UJAT
Universidad Juárez Autónoma de Tabasco
"Estudio en la duda. Acción en la fe"

DAIS 11111000011
División Académica de Informática y Sistemas

Universidad Juárez Autónoma de Tabasco

Tesis Doctoral

Differential-Evolution-based methods for inducing Decision Trees

Que presenta
Rafael Rivera López

Para obtener el grado de:
Doctor en Ciencias de la Computación

Directora:
Dra. Juana Canul Reich

Línea de generación y aplicación del conocimiento:
Sistemas Inteligentes

Institución sede:
Universidad Juárez Autónoma de Tabasco

Cunduacán, Tabasco, México ***Febrero 2018***

Universidad Autonoma de Aguascalientes

Universidad Juárez Autónoma de Tabasco
"Estudio en la duda. Acción en la fe"

Universidad Veracruzana

UJAT
Universidad Juárez Autónoma de Tabasco
"Estudio en la duda. Acción en la fe"

División Académica de Informática y Sistemas

Universidad Juárez Autónoma de Tabasco

Tesis Doctoral

Differential-Evolution-based methods for inducing Decision Trees

Que presenta
Rafael Rivera López

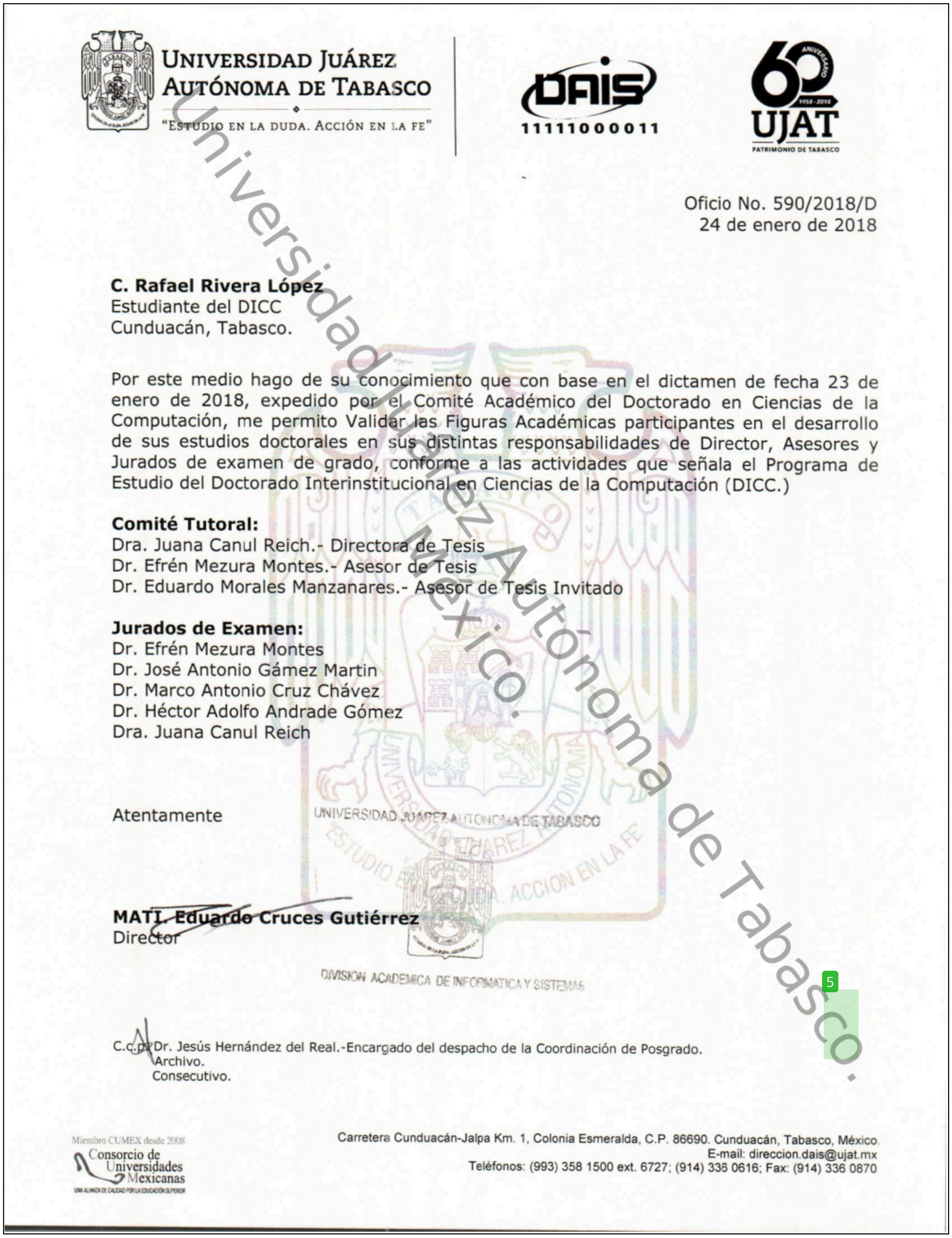
Para obtener el grado de:
Doctor en Ciencias de la Computación

Comité tutoral: **Dra. Juana Canul Reich**
Dr. Efrén Mezura Montes
Dr. Eduardo Morales Manzanares (Invitado)

Jurado: **Dra. Juana Canul Reich**
Dr. Efrén Mezura Montes
Dr. Héctor Adolfo Andrade Gómez
Dr. José Antonio Gámez Martín
Dr. Marco Antonio Cruz Chávez

Cunduacán, Tabasco, México ***Febrero 2018***

Universidad Autonoma de Aguascalientes
Universidad Juárez Autónoma de Tabasco
"Estudio en la duda. Acción en la fe"
Universidad Veracruzana



UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO

"ESTUDIO EN LA DUDA. ACCIÓN EN LA FE"

Oficio No. 590/2018/D
24 de enero de 2018

C. Rafael Rivera López
Estudiante del DICC
Cunduacán, Tabasco.

Por este medio hago de su conocimiento que con base en el dictamen de fecha 23 de enero de 2018, expedido por el Comité Académico del Doctorado en Ciencias de la Computación, me permito Validar las Figuras Académicas participantes en el desarrollo de sus estudios doctorales en sus distintas responsabilidades de Director, Asesores y Jurados de examen de grado, conforme a las actividades que señala el Programa de Estudio del Doctorado Interinstitucional en Ciencias de la Computación (DICC.)

Comité Tutoral:
Dra. Juana Canul Reich.- Directora de Tesis
Dr. Efrén Mezura Montes.- Asesor de Tesis
Dr. Eduardo Morales Manzanares.- Asesor de Tesis Invitado

Jurados de Examen:
Dr. Efrén Mezura Montes
Dr. José Antonio Gámez Martin
Dr. Marco Antonio Cruz Chávez
Dr. Héctor Adolfo Andrade Gómez
Dra. Juana Canul Reich

Atentamente

UNIVERSIDAD JUAREZ AUTONOMA DE TABASCO

[illegible]

MATI. Eduardo Cruces Gutiérrez
Director

DIVISION ACADEMICA DE INFORMATICA Y SISTEMAS

C.c.p. Dr. Jesús Hernández del Real.-Encargado del despacho de la Coordinación de Posgrado.
Archivo.
Consecutivo.

Miembro CUMEX desde 2008
Consorcio de Universidades Mexicanas
UNA ALIANZA DE CALIDAD POR LA EDUCACIÓN SUPERIOR

Carretera Cunduacán-Jalpa Km. 1, Colonia Esmeralda, C.P. 86690. Cunduacán, Tabasco, México.
E-mail: direccion.dais@ujat.mx
Teléfonos: (993) 358 1500 ext. 6727; (914) 336 0616; Fax: (914) 336 0870

UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO
"ESTUDIO EN LA DUDA. ACCIÓN EN LA FE"

DAIS 11111000011

60 ANIVERSARIO 1958 - 2018 UJAT PATRIMONIO DE TABASCO

Cunduacan, Tab. a 2 de febrero de 2018

En la Universidad Juárez Autónoma de Tabasco, de acuerdo al Reglamento de Estudios de Posgrado vigente, se revisó el trabajo de investigación titulado **"Differential-Evolution-based methods for inducing Decision Trees"**, realizado por el C. Rafael Rivera López, estudiante del Doctorado Interinstitucional en Ciencias de la Computación para obtener el grado de Doctor en Ciencias de la Computación bajo la modalidad de tesis.

Los integrantes del jurado, después de revisar el trabajo, y en virtud de que se han atendido satisfactoriamente las observaciones y recomendaciones, otorgamos nuestra aprobación para que se continúen los trámites correspondientes a la obtención del grado.

[illegible]
Dr. Efrén Mezura Montes
Profesor Investigador

[illegible]
Dr. José Antonio Gámez Martín
Profesor Investigador

[illegible]
Dr. Marco Antonio Cruz Chávez
Profesor Investigador

[illegible]
Dr. Héctor Adolfo Andrade Gómez
Profesor Investigador

[illegible]
Dra. Juana Canul Reich
Profesora Investigadora

UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO

"ESTUDIO EN LA DUDA. ACCIÓN EN LA FE"

DAIS
11111000011

DIVISIÓN ACADÉMICA DE INFORMÁTICA Y SISTEMAS

Oficio No. 244/18/DAIS/D
02 de febrero 2018

C. Rafael Rivera López
Matrícula 142H9001

En virtud de que cumple satisfactoriamente los requisitos establecidos en el Reglamento General de Estudio de Posgrado vigente en la Universidad, informo a Usted que se autoriza la impresión del trabajo recepcional **"Differential-Evolution-based methods for inducing Decision Trees"**, para presentar examen y obtener el Grado de Doctor en Ciencias de la Computación bajo la modalidad de Tesis.

Sin otro particular, aprovecho la oportunidad para saludarle.

Atentamente

UNIVERSIDAD JUAREZ AUTONOMA DE TABASCO

DIVISION ACADEMICA DE INFORMATICA Y SISTEMAS

MATI. Eduardo Cruces Gutiérrez
Director

C.c.p. Dr. Jesús Hernández del Real.- Encargado del Despacho de la Coordinación de Posgrado.
Archivo.
Consecutivo.

Cesión de Derechos

Cunduacán, Tabasco, a 19 de enero de 2018

A quien corresponda:

Los abajo firmantes, declaramos que el trabajo de tesis doctoral titulado "**Differential-Evolution-based methods for inducing Decision Trees**" es de nuestra autoría intelectual y por lo tanto cedemos los derechos de comunicación pública, reproducción, difusión general y pyesta a disposición electrónica de la citata tesis doctoral, de forma gratuita y no exclusiva, a la Universidad Juárez Autónoma de Tabasco, a la cual relevamos de cualquier sanción y asumimos responder a cualquier reclamo de derechos de autor ante las autoridades competentes

Atentamente

Autores:

Rafael Rivera López	Laguna Superior 262, U.H. El Coyol, C.P. 91779, Veracruz, Veracruz	[illegible]
Dra. Juana Canul Reich	Revolución 361-1, Col. Atasta, C.P. 81100, Villahermosa, Centro, Tabasco	[illegible]

CARTA DE AUTORIZACIÓN

El que suscribe, autoriza por medio del presente escrito a la Universidad Juárez Autónoma de Tabasco para que utilice tanto física como digitalmente la Tesis de grado denominada "**Differential-Evolution-based methods for inducing Decision Trees**", de la cual soy autor y titular de los Derechos de Autor.

La finalidad del uso por parte de la Universidad Juárez Autónoma de Tabasco de la tesis antes mencionada, será única y exclusivamente para difusión, educación y sin fines de lucro; autorización que se hace de manera enunciativa más no limitativa para subirla a la Red Abierta de Bibliotecas Digitales (RABID) y a cualquier otra Red Académica con las que la Universidad tenga relación Institucional.

