
[309] S.A. Mitrofanov, Application of genetic programming algorithm for designing decision trees and their ensembles, IOP Conf. Series: Materials Science and Engineering 734 (2020) 012098, doi:10.1088/1757-899x/734/1/012098.
[310] A.R.R. Freitas, R.C.P. Silva, F.G. Guimarães, Differential Evolution and Perceptron Decision Trees for Fault Detection in Power Transformers, in: V. Snášel (Ed.), SOCO Models in Industrial & Environmental Appl., Vol. 188 of AISC, Springer, 2013, pp. 143–152, doi:10.1007/978-3-642-32922-7_15.

[311] I.L.S. Russo, H.S. Bernardino, C.C.H. Borges, H.J.C. Barbosa, An Initialization Method for Grammatical Evolution Assisted by Decision Trees, in: CEC-2016, IEEE, 2016, pp. 3300–3307, doi:10.1109/CEC.2016.7744207.
[312] W. Weihong, R. Wei, L. Qu, Fuzzy Decision Tree Construction with Gene Expression Programming, in: ISKE 2010, IEEE, 2010, pp. 244–248, doi:10.1109/ISKE.2010.5680877.
[313] D. Wickramarachchi, B. Robertson, M. Reale, C. Price, J. Brown, HHCART: An oblique decision tree, Computational Statistics & Data Analysis 96 (2016) 12–23, doi:10.1016/j.csda.2015.11.006.
[314] M. Zorman, S.H. Babic, M. Sprogar, Advanced Tool for Building Decision Trees Mtdecit 2.0, in: H.R. Arabnia (Ed.), IC-AI'99, volume 1, CSREA Press, Las Vegas, USA, 1999, pp. 315–318.
[315] Y. Freund, R.E. Schapire, P. Vitányi, A decision-theoretic generalization of on-line learning and an application to boosting, EuroCOLT'95, Vol. 904 of LNCS (1995) 23–37, doi:10.1007/3-540-59119-2_166.
[316] H. Shi, Best-first decision tree learning, Master's thesis, 2007. University of Waikato
[317] M. Bursa, L. Lhotska, Automated classification tree evolution through hybrid metaheuristics, in: Innovations in Hybrid Intelligent Systems, Springer, 2007, pp. 191–198, doi:10.1007/978-3-540-74972-1_26. 44 of ASC
[318] U. Boryczka, J. Kozak, An Adaptive Discretization in the ACDT Algorithm for Continuous Attributes, in: Computational Collective Intelligence. Technologies and Applications, Springer, 2011, pp. 475–484, doi:10.1007/978-3-642-23938-0_48.
[319] R.D. Santos, P.C. Naval, Induction of Multiple Decision Trees Using Multiobjective Particle Swarm Optimization, in: L. Jiao (Ed.), Advances in Natural Computation and Data Mining, Xidian University Press, Xi'an, China, 2006, pp. 102–113.
[320] A.J. Malik, F.A. Khan, A hybrid technique using binary particle swarm optimization and decision tree pruning for network intrusion detection, Cluster Comput 21 (1) (2018) 667–680, doi:10.1007/s10586-017-0971-8.
[321] C.B. Veenhuis, M. Koppen, J. Kruger, B. Nickolay, Tree Swarm Optimization: An Approach to PSO-Based Tree Discovery, in: CEC-2005, volume 2, IEEE, Edinburgh, Scotland, 2005, pp. 1238–1245, doi:10.1109/CEC.2005.1554832.
[322] I. Bida, S. Aouat, A New Approach Based on Bat Algorithm for Inducing Optimal Decision Trees Classifiers, in: A. Rocha, M. Serrhini (Eds.), EMENA-ISTL 2018, Vol. 111 of SIST, Springer, 2019, pp. 631–640, doi:10.1007/978-3-030-03577-8_69.
[323] M. Bursa, L. Lhotska, E.M. Renda, et al., Ant-inspired Algorithms for Decision Tree Induction, in: ITBAM 2015, Vol. 9267 of LNCS, Springer, 2015, pp. 95–106, doi:10.1007/978-3-319-22741-2_9.
[324] J. Kozak, U. Boryczka, Collective data mining in the ant colony decision tree approach, Inf Sci (Ny) 372 (2016) 126–147, doi:10.1016/j.ins.2016.08.051.
[325] J.E. Fieldsend, Optimizing Decision Trees Using Multi-objective Particle Swarm Optimization, in: C.A. Coello-Coello (Ed.), Swarm Intelligence for Multi-objective Problems in Data Mining, Vol. 242 of SCI, Springer, Berlin, 2009, pp. 93–114, doi:10.1007/978-3-642-03625-5_5.
[326] C.L. Chan, C.Y. Lee, N.P. Yang, S.Y. Shen, Classification method incorporating decision tree with particle swarm optimization, in: ICGEC 2011, IEEE, 2011, pp. 216–219, doi:10.1109/ICGEC.2011.59.
[327] Y.J. Cho, H.S. Lee, C.H. Jun, Optimization of decision tree for classification using a particle swarm, Industrial Engineering and Management Systems 10 (4) (2011) 272–278, doi:10.7232/iems.2011.10.4.272.
[328] B.H. Nguyen, B. Xue, M. Zhang, A survey on swarm intelligence approaches to feature selection in data mining, Swarm Evol Comput 54 (2020) 100663, doi:10.1016/j.swevo.2020.100663.