Por lo antes mencionado, libero a la Universidad Juárez Autónoma de Tabasco de cualquier reclamación legal que pudiera ejercer respecto al uso y manipulación de la Tesis mencionada y para los fines estipulados en éste documento.

Se firma la presente autorización en la Ciudad de Villahermosa, Tabasco a los 8 días del mes de noviembre del año 2017.

AUTORIZO

[illegible]

RAFAEL RIVERA LÓPEZ

Agradecimientos

Es muy importante alcanzar una meta, pero es más importante todo lo que se gana al perseguirla

ESTE documento de tesis tiene un gran significado para mí y para mis seres queridos. Representa la culminación de un anhelo, representa muchos años buenos y malos, representa momentos de alegría y de tristeza y frustración, de decisiones acertadas y erróneas, de todo eso que es intrínseco en la vida de una persona, pero sobre todo representa una promesa cumplida.

Este trabajo se lo dedico a mi familia: A Maripaz que ha estado siempre a mi lado, en buenas y malas, y que ha sufrido conmigo y por mí. A mis hijos que están ya haciendo sus vidas y que me han dado muchas alegrías y motivos para sentirme orgulloso. A mis padres, a quienes me debo, y a mis hermanos que siempre han estado conmigo y me han apoyado.

Son muchas las personas a quienes tengo que agradecer todo su apoyo para la culminación de este trabajo, pero sobre todo quiero expresar mi más profundo agradecimiento a la Doctora Juana Canul Reich, que no solo fue mi asesora, sino que ha sido mi guía y mi ejemplo a seguir. Le agradezco por toda su paciencia y sus atinadas sugerencias que le dieron forma a esta tesis, pero sobre todo por darme la oportunidad de trabajar con ella y poder culminar este gran objetivo de vida.

Agradezco a mis amigos Héctor Adolfo Andrade Gómez y Marco Antonio Cruz Chávez, quienes siempre estuvieron alentándome para concluir este trabajo, y que me honraron con participar como mi jurado. A los doctores Efrén Mezura Montes, Eduardo Morales Manzanares, María del Pilar Pozos Parra y José Antonio Gámez Martín, por participar como mis tutores y jurados, y por sus acertados comentarios y observaciones a este documento.

Agradezco a la Universidad Juárez Autónoma de Tabasco por permitirme formar parte de su programa de doctorado y por todas las atenciones prestadas para concluir en tiempo y forma con este trabajo, al Programa para el Desarrollo Profesional Docente (PRODEP) por el apoyo económico para solventar este proyecto de tesis y al Instituto Tecnológico de Veracruz por su apoyo para concluir esta tesis.

Abstract

THIS thesis describes the application of the differential evolution algorithm to induce oblique and axis-parallel decision trees. The differential evolution algorithm distinguishes itself by being a simple and straightforward metaheuristic that has been successfully applied to efficiently solve a large number of problems whose parameters are real-valued variables, producing better results than those obtained by other approaches. Even though the differential evolution algorithm has already been utilized in data mining tasks, its use to induce decision trees is reduced to one approach to building oblique trees in a global search approach.

The differential evolution algorithm is applied in this thesis to induce decision trees through two strategies: by its use inside a traditional recursive partition scheme, and by utilizing it to conduct a global search in the space of possible trees. In the first case, this metaheuristic searches for the most appropriate hyperplane coefficients to better split a set of training instances, optimizing some splitting criterion. With this scheme, the differential evolution algorithm is applied as many times as internal nodes are required to build the oblique decision tree. On the other hand, this evolutionary algorithm carries out a global search of one near-optimal decision tree. Each individual in the population encodes only the internal nodes of a complete binary decision tree stored in a fixed-length real-valued vector. The size of this vector is determined using both the number of attributes and the number of class labels of the training set whose model is induced. To obtain a feasible decision tree, first the vector is analyzed to build a partial tree with only internal nodes, and then the training set is used to insert the corresponding leaf nodes.

Two types of decision trees can be obtained using the global search strategy: oblique and axis-parallel decision trees. Using the differential evolution algorithm to find near-optimal hyperplanes of an oblique decision tree is very intuitive since the hyperplane coefficients values are taken from a continuous space, and this metaheuristic was devised to optimize real-valued vectors. However, the construction of axis-parallel decision trees encoded with real-valued vectors is a more complicated task due to each of its internal nodes uses only one attribute to split the training instances. In this thesis, one procedure to select each attribute of each test condition of this type of decision tree is introduced. In this procedure, a mapping scheme using the smallest-position-value rule and the training instances to build a feasible axis-parallel decision tree from one individual in the population is successfully applied. Once the evolutionary process reaches its stop condition, the best individual in the final population is refined to replace non-optimal leaf nodes with sub-trees, as well as it is pruned to reduce the possible overfitting generated by applying this refinement. This procedure allows inducing feasible decision trees with a different number of nodes, although they are represented using a fixed-length parameters vector.

To obtain reliable estimates of the predictive performance of the two strategies implemented in this thesis, and to compare their results with those achieved by other classification methods, a repeated stratified ten-fold cross-validation procedure is applied in the experimental study. Since the evolutionary process to find a near-optimal decision tree uses the training accuracy of each tree as its fitness value, the decision trees in the last population could be overtrained, and the tree with the best training accuracy in this population could show a worse predictive ability. In this thesis, with the aim of mitigating the effects of this overtraining, an

alternative scheme to select one decision tree from the population of trained trees is introduced. This scheme uses a subset of instances of the dataset, which are not utilized in the cross-validation process, to determine an independent accuracy for each decision tree in the final population and to select the best one with this new value. This new accuracy is referred in this thesis as the selection accuracy, so the tree with the best selection accuracy in the final population is used to calculate the test accuracy of the fold.

Finally, a statistical analysis of these results suggests that our approach is better as a decision tree induction method as compared with other supervised learning methods. Also, our results are comparable to those obtained with other robust classifiers such as Random Forest, and one multilayer-perceptron-based classifier.

Publications

1. Rafael Rivera-Lopez and Juana Canul-Reich (2018) Construction of near-optimal axis-parallel decision trees using a differential-evolution-based approach, in *IEEE Access*, vol. PP, no. 99, pp. 1-16, DOI : 10.1109/ACCESS.2017.2788700

2. Rafael Rivera-Lopez, Juana Canul-Reich, José Antonio Gámez Martín, José María Puerta Callejón (2017) OC1-DE: A Differential Evolution Based Approach for Inducing Oblique Decision Trees. In: Rutkowski L., Korytkowski M., Scherer R., Tadeusiewicz R., Zadeh L., Zurada J. (eds), *Proceeding of the 16th International Conference in Artificial Intelligence and Soft Computing (ICAISC 2017)*, Zakopane, Poland, Lecture Notes in Computer Science, vol 10245. Springer, DOI : 10.1007/978-3-319-59063-9_38

3. Rafael Rivera-Lopez, Juana Canul-Reich (2017) A Global Search Approach for Inducing Oblique Decision Trees Using Differential Evolution. In: Mouhoub M., Langlais P. (eds), *Proceeding of the 30th Canadian Conference on Artificial Intelligence (AI 2017)*, Edmonton, Canada, Lecture Notes in Computer Science, vol 10233. Springer, DOI : 10.1007/978-3-319-57351-9_3

Contents

List of Figures

List of Tables

Chapter 1

Introduction

All data has its beauty, but not everyone sees it

Damian Mingle

KNOWLEDGE discovery refers to the process of non-trivial extraction of potentially useful and previously unknown information from a dataset [123]. Within the stages of this process, data mining stands out since it allows analyzing the data and producing models for their representation. In particular, machine learning provides data mining with useful procedures to build these models, since many of the techniques aimed at information discovery are based on inductive learning. Decision trees, artificial neural networks, and support vector machines, as well as clustering methods, have been widely-used to build predictive models. The use of one particular machine learning technique to build a model from a specific set of training instances depends on the required level of interpretability, scalability, and robustness of the model produced. The ability to track and evaluate every step in the information extraction process is one of the most crucial factors for relying on the models gained from data mining methods [338]. In particular, decision trees are classification models characterized by their high levels of interpretability and robustness. Knowledge learned via a decision tree is understandable due to its graphical representation [169], and also decision trees can handle noise or data with missing values and to make correct predictions [264].

A decision tree is a hierarchical structure composed of a set of internal and leaf nodes. Each internal node evaluates a test condition consisting of a combination of one or more attributes of the dataset, and each leaf node has a class label. Each tree branch represents a sequence of decisions made by the model to determine the class membership of a new unclassified instance. Of the different types of decision trees described in the literature, both axis-parallel and oblique decision trees have aroused the interest of the researchers in the machine learning community. An axis-parallel decision tree evaluates a single attribute in each test condition, and oblique decision trees utilize a linear combination of attributes to split the instance space. Oblique decision trees commonly show better performance, and they are more compact than the axis-parallel decision trees, but they require more computational effort to induce them.

The performance and expressiveness of a decision tree depend both on the quality of the training instances used to build the model, and on its induction procedure. In the first case, appropriate sampling methods must be applied to obtain a set of instances as representative as possible of the problem. On the other hand, the use of an adequate partition criterion, the capacity to deal with continuous and multi-valued attributes, and the ability to handle with missing values, among other elements, define the relevance of an induction procedure. Since one or more induction procedures can be applied to the same training instances, several decision trees are consistent with these instances [257]. In practice, a decision maker prefers accurate and compact predictive models: accurate as they correctly describe the dataset from which they were induced and have a reduced generalization error, and compact because they have a small number of alternatives. Since each leaf node of a decision tree represents a possible sequence of decisions made by the user to determine the class membership of a new unclassified instance, decision trees are considered compact when having a reduced number of leaf nodes. The compactness of the tree can also be evaluated by the length of its branches, i. e. the size of the sequence of decisions made to classify one new unclassified instance.

Although 1 is known that one greedy criterion does not guarantee to find an optimal solution [147], decision trees are ordinarily constructed through a recursive partition strategy that searches an optimal local split of the training set at each stage of their induction process. On the other hand, algorithms implementing a global search strategy are capable of finding near-optimal decision trees, but they are computationally expensive [14], and a way of coping with this disadvantage is the use of metaheuristics such as evolutionary algorithms and swarm intelligence methods.

1.1 Motivation

Notwithstanding a considerable number of heuristic algorithms and classification approaches have been described in the existing literature, as well as the traditional decision tree induction algorithms such as C4.5 [303], CART [46], and OC1 [260] are faster and accurate, this thesis describes one differential-evolution-based approach to induce decision trees. This work is motived by the fact that metaheuristics can perform a global search in the space of classifications models contrary to the constructive approach of the traditional methods [125], and also that the differential evolution algorithm has demonstrated to be a very competitive and successful method to solve complex problems in comparison with other algorithms [70, 86, 281, 371]. In particular, since metaheuristics use intelligent search procedures combining their exploration and exploitation skills, thus providing a better way to discover the relationships between the attributes of the training set, their application to build classification models allows creating decision trees more compact and accurate than those induced with traditional methods.

It is important to point out that to locate near-optimal decision trees, the encoding scheme used by a metaheuristic to build decision trees must correctly represent their symbolic elements: test conditions and class labels. The representation schemes of several metaheuristics such as genetic algorithms, and genetic programming are capable of encoding these elements, and they have been commonly applied to induce decision trees. However, this is a challenge for other metaheuristics such as the differential evolution algorithm and the particle swarm optimization method, which have proven to be very efficient in solving complex prob-8ms, but they have been designed to handle real-valued representations. The differential evolution algorithm has been applied for solving optimization problems arising in several domains of science and engineering including economics, medicine, biotechnology, manufacturing and production, big data and data mining [288]. In data mining, it has been utilized to build models of 8assification [218], clustering [78], and rule generation [83]. Also, the differential evolution algorithm has been used in conjunction with artificial neural networks [218], support vector machines [221], Bayesian classifiers [141], instance-based classifiers [138] and decision trees [362] for the induction of classifiers.

Since the differential evolution algorithm is one of the most powerful metaheuristics to solve real-valued optimization problems, we apply it to build both oblique and axis-parallel decision trees. In the first case, as the task of finding a near-optimal oblique hyperplane with real-valued coefficients is an optimization problem in a continuous space, a recursive partitioning strategy to find the most suitable oblique hyperplane of each internal node of a decision tree is implemented in this thesis. In this strategy, the differential evolution algorithm replaces the standard splitting criterion. Furthermore, to take advantage of carrying out a global search with one metaheuristic, this thesis also proposes an approach in which the set of hyperplanes used as test conditions of one complete oblique decision tree are encoded in a real-valued vector, and the differential evolution algorithm evolves a population of oblique decision trees. On the other case, a scheme to represent an axis-parallel decision tree with a real-valued vector is also introduced in this thesis. Each individual in the population encodes the elements of one univariate test condition: 1) the categorical and numerical attributes, and 2) the threshold values associated with the numerical attributes. The differential evolution algorithm carries out a global search of a near-optimal axis-parallel decision tree. An advantage of this global search approach is that the differential evolution operators can be applied without any modification, and the individuals in the population represent only feasible decision trees.

1.2 Research objectives

The general objective of this thesis is the follows:

> To apply the differential evolution algorithm to build more compact and accurate oblique and axis-parallel decision trees than those obtained using other decision tree induction methods.

To achieve this objective an exhaustive analysis of the literature related to the application of metaheuristic-based approaches to induce decision trees is first developed. Four specific objectives were formulated for this dissertation, and they are listed below:

1. To find the coefficients of a near-optimal hyperplane which splits a set of training instances.
2. To define one procedure to estimate the size of one real-valued vector encoding the internal nodes of a complete decision tree, based on the characteristics of the dataset whose model is constructed.
3. To define one procedure to map a feasible decision tree from one real-valued vector.
4. To find a near-optimal decision tree through one global search in the space of the decision trees.

1.3 Hypothesis

The differential-evolution-based approaches to build decision trees generate more precise and compact classifiers as compare to other decision tree induction methods with a statistical significance level not greater than 0.05.

1.4 Contributions

The main contribution of this thesis is in the field of data mining, by introducing an approach based on the differential evolution algorithm to induce decision trees. In particular, to the best of our knowledge, it is the first algorithm to build axis-parallel decision trees through this metaheuristic. A specific breakdown of the main contributions is as follows:

- A procedure to estimate the number of nodes in a decision tree based on the characteristics of the dataset whose classification model is constructed. This scheme can be used in any metaheuristic that represents its candidate solutions as a fixed length sequence. This procedure is introduced in publication number 1.

- A scheme to build a feasible oblique tree: 1) from a real-valued vector representing only the internal nodes of the tree, and from 2) the training set to add leaf nodes. This scheme is introduced in publication number 2.

- A method to construct a feasible axis-parallel decision tree: 1) from a real-valued vector encoding only the internal nodes of the tree, using a rule to discretize the values representing the attributes evaluated in the test conditions, and from 2) the training set to add leaf nodes. This method is introduced in publication number 1.

1.5 Organization

This thesis is related to the knowledge discovery techniques, specifically in the development of supervised learning methods through bio-inspired algorithms. This work belongs to the Generation and Application of Knowledge Line named "Intelligent Systems" of the Artificial Intelligence area of the Computer Sciences Doctoral program of the Universidad Juárez Autónoma de Tabasco.

This document is organized as follows: Chapter 2 outlines the elements commonly considered when a decision tree induction method is implemented and describes the main drawbacks of greedy heuristics to induce decision trees. An overview of soft computing techniques is also presented in this chapter, with emphasis on the types of metaheuristics. A large part of this chapter is devoted to describing the studies in the existent literature implementing some metaheuristic to build decision trees, with focus on 1) the representation scheme of the candidate solutions and 2) the fitness function utilized for their quality evaluation. Finally, a comparative analysis of these studies is conducted considering two elements: an analysis of their constituent components and a comparison of their experimental studies.

In Chapter 3, the three algorithms proposed in this thesis are described. First, the application of a differential-evolution-based approach named OC1-DE to induce oblique decision trees in a recursive partitioning strategy is detailed. Next, a differential-evolution-based approach named DE-ODT to induce oblique decision trees in a global search strategy is outlined. In the DE-ODT algorithm, the internal nodes of a decision tree are encoded in a real-valued vector, and a population of them evolves using the training accuracy of each one as its fitness value. The height of a complete binary decision tree whose number of internal nodes is not less than the number of attributes in the training set is used to compute the size of the individual, and a procedure to map a feasible oblique decision tree from one individual is applied. The best decision tree in the final population is refined replacing some leaf nodes with sub-trees to improve its accuracy. The representation scheme of the candidate solutions used by the DE-ODT algorithm allows applying the differential evolution operators without any modification, and the procedure for mapping a real-valued chromosome into a feasible decision tree ensures to carry out an efficient search in the solution space. The same considerations utilized to represent an oblique decision tree are also applied in a differential-evolution-based approach to finding a near-optimal axis-parallel decision tree. In this approach, named DE-ADT, the procedure to map a feasible axis-parallel decision tree from one individual uses both the smallest-position-value rule and the training instances.

The experimental study carried out to analyze the performance of the three differential-evolution-based methods implemented in this thesis is detailed in Chapter 4. First, a description of the datasets used in this study as well as the definition of the parameters of each method is given. Then, both the model validation technique used in the experiments and the statistical tests applied to evaluate the results obtained are outlined, and a discussion about the performance of the DE-based methods is provided. Finally, in the last chapter, the conclusions and the future works are provided.

Chapter 2

Background

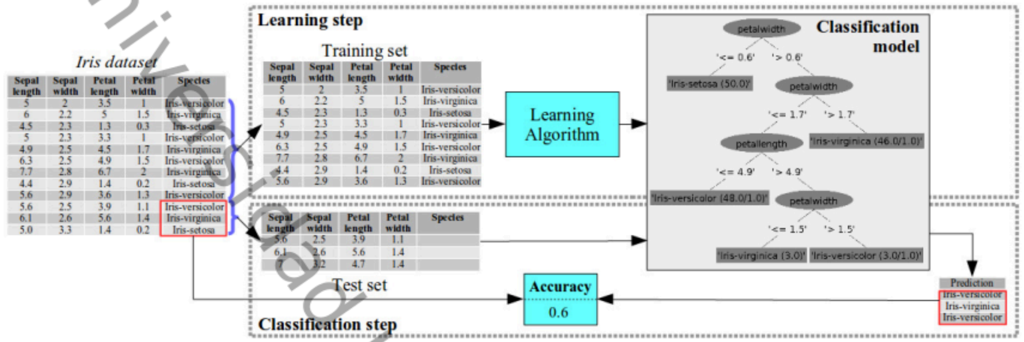
Learning algorithms are the seeds, data is the soil, and the learned programs are the grown plants. The machine-learning expert is like a farmer, sowing the seeds, irrigating and fertilizing the soil, and keeping an eye on the health of the crop but otherwise staying out of the way.

Pedro Domingos, The Master Algorithm

2.1 Classification techniques in machine learning

MACHINE learning is an exciting area of the artificial intelligence whose objective is that an artificial entity becomes able to improve its performance (it learns) from previously obtained results (its experience). Machine learning techniques to build models from known data have gained importance over the past few years due to the growing demand for data analysis in disciplines such as data science, business intelligence, and big data. It is widely known that the most representative machine learning approaches are supervised learning, where a model is learned from labeled data, and unsupervised learning, where a model is obtained from unlabeled data. Main supervised techniques are classification and regression, while clustering is the main unsupervised technique. Decision trees, artificial neural networks, and support vector machines have been widely-used to build predictive models, as well as clustering methods have been applied to construct descriptive methods. Many real-world problems such as the prediction of high-impact weather events [244], the analysis of traffic situations [321], the study of customer feedbacks [318], and the evaluation of credit risks [155], among other diverse applications, have benefited with such models.

Han, *et al.* [150] point out that data classification is a two-step process consisting of 1) a learning step when a classification model is built, and 2) a classification step when the model is utilized to predict the class membership of new unclassified instances, as shown in Fig. 2.1. The learning step uses a *training set* which is a group of pre-classified instances described by a vector $a = (a_1, a_2, \ldots, a_d)$ of d *attributes* representing the variables of one problem and by a vector $c = (c_1, \ldots, c_s)$ of s labels identifying the class membership of each instance. Each training instance is composed of a collection of attributes values and one class label. Each k-th attribute in the training set has associated a set of possible values known as its *domain* $D(a_k)$. The domain of a *categorical attribute* is a collection of unordered values and is a set of real numbers or integers for a *numerical attribute* [253].



Iris dataset

Sepal length	Sepal width	Petal length	Petal width	Species
5	2	3.5	1	Iris-versicolor
6	2.2	5	1.5	Iris-virginica
4.5	2.3	1.3	0.3	Iris-setosa
5	2.3	3.3	1	Iris-versicolor
4.9	2.5	4.5	1.7	Iris-virginica
6.3	2.5	4.9	1.5	Iris-versicolor
7.7	2.8	6.7	2	Iris-virginica
4.4	2.9	1.4	0.2	Iris-setosa
5.6	2.9	3.6	1.3	Iris-versicolor
5.6	2.5	3.9	1.1	Iris-versicolor
6.1	2.6	5.6	1.4	Iris-virginica
5.0	3.3	1.4	0.2	Iris-setosa

Training set

Sepal length	Sepal width	Petal length	Petal width	Species
5	2	3.5	1	Iris-versicolor
6	2.2	5	1.5	Iris-virginica
4.5	2.3	1.3	0.3	Iris-setosa
5	2.3	3.3	1	Iris-versicolor
4.9	2.5	4.5	1.7	Iris-virginica
6.3	2.5	4.9	1.5	Iris-versicolor
7.7	2.8	6.7	2	Iris-virginica
4.4	2.9	1.4	0.2	Iris-setosa
5.6	2.9	3.6	1.3	Iris-versicolor

Test set

Sepal length	Sepal width	Petal length	Petal width	Species
5.6	2.5	3.9	1.1	
6.1	2.6	5.6	1.4	
7	3.2	4.7	1.4	

Prediction
Iris-versicolor
Iris-virginica
Iris-versicolor

Figure 2.1: The learning and classification steps in a classification process.

Classification model performance is usually evaluated through its predictive accuracy, which is computed through the *test set*, although criteria such as speed, robustness, scalability, and others can also be applied [148]. Among the most widely used classification methods, the following stand out:

Decision trees: A decision tree (DT) is a hierarchical model classifying an instance using an ordered sequence of decisions. Typically, a top-down approach is used to induce a DT, starting with the complete training set that is recursively divided by a partition rule. This rule applies some metric to select the most appropriate test conditions to split the data. The most well-known algorithms to build DTs are ID3 [301], C4.5 [303] and CART [46].