[329] M. Rostami, K. Berahmand, E. Nasiri, S. Forouzande, Review of swarm intelligence-based feature selection methods, Eng Appl Artif Intell 100 (2021) 104210, doi:10.1016/j.engappai.2021.104210.
[330] J. Yang, L. Qu, Y. Shen, Y. Shi, S. Cheng, J. Zhao, X. Shen, Swarm intelligence in data science: Applications, opportunities and challenges, in: Advances in Swarm Intelligence, Vol. 12145 of LNCS, Springer, Cham, 2020, pp. 3–14, doi:10.1007/978-3-030-53956-6_1.
[331] R.S. Parpinelli, H.S. Lopes, A.A. Freitas, Data mining with an ant colony optimization algorithm, IEEE Trans. on Evolutionary Computation 6 (4) (2002) 321–332, doi:10.1109/TEVC.2002.802452.
[332] F.E.B. Otero, A.A. Freitas, C.G. Johnson, Cant-miner: An Ant Colony Classification Algorithm to Cope with Continuous Attributes, in: D. M. (Ed.), ANTS 2008, Vol. 5217 of LNCS, Springer, Berlin, 2008, pp. 48–59, doi:10.1007/978-3-540-87527-7_5.
[333] T. Nyathi, N. Pillay, Automated Design of Genetic Programming Classification Algorithms Using a Genetic Algorithm, in: G. Squillero (Ed.), EvoApplications 2017, Vol. 10200 of LNCS, Springer, Amsterdam, The Netherlands, 2017, pp. 224–239, doi:10.1007/978-3-319-55792-2_15.
[334] R.C. Barros, M.P. Basgalupp, A.C.P.L.F. Carvalho, A.A. Freitas, N. Krasnogor, Towards the automatic design of decision tree induction algorithms, 2011, GECCO'11, ACM. 567–574, 10.1145/2001858.2002050
[335] S. Kumar, S. Ratnoo, J. Vashishtha, Hyper-heuristic evolutionary approach for constructing decision tree classifiers, Journal of Information and Communication Technology 20 (2) (2021) 249–276, doi:10.32890/jict2021.20.2.5.
[336] A. Vella, D. Corne, C. Murphy, Hyper-heuristic decision tree induction, in: NaBIC 2009, IEEE, 2009, pp. 409–414, doi:10.1109/NABIC.2009.5393568.
[337] M.P. Basgalupp, R.C. Barros, V. Podgorelec, Evolving decision-tree induction algorithms with a multi-objective hyper-heuristic, 2015, SAC'15, ACM. 110–117, 10.1145/2695664.2695828,
[338] T. Nyathi, N. Pillay, Comparison of a genetic algorithm to grammatical evolution for automated design of genetic programming classification algorithms, Expert Syst Appl 104 (2018) 213–234, doi:10.1016/j.eswa.2018.03.030.
[339] M.P. Basgalupp, R.C. Barros, T. Barabasz, A grammatical evolution based hyper-heuristic for the automatic design of split criteria, C. Igel, 2014, GECCO'14, ACM. 1311–1318, 10.1145/2576768.2598327.
[340] R.C. Barros, M.P. Basgalupp, A.C.P.L.F. Carvalho, A.A. Freitas, Automatic design of decision-tree algorithms with evolutionary algorithms, Evol Comput 21 (4) (2013) 659–684, doi:10.1162/EVCO_a_00101.
[341] M. Jovanović, B. Delibašić, M. Vukićević, M. Suknović, M. Martić, Evolutionary approach for automated component-based decision tree algorithm design, Intell. Data Anal. 18 (1) (2014) 63–77, doi:10.3233/IDA-130628.
[342] T.M. Therneau, E.J. Atkinson, An introduction to recursive partitioning using the RPART routines, Tech. rep., Mayo Foundation (1997).
[343] Y. Freund, L. Mason, L. Saitta, The alternating decision tree learning algorithm, 1999, ICML'96, Morgan Kaufmann. 124–133,
[344] N. Landwehr, M. Hall, E. Frank, Logistic model trees, Mach Learn 95 (1–2) (2005) 161–205, doi:10.1007/s10994-005-0466-3.

Scopus

# Citation overview results

< Back to citation overview

Induction of decision trees as classification models through metaheuristics

Rivera-Lopez R., Canul-Reich J., Mezura-Montes E., Cruz-Chavez M.A.

2022, Swarm and Evolutionary Computation,

Is cited 1 time in Scopus by:

Search within results...

Refine results

Limit to Exclude

Year

| | |
|---|---|
| 2022 | (1) |

Author name

| | |
|---|---|
| Czajkowski, M. | (1) |
| Jurczuk, K. | (1) |
| Kretowski, M. | (1) |

Subject area

| | |
|---|---|
| Computer Science | (1) |

Document type

Publication stage

Source title

Keyword

Affiliation

Funding sponsor

Country/territory

Source type

Language

Limit to Exclude

Export refine

Analyze search results Show all abstracts Sort on: Date (newest)

All Export Download View citation overview View cited by Add to List •••

| | Document title | Authors | Year | Source | Cited |
|---|---|---|---|---|---|
| 1 | GPU-based acceleration of evolutionary induction of model trees | Jurczuk, K., Czajkowski, M., Kretowski, M. | 2022 | Applied Soft Computing 119,108503 | 0 |

View abstract View at Publisher Related documents

Display: 100 results per page

1

Top of page