Bayesian classifiers: Mitra and Acharya [250] point out that statistical methods are one of the oldest learning paradigms working under the assumption that statistical models can represent relationships between the attributes of a dataset. Han *et al.* [150] indicate that Bayesian classifiers predict the probability that an instance belongs to one class. Naïve Bayes Classifier (NB) is the simplest form of this type of classifiers. NB assumes that all the attributes of a sample are statistically independent given their class labels (conditional independence). On the other hand, Bayesian networks (BN) are graphical models allowing the representation of dependencies between subsets of attributes.

Artificial neural networks: Lippmann [226] indicates that an artificial neural network (ANN) consists of many nonlinear computational elements (nodes) connected by links associated with weighted variables operating in parallel. Basheer and Hajmeer [23] point out that ANNs have been successfully applied to solve many complex real-world problems in which learning is performed iteratively as the network processes the training instances, trying to simulate the way a human being learns from previous experiences. The ANNs study has its origins in 1943 with the McCulloch and Pitts work [243], but the interest in these models resurfaces in 1988 when Rumelhart *et al.* [312] publish one method to train a multi-layer perceptron (MLP) ANN. There are several ANN models such as the backpropagation neural network (BP-NN) [312], the recurrent neural network (RNN) [163], and the radial basis function neural network (RBF-NN) [50]. ANNs have a low level of interpretability and require some parameters usually determined empirically, but they are very noise tolerant.

Support Vector Machines: Vapnik *et al.* [367] introduced the vector support machines (SVM) in 1982 as a kernel-based method used to find the hyperplane best separating the training instances into two different classes. An SVM first maps these instances into an attribute space and then finds the separating hyperplane maximizing the margin[1] between their classes. Without any knowledge of the mapping scheme, the SVM finds the optimal hyperplane using a set of functions called kernels. The optimal hyperplane is described with a combination of entry points known as support vectors.

Instance-based classifiers: Mitra and Acharya [250] indicate that these classifiers use the characteristics of the nearest neighbor to estimate the class membership of an unclassified instance. The *k*-nearest neighbors (kNN) algorithm, proposed in 1967 by Cover and Hart [73], is one of the most representative instance-based classifiers. kNN finds the majority class among the *k* most similar training instances of a new unclassified instance and assigns it as its class label.

2.2 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 both *test conditions* (internal nodes) and *class labels* (leaf nodes). These nodes are joined by arcs simbolizing the possible outcomes of each test condition in the tree. A DT is a *rooted directed tree* $T = \big(G(V,E), v_1\big)$, where V is the set of nodes, E is the set of edges joining pairs of nodes in V, and v_1 is its *root node* [310]. In particular, if V has m nodes, for any $j = \{1,\dots,m\}$, the set of successor nodes of $v_j \in V$ is defined as follows:

$$\mathcal{N}^+(v_j) = \Big\{v_k \in V : k = \{1,\dots,m\} \wedge k \neq j \wedge (v_j, v_k) \in E\Big\}. \tag{2.1}$$

Furthermore, a DT is a data-driven classification model first induced using a training set and then applied to predict the class membership of new unclassified instances. Fig. 2.2 shows an example of a DT induced from the iris dataset [115] using the J48 method [386]. This dataset has four attributes, three class labels, and 150 instances.

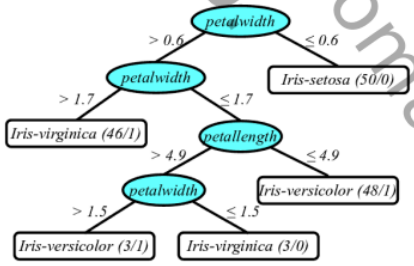


Figure 2.2: A DT induced from the iris dataset.

DTs stand out for their simplicity and their high level of interpretability, and since the decision tree induction (DTI) process determines the importance of the attributes when builds the test conditions, DTs provide an embedded feature selection mechanism [214]. These characteristics along with its predictive power allow placing to DT as one of the most widely used classifiers. DTs have been applied in several domains of science and engineering such as cellular biology [344], pharmaceutical research [32], public health [389], electrical energy consumption [358], and transport studies [167], among others.

Ritschard [308] discusses DTI origins. He argues that the work published by Belson in 1959 [23] could be the first study about a DTI process applied to analyze survey data previously collected. The Automatic

[1] One margin is the gap between the training instances closest to the hyperplane separating them [1].

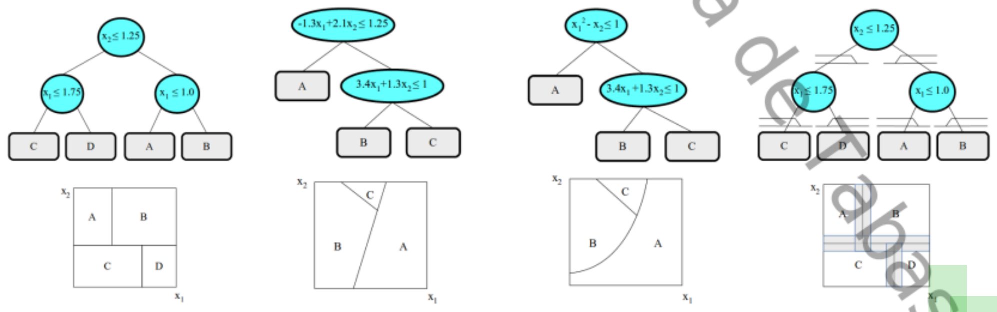
Interaction Detector (AID) method [252] is known as the first algorithm developed to induce regression trees, and the Concept Learning System (CLS), introduced by Hunt in 1966 [168], is considered as the patriarch of the classification tree induction methods [301]. It should be noted that CART [46] and C4.5 [303] have been recognized as two of the ten most influential methods of data mining [388].

The quality of the DTI procedure affects both performance and expressiveness of one DT. This procedure involves 1) the splitting criterion to measure the quality of the test conditions, 2) the way of dealing with numerical and categorical attributes, and 3) the mechanism to handle missing values. Furthermore, tree pruning techniques are also applied to eliminate overfitted tree branches and to try to improve the predictive power of the induced tree. Several studies have been conducted to describe, analyze, categorize and compare techniques for DTI such as those of Mingers [248], Safavian and Landgrebe [315], Brodley and Utgoff [49], Esposito *et al.* [104], Breslow and Aha [47], Murthy [259], Rokach and Maimon [310], Kotsiantis [198], Lomax and Vadera [237], Loh [234], and Barros *et al.* [17], among others.

2.2.1 Types of decision trees

Two types of DTs can be induced, following the number of attributes evaluated in each test condition: univariate and multivariate DTs. In a univariate DT, each test condition evaluates a single attribute to split the training set. On the other hand, a combination of attributes is used in each test condition of a multivariate DT. Two advantages of univariate DTs are their great interpretability as well as the simplicity of algorithms to build them; however, when the distribution of instances in the training set is complex, induced DTs include many internal nodes. On the other hand, multivariate DTs commonly show better performance, and they are smaller than univariate DTs, but these are less expressive and might require more computational effort to induce them.

Univariate DTs are also known as axis-parallel (AP) DTs since their test conditions represent axis-parallel hyperplanes dividing the instance space into two or more disjoint regions. If the test condition evaluates a numerical attribute, this hyperplane is defined with the inequality $x_i \leq c$, where x_i is the value of the i-th attribute in the training set, and c is a threshold value used to define the partition (Fig. 2.3(a)). Otherwise, if one categorical attribute is evaluated, the training set is split into as many subsets as values there are in the domain of the attribute.



(a) AP-DT (b) OB-DT (c) NL-DT (d) SF-DT

Figure 2.3: Decision tree types (Adapted from [287] and [14]).

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

$$\sum_{i=1}^{d} w_i x_i \leq \theta \tag{2.2}$$

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 θ represents the independent term in the hyperplane. Fig. 2.3(b) shows an example of one oblique DT. Similarly, non-linear (NL) DTs produce curved hypersurfaces as they utilize a non-linear combination of attributes. For example, the test condition used in the root-node of the DT shown in Fig. 2.3(c) represents a quadratic hypersurface.

Moreover, Wang *et al.* [381] argue that uncertainty such as fuzziness and ambiguity should be incorporated into the process of learning from training instances. Soft (SF) DTs implement a soft test at each internal node representing the probability that a branch from a node is selected based on the evaluation of its test condition. The soft DT shown in Fig. 2.3(d) represents a fuzzy DT.

2.2.2 Splitting criteria for decision tree induction

The splitting criterion applied to measure the quality of the test conditions in a DT is perhaps the element that has the greatest impact to determine both the effectiveness and the expressiveness of the classifier. These criteria can measure the impurity of a partition or estimate some discriminant value, and they also can evaluate some cost value. Besides, to consider the presence of uncertainty and ambiguity in the information, a soft measure should be included in a splitting criterion.

A typical way to group the vast number of partition criteria found in the existing literature is by the number of attributes used in each test condition. Several authors classify the univariate splitting criteria as information-theory-based criteria, distance-based criteria, and other criteria. Information gain (IG) [301], the G statistics [247] and the gain ratio (GR) [303] are the most representative information-theory-based splitting criteria. Gini index and the twoing rule [46] are the most commonly applied distance-based splitting criteria. Furthermore, several strategies such as the minimum description length (MDL) principle [305], the area under the curve (AUC) of the receiver operator characteristic (ROC) curve[2] [112] and other procedures also have been utilized as univariate splitting criteria. On the other hand, the linear discriminant analysis (LDA) [235], some mathematical-programming-based procedures [269, 341] and other mechanisms have been applied to define multivariate splitting criteria. Details of the splitting criteria used for DTI are discussed in several surveys such as those of Safavian and Landgrebe [315], Murthy [259], Rokach and Maimon [310], Lomax and Vadera [237], Lee *et al.* [217] and Barros *et al.* [17], among others.

2.2.3 Tree pruning approaches

Kotsiantis [198] indicates that tree pruning permits to generalize previously induced DTs by removing both nodes and sub-trees with the aim of avoiding overfitting. Also, tree pruning is applied to improve the comprehensibility level of a DT. Cost-complexity pruning [46], reduced-error pruning, and pessimistic-error pruning [302], as well as error-based pruning [303], are considered the most typical pruning methods. Detailed studies about tree pruning have been carried out by Mingers [248], Reed [307], Esposito *et al.* [104], and Breslow and Aha [47], among others.

[2]ROC curves are two-dimensional graphs in which the proportion of true positive cases in the data that are correctly identified as positive (sensitivity) is plotted on the Y-axis, and the proportion of true negative cases that are mistakenly identified as positive (false positive rate) is plotted on the X-axis [105].

2.2.4 Decision tree induction methods

There are several methods to build univariate DTs. The ID3 algorithm is introduced by Quinlan [301] to induce DTs from a training set with categorical attributes. Next, he develops the C4.5 method [303, 304] as an ID3 improvement including the use of real-valued attributes and one procedure to handle missing values. C4.5 also introduces a pruning tree stage. Furthermore, he implements the C5.0 version which first generates several classifiers rather than just one and then applies a voting scheme to predict the class membership of a new instance. On the other hand, the J48 procedure described by Witten *et al.* [387] is a well-known Java-based implementation of C4.5, and the REPTree algorithm [386] utilizes the reduced error pruning method to build a pruned DT.

In the case of multivariate DTs, two types of DTI procedures can be identified. First, those inducing only oblique DTs such as the linear machine DT (LMDT) algorithm described by Utgoff and Brodley [365] utilizing a linear machine as one test condition, and the multi-surface method (MSMT) implemented by Bennet [24] in which a linear-programming-based algorithm is applied to build test conditions. Next, those producing mixed DTs with both axis-parallel and oblique hyperplanes, such as the CART method introduced by Breiman *et al.* [46] to induce classification and regression trees, the LTree algorithm described by Gama [136] based on LDA to find better test conditions, the QUEST (quick, unbiased, efficient, statistical tree) algorithm developed by Loh and Shih [235] in which both statistical tests and one LDA-based procedure are applied to build test conditions, and the CRUISE (classification rule with unbiased interaction selection and estimation) approach described by Kim and Loh [190]. Furthermore, the non-linear DT (NDT) induction procedure [172] based on a combination of attributes and the augmentation of the attribute space, and the cost-sensitive non-linear DT (CSNL) method [366] are examples of non-linear DTI procedures.

On the other hand, the cost-sensitive ID3 (CS-ID3) algorithm [347], the IDX method [270], the economic generalizer 2 (EG2) algorithm [271], and the cost-sensitive J48 (CSJ48) method [386] are examples of cost-sensitive DTI algorithms. Finally, the fuzzy c-means clustering (FCM) algorithm [28], the fuzzy ID3 (F-ID3) method [69], and the C-Fuzzy DT approach [285] are examples of soft DTI methods.

2.2.5 Recursive partitioning problems

Most of the DTI methods described in the existing literature apply a recursive partitioning strategy implementing some splitting criterion to separate the training instances. This plan is usually complemented with a pruning procedure to improve the performance of the classifier. Several studies point out that this strategy has three fundamental problems: overfitting, selection bias towards multi-valued attributes, and instability to small changes in the training set.

A DT suffers from overfitting when its classification performance is lower than its learning performance. Overfitted DTs are more complex than necessary. Mitra and Acharya [250] indicate that this problem occurs when the data contain noise or irrelevant attributes, and when the training set size is small. On the other hand, several studies have shown that some splitting criteria are biased to select an attribute type over others, even though this selection affects the DT performance. For example, both the IG criterion and the Gini index are biased in favor of multi-valued attributes [384]. Hothorn *et al.* [165] established that DT expressiveness is affected by the biased attribute selection. Finally, Strobl *et al.* [342] 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 cutpoint, as well as the selection of the splitting attribute, strongly depend on the particular distribution of the training instances.

The incremental tree induction approach [364] and the use of ensembles of classifiers such as bagging, boosting, decision forest (DF) [157] and random forest (RD) [45] have been implemented as alternatives to avoid the problems of recursive partitioning strategies for DTI. On the other hand, algorithms implementing a global search strategy can ensure an efficient exploration of the solution space although it is known that building optimal DTs is NP-Hard [170]. In particular, the implementation of metaheuristic-based approaches for DTI allows constructing DTs that are more accurate than those inducing with traditional methods due to they use intelligent search procedures combining their exploration and exploitation skills, thus providing a better way to discover the relationships between the attributes used in the training set.

2.3 Metaheuristics as a component of soft computing

Soft-computing-based approaches have been widely used to solve complex problems in almost all areas of science and technology such as medicine, manufacturing, education, economics, big data, and data mining, among others. These approaches try to imitate the human reasoning process when solving a problem with the objective of obtaining acceptable results in a reasonable time. Bonissone [35] points out that real-world problems typically are ill-defined systems, difficult to model, and can have large solution spaces, and he also argues soft computing technologies provide us with a set of flexible computing tools to handle the available information and to solve these problems. Zadeh [394] coined the term soft computing to refer to three approaches used to solve problems with conditions of uncertainty or imprecision: fuzzy logic, ANN, and probabilistic reasoning. Fuzzy Logic [395] is considered an extension of boolean logic using a set of membership functions to represent the degree of truth of linguistic variables, and probabilistic reasoning is characterized by its ability to update previous outcome estimates by conditioning them with newly available evidence [35]. Fuzzy logic has been applied to build soft DTs and ANNs, and probabilistic reasoning methods such as NB have been used for classification tasks. As time went by, other techniques such as genetic algorithms [43] and SVM [184] have been included under this term.

On the other hand, Birattari [30] indicates that metaheuristics (MHs) are general algorithmic templates that can be easily adapted to solve almost all optimization problems. Du and Swamy [90] emphasize that the MHs are higher level procedures used to generate lower level search heuristics. MHs try to simulate both intelligent processes and behaviors observed in nature and other disciplines. These are characterized by combining the exploration of the search space to identify promising areas and exploitation of these areas to improve the known solution or solutions. MHs might or not might provide optimal solutions, however usually are largely satisfactory, in contrast to other techniques failing to solve the problem or spending excessive time finding the best solution. Nowadays MHs are widely accepted as a soft computing component [35, 43, 370].

Talbi [346] classifies MHs as single-solution-based (SS-based) MHs and population-based MHs. SS-based MHs implement intelligent search procedures that iteratively replace a candidate solution with a neighboring solution with the aim of reaching a near-optimal solution. Fig. 2.4 shows the general scheme of SS-based MHs. Main elements of this type of MHs are 1) a criterion to define neighborhood structure, 2) a selection process choosing the solution to replace the current solution, and 3) a stop condition of the search.

The following SS-based MHs have been used to implement DTI methods:

Stochastic Local Search (SLS): Hoos [162] indicates that SLS methods apply stochastic mechanisms such as probabilistic heuristics to find a near-optimal solution for a complex problem.

Simulated Annealing (SA): Kirkpatrick *et al.* [193] introduce SA as a method simulating the physical process of annealing in solids, where the aim is to achieve the lowest energy state for a system. Johnson *et al.* [180] point out that SA is motivated by the desire to avoid getting trapped in a local optimum, hence occasionally selects one worst solution to explore other areas of the search space.

Tabu Search (TS): Glover [142] describes TS as an iterative method applying a local search in the neighborhood of a candidate solution to provide a near-optimal solution for a complex problem. TS discards

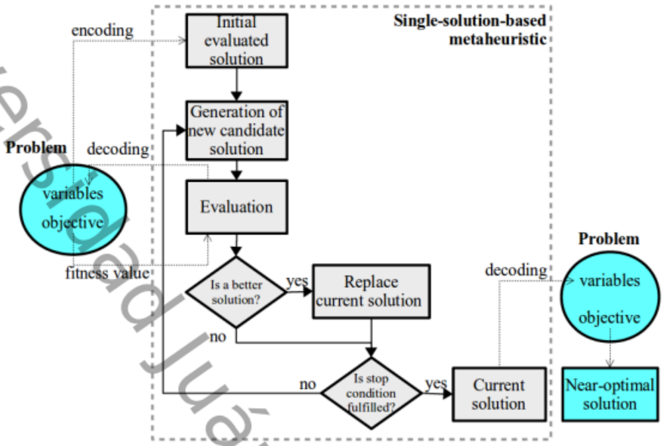


Figure 2.4: The general scheme of an SS-based MH.

the neighbors previously visited and puts them in a tabu list. Bennett and Blue [26] argue that the dynamically changing tabu lists, as well as the neighborhoods, help to move out of a local optimum and continue searching for better solutions without cycling.

Greedy randomized adaptive search procedure (GRASP): Feo and Resende [108] develop GRASP as a greedy search method that, instead of always selecting the best solution in the neighborhood of a current solution, it generates a group of best solutions and randomly selects one of them to replace the current solution.

On the other hand, population-based MHs use a group of candidate solutions in each step of their iterative process. Some of them generate new candidate solutions by recombining information from the current solutions, and some others update the properties of the candidate solutions. The most commonly used population-based MHs are related to evolutionary algorithms and swarm intelligence methods.

Evolutionary algorithms (EAs) are inspired by the theories synthesizing the Darwinian evolution through natural selection with the Mendelian genetic inheritance. Fig. 2.5 shows the general scheme of an EA.

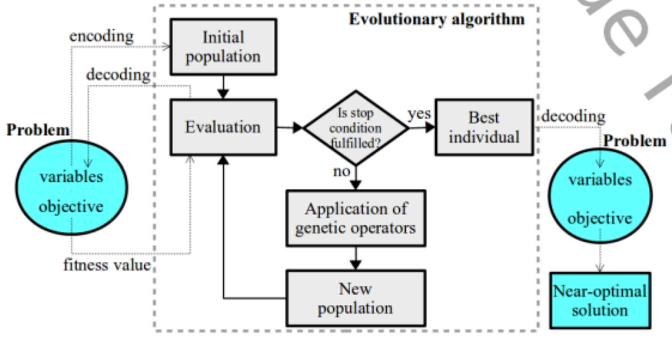


Figure 2.5: The general scheme of an EA.

In each iteration of its evolutionary process (known as a generation), a group of candidate solutions (individuals) evolves by applying selection (SEL), crossover (Xover) and mutation (MUT) operators. New populations of individuals are created until a stop condition is reached and then the best solution of the last population is returned. Commonly, two individuals (parents) are selected, and their values (genes) are recombined, and sometimes they are mutated, to create new candidate solutions (offsprings). Offsprings are considered to be part of the new generation. The following EAs have been used to implement DTI methods:

Evolutionary Strategies (ES): Rechenberg [306], Schwefel [320], and others developed ES as a global optimization algorithm to solve problems with real-valued parameters. An individual in ES encodes a candidate solution as well as its mutation parameters. Several ES variants have been developed, such as the (1+1)-ES generating a single offspring from a single parent, and the $(\mu+\lambda)$-ES using μ parents to create λ offsprings.

Genetic Algorithm (GA): Holland [158] describes a GA as a strategy to move from one population of candidate solutions to another applying a kind of natural selection along with the genetic operators of crossover and mutation. In GA, a candidate solution is commonly encoded using a linear chromosome, but other schemes have been applied. Each chromosome in the population is evaluated with a fitness function indicating the quality of the candidate solution.

Genetic Programming (GP): Koza [199] introduces GP to optimize a population of computer programs according to their ability to perform a task. Poli *et al.* [295] point out that GP programs are usually expressed as Lisp S-expressions (S-Exps) [242] in which variables and constants are leaf nodes (*terminals*) and operations are considered internal nodes (*functions*). Several GP variants such as the grammar-based GP (GGP) and the strongly-typed GP (TGP) have been implemented, to ensure that each chromosome in the population represents a feasible tree structure. In GGP, a grammar encoded using the Backus-Naur form (BNF) allows formally define GP structures and automatically ensures that the typing and syntax are maintained by manipulating the explicit derivation tree from the grammar [383]. On the other case, TGP is an enhanced version of GP applying data type constraints [251].

Co-evolutionary Algorithms (CEA): Lohn *et al.* [236] indicate that CEAs are inspired by the cooperation and the competition among populations of organisms in nature. Cooperation is presented to achieve a common objective and competition occurs when there are limited resources for many individuals. Hillis [156] points out that a competitive CEA is implemented when individual fitness value is evaluated through competition with other individuals in the population, rather than through an absolute fitness measure. On the other hand, Potter [296] argues that a cooperative CEA splits a problem into several components evolving independently, improving some fitness value. The interaction between populations occurs in the cooperative evaluation of each of their individuals.

Differential Evolution (DE): Storn and Price [340] develop DE as an EA evolving a population of real-valued vectors to find a near-optimal solution to an optimization problem. Instead of implementing traditional crossover and mutation operators, DE applies a linear combination of several randomly selected individuals to produce a new individual. DE is characterized by using fewer parameters than other EAs, and by its spontaneous self-adaptability, its diversity control, and its continuous improvement [109].

Grammatical Evolution (GE): Ryan *et al.* [313] introduce GE as an EA that uses a grammar to generate computer programs. GE simulates the gene expression process to produce a protein in an organism *(DNA $\rightarrow$ RNA $\rightarrow$ Protein)*. In the GE process *(binary string $\rightarrow$ integer string $\rightarrow$ tree)* one individual is a binary string that is decoded into an integer string, and one grammar is applied to build a computer program [18].

Gene Expression Programming (GEP): Ferreira [110] implements GEP as an EA using fixed-length linear chromosomes (genotypes) encoding expression trees (phenotypes). A linear chromosome is composed of one or more genes each one structurally divided into a head and a tail. The head encodes elements of the function and terminal sets, and tail works as a buffer of terminals to guarantee the formation of only valid structures. In GEP, a multigenic chromosome encodes several expression trees. GEP uses replication, mutations, transpositions, and recombinations as genetic operators. In the case of problems with real-valued constants, GEP introduces a new element in the chromosome structure known as the *Dc* domain. *Dc* is composed of symbols representing randomly created numerical constants (RNC). For simplicity, *Dc* symbols are the indexes of an RNC array.

Furthermore, Swarm intelligence (SI) methods are inspired by the collective behavior of some groups of animals such as ants, bees or birds. Vicsek and Zafeiris [372] indicate that the main feature of this behavior is that one individual action is dominated by the influence of the others. In SI methods, a group of candidate solutions (particles, agents) are moved in the search space by updating their properties combining local information with global information of the swarm transmitted by some type of communication scheme (Fig. 2.6).

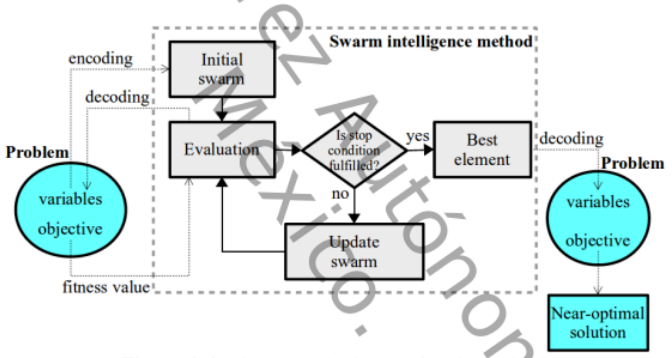


Figure 2.6: The general scheme of an SI method.

Two SI methods have been applied to build DTs:

Ant Colony Optimization (ACO): Dorigo [89] develops ACO as a swarm intelligence approach in which a colony of artificial ants is used to find near-optimal solutions to hard optimization problems. ACO mimics the foraging behavior of a colony of real ants. Blum [33] points out that ACO is inspired by the indirect communication between ants using chemical pheromone trials, which enables them to find short paths between nest and food.

Particle Swarm Optimization (PSO): Eberhart and Kennedy [97] introduce PSO as a population-based optimization technique inspired by the social behavior of a swarm of animals (ants, bees or birds) to obtain a promising position and to achieve specific objectives. In PSO each particle has a position and moves based on velocity updates.

Finally, Du and Swamy [90] point out that a hyperheuristic (HH) is a search procedure implemented to build algorithms that efficiently solve hard search problems. HHs explore the space of problem solvers rather than searching in the space of problem solutions (Fig. 2.7).

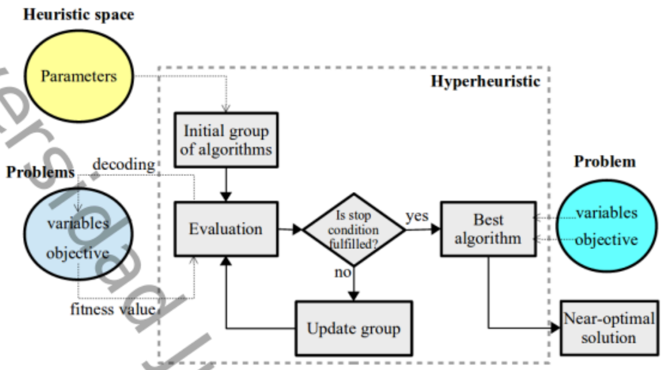


Figure 2.7: The general scheme of an HH method.

2.4 Differential evolution algorithm

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

In this thesis, the standard DE algorithm [340], named DE/rand/1/bin in agreement with the nomenclature adopted to refer DE variants, is used as a procedure to find a near-optimal solution. DE can be considered a three-step process including an initialization phase, the evolutionary process, and the final step determining the result obtained.

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

$$x_j^i = x_j^{\min} + r(x_j^{\max} - x_j^{\min}) \tag{2.3}$$

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

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

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

$$v^i = x^{r_1} + \mathrm{F}(x^{r_2} - x^{r_3}), \tag{2.4}$$

where F is a user-specified value representing a scale factor applied to control the differential variation.

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

$$u^i_j = \begin{cases} v^i_j & \text{if } r \leq \text{CR or } j = l, \\ x^i_j & \text{otherwise.} \end{cases} \tag{2.5}$$

4

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

In the final step, when a *stop condition* is fulfilled, DE returns the best candidate solution in the current population. The Algorithm 1 shows the structure of the classical DE/rand/1/bin method. Figure 2.8 shows an scheme of the application of the DE operators to build a new individual for the next population.

Algorithm 1 Classical DE algorithm introduced by Storn and Price in [340].

```
function DIFFERENTIALEVOLUTION(CR, F, NP, $n$)
    Input: The crossover rate (CR), the scale factor (F), the population size (NP), and the size of the
           real-valued vector ($n$).
    Output: The best individual in the last population ($x^{\text{best}}$).

    $g \leftarrow 0$
    $X_g \leftarrow \varnothing$
    for each $i \in \{1,\ldots,\text{NP}\}$ do
        for each $j \in \{1,\ldots,\text{n}\}$ do
            1 $x^i_j \leftarrow$ A randomly generated parameter using (2.3)
        end for
        $X_g \leftarrow X_g \cup \{(i, x^i)\}$
    end for
    while stop condition is not fullfilled do
        $g \leftarrow g + 1$
        $X_g \leftarrow \varnothing$
        for each $i \in \{1,\ldots,\text{NP}\}$ do
            $x^i \leftarrow$ Target vector from $X_{g-1}$
            $v^i \leftarrow$ Mutated vector generated using (2.4)
            $u^i \leftarrow$ Trial vector constructed using (2.5)
            $X_g \leftarrow X_g \cup \begin{cases} \{(i, u^i)\} & \text{if } f(u^i) \text{ is better than } f(x^i) \\ \{(i, x^i)\} & \text{otherwise} \end{cases}$
        end for
    end while
    $x^{\text{best}} \leftarrow$ The best individual in $X_g$
    return $x^{\text{best}}$
end function
```

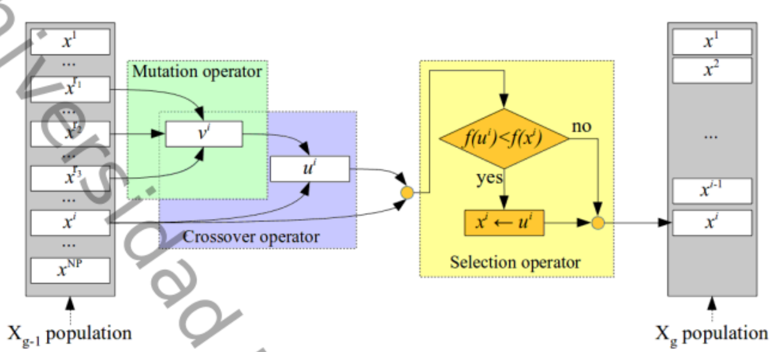


Figure 2.8: DE operators applied to build a new candidate solution for the next population.

DE has several advantages in comparison with other MHs such as the simplicity of its implementation, its ability to produce better results than those obtained by the others, and its low space complexity [80]. The mutation operator propitiates that the method exhibits a good trade-off between its exploitation and exploration skills, i.e., it is more explorative at the beginning, but it is more exploitative as the evolutionary process progresses [263]. Furthermore, the crossover and selection operators provide diversity control and continuous improvements in the DE algorithm, respectively [109]. On the other hand, although DE requires the definition of a smaller number of parameters compared to other MHs, its performance is sensitive to the values selected for CR, F, and NP [401].

DE has been utilized to implement classification methods in conjunction with SVMs [221], ANNs [218], Bayesian classifiers [141] and instance-based classifiers [138]. In the case of its use with DTs, DE is applied in the DEMO (DE for multiobjective optimization) algorithm [362] to find the most suitable parameters so that the J48 method [387] yields more accurate and small DTs. DE also is used in the PDT (Perceptron Decision Tree) algorithm [238] to find a near-optimal oblique DT. Each individual in the PDT method encodes the coefficients of all possible hyperplanes of one fixed-depth oblique DT.

Although DE was defined to solve problems with real-valued parameters, it has been applied with success to many combinatorial optimization problems such as the traveling salesman problem [350], the vehicle routing problem [249], and the flow shop scheduling problem [223], among others. Prado et al. [297] describe several DE adaptations to solve this type of problems such as those implementing a permutation matrix, those sorting the parameters vector, and those applying a forward/backward transformation between a real-valued vector and an integer-valued vector. In the permutation-based approach [298] a modified permutation matrix representing the difference between two randomly chosen vectors is applied to permute another randomly chosen vector, to build a mutant vector. Furthermore, in two relative-position-indexing-based approaches, the parameters values of a vector are ranked either in descending order with the largest-order-value (LOV) rule [299] or in ascending order with the smallest-position-value (SPV) rule [349] to obtain a sequence of discrete values. Finally, a forward/backward transformation [275] involves a three-step procedure: First, a set of integer-valued vectors are transformed into a set of real-valued individuals. Next, these individuals are manipulated using the DE operators, and then the resulting individuals are turned back into the integer domain.

2.5 Metaheuristics for decision tree induction

Several surveys describing the implementation of MH-based approaches for DTI have been previously published. Galea *et al.* [135] analyze different EA-based strategies to automated fuzzy knowledge acquisition through DTs and classification rules. Espejo *et al.* [103] surveys the existing literature on GP-based approaches to generate classification models such as DTs, classification rules, and discriminant functions. Kokol *et al.* [195] describe several EA-based approaches for DTI with focus on their application in medical domains such as breast cancer diagnosis, pediatric cardiology studies, and orthopedic fracture analysis, among others. Barros *et al.* [14] provide a detailed description of the application of EAs to induce classification and regression trees. Also, they describe the application of EAs to implement pruning methods and to handle cost-sensitive mechanisms. Finally, Kolçe and Frasheri (2014) [196] summarize some MH-based approaches for DTI using GAs and SS-based MHs such as SA and TS.

To describe the studies in the existing literature concerning MH-based approaches for DTI, in this thesis each study is associated to one of the following implementation strategies: recursive partitioning (RP), global search (GS), and subsequent optimization (SO). In the first case, several studies implement a recursive partitioning strategy utilizing an MH-based approach to find a near-optimal test condition for each internal node. Recursive partitioning with MHs is most commonly used to induce multivariate DTs. On the other hand, MH-based methods can perform a global search in the space of DTs with the aim of finding near-optimal DTs. Global search is commonly implemented through EAs. Finally, a different strategy is to apply an MH-based algorithm to optimize a DT previously induced by some classification method. This strategy is implemented to improve the DT performance or to introduce new characteristics such as membership functions for soft DTs. Fig. 2.9 shows a graphical scheme of these strategies.

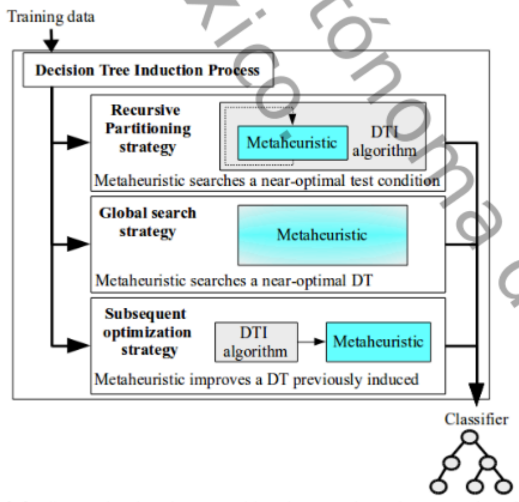

Training data
Decision Tree Induction Process
Recursive Partitioning strategy
Metaheuristic
DTI algorithm
Metaheuristic searches a near-optimal test condition
Global search strategy
Metaheuristic
Metaheuristic searches a near-optimal DT
Subsequent optimization strategy
DTI algorithm
Metaheuristic
Metaheuristic improves a DT previously induced
Classifier

Figure 2.9: Strategies implemented by the MH-based approaches for DTI.

In the following sections, a review of the studies in the existing literature implementing MH-based approaches for DTI is presented. This review is organized according to the type of MH used (SS-based MHs, EAs, and SI methods) and the type of the DT induced (AP, OB, NL, and SF). The implemented strategy in each study is identified as well as the main features of each MH-based approach such as the fitness function (FF), the set of variation operators, the candidate solutions representation scheme, and the procedure to build initial solutions, among others.

Several sampling methods, different performance measures, and various statistical tests have been used to carry out an experimental analysis of the MH-based approaches for DTI. Two types of studies can be distinguished in this review: those providing only a general description of their implementations, and those analyzing their experimental results. For the last case, the results obtained by one MH-based approach for DTI are compared with those obtained by other classification methods, whose list is given in Table 2.1.

Table 2.1: Methods used to compare the performance of MH-based approaches for DTI.

Method	Description	Method	Description
Methods to build univariate DTs:			
CHAID	Chi-square automatic interaction detection [183]	ID3	*Quinlan* (1986) [301]
ID3-S	*Fayyad & Irani* (1992) [107]	C4.5	*Quinlan* (1993, 1996) [303, 304]
RID3	Rank-based ID3 [280]	PART	*Frank & Witten* (1998) [121]
J48	*Witten & Frank* (1999) [387]	SampleC4.5	Statistical sampling method into C4.5 [129]
MtDeciT	Advanced tool for building DTs [11]	REPTree	Reduced-error pruning tree [386]
RTree	Random tree [386]	BFTree	Best-first DT [324]
jaDTI	Java DT implementation [197]		
Methods to build multivariate DTs:			
CART	Classification and regression trees [46]	LMDT	Linear machine DT [365]
MSMT	Multi-surface method [24]	NDT	Non-linear DT [172]
QUEST	Quick, unbiased, efficient, statistical tree [235]	RPART	Recursive Partitioning and Regression Trees [351]
LTree	Linear tree [137]	APDT	Alopex Perceptron DT [322]
CRUISE	Classification rule with unbiased interaction selection and estimation method [190]	LDSDT$_{BB}$	Discrete support vector DT method with B&B [277]
Methods to build cost-sensitive DTs:			
CS-ID3	Cost-sensitive ID3 [347]	IDX	*Norton* (1989) [270]
EG2	Economic generalider 2 [271]	MetaCost	*Domingos* (1999) [88]
CSJ48	Cost-sensitive J48 [386]		
Methods to build soft DTs:			
MB	Merging Branches [379]	C-Fuzzy DT	Clustered-oriented fuzzy DT [285]
Methods to build classification rules:			
CN2	*Clark & Niblett* (1989) [71]	1R	One-rule method [160]
ESIA	An extended supervised inductive algorithm [227]	CEFR-Miner	Co-Evolutionary Fuzzy Rule Miner [245]
PRIE	*Fawcett* (2008) [106]		
Ensemble methods:			
AdaBoost	Adaptive Boosting [127]	ADTree	Alternating DT [126]
Other methods:			
FR	Linear ratio method [115]	FW	Frank-Wolfe method [122]
LR	Logistic regression [74]	Huffman	Huffman algorithm [139]
IB1	Instance-based learning [5]	TF	Thresholding on one feature method [63]
PCA	Principal component analysis [63]	M5′	*Frank & Witten* (1998) [121]
LMT	Logistic model tree [216]	FlatOE	Flat Operator Equalization method [328]

Fitness function: This function defines the quality measure utilized to guide the search of a near-optimal candidate solution. In general, both uni-objective FF (UF) and multi-objective evaluation criteria have been used with MH-based approaches for DTI. A single fitness measure is applied with a uni-objective FF, and two or more fitness measures are assessed to estimate the quality of a candidate solution in the case of a multi-objective FF.

Following the multi-objective literature [72, 81], two schemes to determine the quality of a candidate solution are described in this review: On the one hand, an aggregating FF (AF) defines a weighted combination of fitness measures to assess the quality value of a candidate solution. Weighting coefficients represent the relative importance of each measure in the combination. On the other hand, a multi-objective FF (MF) first evaluates each measure separately and then applies: 1) a lexicographic ordering criterion to rank the fitness measures in order of importance, or 2) a Pareto-based fitness assignment strategy to find a set of non-dominated candidate solutions[3].

Several splitting criteria and various performance measures have been used as fitness measures in these approaches. The IG criterion [301], the MDL principle [305], and the GR [303, 304], as well as the Gini index and the twoing rule [46], have been utilized in several MH-based methods implementing a recursive partitioning strategy. Furthermore, the margin of separation (SepM) [277], the degree of linear separability (DLS) [322], and the dipolar criterion [34], as well as the Max minority (MaxM) [153], the sum minority (SumM) [153] and the sum of variances (SumV) [153] measures have been used to select the best hyperplane when an MH induces an oblique DT. Finally, the sum of the square-root-error (SSR) [326] is also applied as fitness measure in clustering-based DTI methods.

On the other hand, several performance measures such as accuracy (Acc), misclassification error (ME), misclassification cost (MC), size and running time have been used as fitness measures in DTI methods implementing both global search and subsequent optimization strategies. Furthermore, the following measures defined through the confusion matrix [194] also have been applied with these approaches: precision (P), sensitivity (Sen), specificity (Spe), false positive rate (FPR), false negative rate (FNR), and the F-measure (FM).

Representation scheme: Candidate solutions of the MH-based approaches for DTI have been represented either as linear sequences of values or as tree structures. Linear sequences of values are commonly used by several EAs such as GA, GE, GEP, and DE, and GP, GA, and CEAs utilize tree structures. If the first scheme is applied to represent DTs, a procedure to map a valid DTs from it must be defined, and when the second is employed, a set of particular variation operators to generate only feasible solutions must be applied. In particular, when a sequence of values is utilized with DTI methods to induce multivariate DTs, one candidate solution represents the combination of attributes evaluated by the test condition of each tree internal node.

Variation operators: These operators are applied to build new candidate solutions through two strategies: 1) by altering the values of the current solution, and 2) by merging the information of several candidate solutions. Each MH-based approach defines its variation operators in function of the representation scheme adopted, and with the aim to construct only feasible solutions.

Initialization procedure: Although the random generation of the initial candidate solutions is the strategy most commonly implemented by the MH-based approaches for DTI, some methods start their search process with a fixed solution, or they build several variants of a DT created with another induction method such as the C4.5 algorithm, the MSMT procedure, and the Branch & Bound (B&B) method [215].

Performances measures: A performance measure helps to compare and to evaluate the predictive power of a classifier [68, 113, 332]. Several performance measures can be defined using the elements of the confusion matrix [194], and the others are based on different criteria such as the ROC curve analysis [334], the AUC value [151], the Hypervolume (HV) metric [403], and the fidelity (Fid) of the induced DT [75].

[3] A solution is non-dominate if it does not exist another candidate solution better than the current one in some objective function without worsening other objective function [81].

Sampling methods: Hu *et al.* [166] indicate that the main goal of sampling is to select a representative subset of data from a dataset which will be used in an experimental study. Sampling methods are accepted as validation procedures to determine the generalization power of a classifier [68]. Both cross-validation (CV) and hold-out (HO) are the sampling methods most commonly used to evaluate the robustness of a classifier [10, 398]. An average of the results obtained from several repetitions of the validation procedure is usually applied to achieve more reliable performance results. Several CV variants such as k-fold CV [339], 5×2-CV [87], and the leave-one-out CV (LOOCV) [339] have been applied as sampling methods in some MH-based methods. Although the induction of DTs with the full dataset (FD) is not an adequate procedure to validate the performance of a classifier, this approach has been applied in some of the early MH-based methods, mainly to demonstrate the feasibility of their implementation.

Statistical tests: Demšar [85] describes and evaluates several statistical tests used to compare the performance of several classifiers. These criteria can be grouped into two categories: Those comparing two classifiers and those evaluating several classifiers. Both the ANOVA test [116] and the Friedman test [128] are the statistical tests most commonly used to examine the differences between the experimental results of several classifiers. Luengo *et al.* [239] indicate that numerous post-hoc tests have been developed to determine if an algorithm has statistical differences concerning other methods. Table 2.2 describes the statistical tests employed to compare the performance of the MH-based approaches.

Datasets: Two types of datasets have been used to compare the performance of MH-based approaches for DTI. The UCI machine learning repository [225] provides the collection of datasets most commonly employed by comparing classifiers. On the other hand, several studies apply different datasets in their experiments. They can be artificial or private datasets, as well as they can be obtained from other sources.

Table 2.2: Statistical tests used to compare the performance of the MH-based approaches for DTI.

Acronym	Description	Acronym	Description
Avg	Average of the results of each algorithm in several datasets	t-test	Paired t-test
WTL	Counts of wins, ties and losses (Sign test)	ANOVA	Analysis of variance [116]
ARR	Average ranks of the results of each algorithm in several datasets	Tuk-t	Tukey test [359]
Wil-t	Wilcoxon sum of ranks test [385]	Fri-t	Friedman test [128]
Nem-t	Nemenyi test [262]	BD-t	Bonferroni-Dunn test [94]
KW-t	Kruskall-Wallis test [209]	Hol-t	Holm test [159]
Hom-t	Hommel test [161]		

2.5.1 Single-solution-based metaheuristics

Several SS-based MHs such as SLS, SA, TS, and GRASP have been applied to induce DTs. These MH-based approaches are commonly implemented within a recursive partitioning strategy. The timeline of the SS-based MHs for DTI described in the existing literature is shown in Fig. 2.10.

Axis-Parallel DTs: Bucy and Diesposti [51, 52] implement a subsequent optimization strategy to improve the performance of a previously induced DT. Starting with a fixed-length binary DT randomly created, SA carries out two types of reconfigurations: 1) to swap two randomly selected test conditions, and 2) to obtain a new DT by randomly reassigning all test conditions, or by keeping the test conditions of one

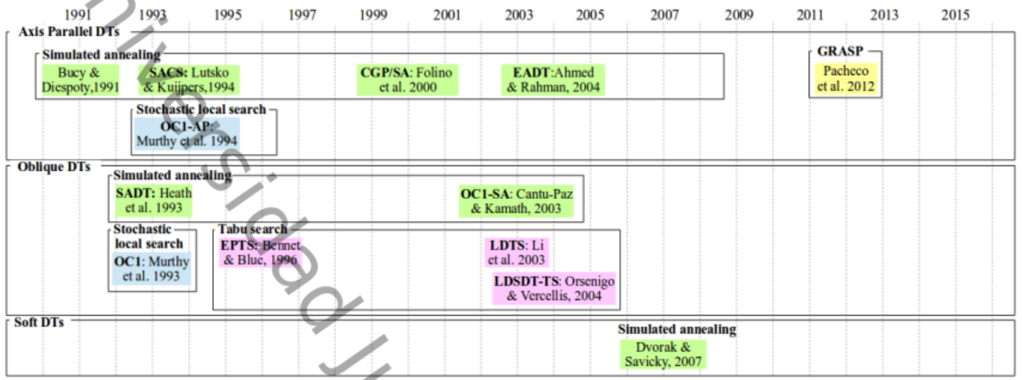


Figure 2.10: Timeline of the SS-based MHs for DTI.

tree branch intact, and randomly reassign the remaining ones. After each reconfiguration, a pruning process is applied to remove infeasible sub-trees. Reconfigured DT is accepted as a new solution through the Boltzmann criterion. A similar approach called Simulated Annealing Classifier System (SACS) to optimize a binary DT previously induced by the ID3 algorithm is detailed by Lutsko and Kuijpers [240]. In this method, one test condition is modified in each iteration. First, using a weighted distribution previously defined, a test condition is selected, and then the attribute in it is swapped by an unused attribute randomly chosen. Finally, the altered DT is evaluated and then is accepted as a new solution according to the Boltzmann criterion.

Folino *et al.* [118] describes the Cellular Genetic Programming algorithm coupled with Simulated Annealing (CGP/SA) as a global search strategy to evolve a population of DTs. Starting with a population of trees randomly created, CGP/SA uses a cellular automaton [353] to place them in a two-dimensional grid. Next, each DT is recombined with its best neighbor to generate two offsprings, and the best one replaces the DT in the grid through the Boltzmann criterion. Another approach using a group of DTs is described by Ahmed and Rahman [6], although in it there is no information exchange between the solutions. First, the collection is initialized with all possible DTs with three nodes. Then, for each DT, SA is applied in a recursive partitioning strategy to select a near-optimal node (an internal node or a leaf node) that will be added to the DT.

On the other hand, Pacheco *et al.* [279] implement a recursive partitioning strategy using GRASP to find the near-optimal test conditions of a binary DT. In each iteration, instead of choosing the attribute with the maximum IG to use it in a test condition, GRASP randomly selects one attribute of a subset of attributes with the highest IG values.

Oblique DTs: Several SS-based MHs have been used to build an oblique DT in a recursive partition strategy. A graphical description of these methods is shown in Fig. 2.11, where $w^T x = \theta$ is one hyperplane, w is the vector of the hyperplane coefficients, x is the vector of attribute values, and θ represents the independent term in the hyperplane. Also, w^s and x^s are subsets of coefficients and attribute values, respectively, and w' is the new vector of coefficients produced by the MH.

Murthy *et al.* [261] introduce the Oblique Classifier 1 (OC1) method which applies a two-step process to find a near-optimal hyperplane. First, it uses a deterministic rule to adjust the hyperplane coefficients, taking one at a time and looking for its optimal value. Next, it applies SLS to jump out of this optimum local value. The induced DT is pruned by removing sub-trees whose impurity value is

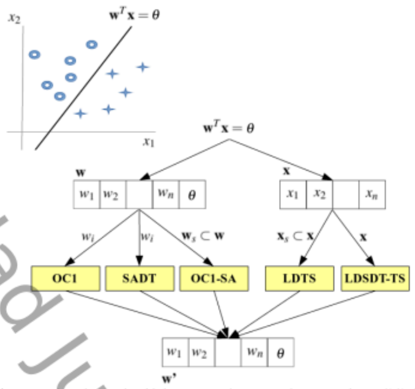

x_2
$\mathbf{w}^T\mathbf{x} = \theta$
x_1
$\mathbf{w}^T\mathbf{x} = \theta$
$\mathbf{w}$
w_1 w_2 w_n θ
$\mathbf{x}$
x_1 x_2 x_n
w_i
w_i
$\mathbf{w}_s \subset \mathbf{w}$
$\mathbf{x}_s \subset \mathbf{x}$
$\mathbf{x}$
OC1
SADT
OC1-SA
LDTS
LDSDT-TS
w_1 w_2 w_n θ
$\mathbf{w}'$

Figure 2.11: Methods to build a new hyperplane using SS-based MHs.

less than a predefined threshold value. Also, an OC1 variant to induce an axis-parallel DT, known as OC1-AP, is described in Murthy *et al.* [260].

Heath *et al.* [154] describe the Simulated Annealing of Decision Trees (SADT) method that, starting with a fixed initial hyperplane, applies SA to adjust it and improve its fitness value. In each iteration of the SADT method, SA modifies one of the hyperplane coefficients, which is randomly chosen, to adjust the position of the hyperplane. This modification is accepted as a valid movement through the Boltzmann criterion. Cantú-Paz and Kamath [59] implement an OC1 variant known as the OC1-SA method in which SA simultaneously modifies several hyperplane coefficients, but in this case, beginning with the best axis-parallel hyperplane found by the OC1-AP method. Cost-complexity pruning method is applied to improve the predictive performance of the induced DT.

Li *et al.* [224] implement an LDA-based approach in the Linear Discriminant and Tabu Search (LDTS) method. To find a near-optimal hyperplane, this method first randomly selects a subset of attributes to build the hyperplane, and iteratively replaces an attribute in this subset with another one outside it. This attribute is included in the subset only if it improves the IG of the new hyperplane, and the replaced attribute is added to the tabu list. Induced DT is pruned applying the cost-complexity pruning method. Also, Orsenigo and Vercellis [277] utilize TS in the discrete support vector DT method (LDSDT$_{\mathrm{TS}}$), implementing a linear discrete support vector machine (LDSVM) as its splitting criterion. A hyperplane in this method is modeled as a mixed integer linear program, which is solved using both the LDSVM and TS.

On the other hand, Bennet and Blue [26] describe a subsequent optimization algorithm known as Extreme Point Tabu Search (EPTS) modifying a DT previously induced by the MSMT method [24] with the aim of minimizing its misclassification error. The oblique DT is represented as a system of disjunctive linear inequalities [25], and TS is applied in the pivoting procedure of the Simplex method [77] 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, and the best neighbor is one that, when is considered as one basic attribute, improves the fitness value of the hyperplane.

Soft DTs: Dvořák and Savický [96] use SA to implement a subsequent optimization strategy in an approach to find soft threshold values that are assigned to the test conditions of an axis-parallel DT previously induced by the CART method. SA perturbs a subset of these values in each step of its iterative process.

Discussion

The components of the 13 studies implementing an SS-based MH for DTI are shown in Table 2.3. In this table is observed that only three studies use an aggregating FF and the remaining apply one uni-objective FF. To the best of our knowledge, the evaluation of candidate solutions with a multi-objective FF has not been applied to this type of DTI methods. In particular, seven studies use a splitting criterion such as IG and the twoing rule as their fitness measures, and six studies utilize performance measures such as ME and size. Furthermore, this table also shows that the linear representation of candidate solutions only has been utilized to build oblique DTs, and the matrix representation is applied in only one TS-based approach [26] in which an oblique DT is encoded as a set of disjunctive linear inequalities. Finally, several strategies have been implemented to generate the initial candidate solution: three studies create it randomly, and other works start their search procedure with a previously induced DT.

Table 2.3: Components of SS-based MHs for DTI.

Strategy	DT	FF	MH	Studies	Repr. scheme	Fitness measures	Initial solution
RP	AP	UF	GRASP	·*Pacheco et al.* (2012) [279]	Tree	IG	An empty DT
		AF	SA	·*Ahmed & Rahman* (2004) [6]	Tree	ME$\wedge$Size	All posible DTs with three nodes
	OB	UF	SLS	·OC1 [260, 261]	Linear	IG, MaxM, SumM, SumV, Gini, Twoing	The best axis-parallel hyperplane found by the OC1-AP method
			SA	·SADT [154]	Linear	SumM	A fixed hyperplane
				·OC1-SA [59]	Linear	Twoing	The best axis-parallel hyperplane found by the OC1-AP method
			TS	·LDTS [224]	Linear	IG	A subset of attributes randomly selected from the dataset
		AF	TS	·LDSDT$_{TS}$ [277]	Linear	ME$\wedge$SepM	A feasible hyperplane modeled as a linear mixed integer problem and solved using a truncated B&B method
GS	AP	UF	SA	·GCP/SA [118]	Tree	ME	A set of axis-parallel DTs randomly created
SO	AP	UF	SA	·SACS [240]	Tree	MDL	A DT induced by the ID3 method
		AF	SA	·*Bucy & Diesposti* (1991) [51, 52]	Tree	ME$\wedge$Size	A fixed-length binary DT randomly created
	OB	UF	TS	·EPTS [26]	Matrix	ME	A binary DT induced by the MSMT method
	SF	UF	SA	·*Dvořák & Savický* (2007) [96]	Tree	ME	A binary DT induced by the C5.0 method

The experimental analysis reported in these studies is described in Table 2.4. In this table is shown that ten studies use UCI datasets, and eight studies employ datasets from other sources. Eight studies implement one k-fold CV as their sampling method, and three approaches apply an hold-out sampling method. The experiments reported by Bucy and Diesposti [51, 52] was carried out using all the instances of the datasets. Accuracy, ME, and size are the performance measures adopted by six, seven, and eleven studies, respectively. Four studies apply a statistical test to analyze their experimental results, and the LDTS method [224] is the only one conducting one post-hoc analysis. Finally, the C4.5 method, the OC1 algorithm, and the CART method have been used to compare the results got by several SS-based MHs. Other classifiers such as kNN, BP-NN, and SVM also have been utilized in the experimental studies conducted by some of these MHs. The results generated by the kNN method and by the BP-NN algorithm are compared with those obtained by the OC1 method, and the results achieved by an SVM-based approach are contrasted with those yielded by the LDSDT$_{TS}$ method [277]. In particular, Pacheco *et al.* [279] implement a traditional DTI procedure using IG

Table 2.4: Experimental analysis reported by SS-based MHs for DTI.

Stra-tegy	**DT type**	**MH**	**Studies**	**Datasets UCI**	**Datasets other**	**Sampling method**	**Performance measures**	**Statistical tests**	**Compared methods**
RP	AP	GRASP	·*Pacheco et al.* (2012) [279]	17	-	10-f CV	Acc, Size	Avg, WTL, t-test	Traditional method
		SA	·*Ahmed & Rahman* (2004) [6]	17	-	10-f CV	Acc, Size	Avg	C4.5
	OB	SLS	·OC1 [260, 261]	2	2	10-f CV	Acc, Size	-	ID3, SADT, kNN, BP-NN
		SA	·SADT [154]	2	2	10-f CV	Acc, Size	-	ID3
			·OC1-SA [59]	10	-	5-f CV	Acc, Size, Time	t-test	CART, OC1, OC1-ES, OC1-GA
		TS	·LDTS [224]	13	-	10-f CV	ME, Size, Time	Avg, Tuk-t, ANOVA	C4.5, CART, OC1, QUEST, LTree
			·LDSDT_{TS} [277]	6	2	10-f CV	Acc, Size, Time	-	C4.5, OC1, SVM, QUEST, LDSDT_{BB}
GS	AP	SA	·GCP/SA [118]	12	-	HO	ME, Size	-	C4.5
SO	AP	SA	·*Bucy & Diesposti* (1991) [51, 52]	-	3	FD	ME, Size	-	Huffman
			·SACS [240]	-	10	HO	ME, Size	-	-
	OB	TS	·EPTS [26]	5	3	5-f CV	ME	-	MSMT, FW
	SF	SA	·*Dvořák & Savický* (2007) [96]	-	1	HO	ME	-	C5.0, CART

as its splitting criterion to compare the experimental results achieved by their GRASP-based method.

2.5.2 Evolutionary algorithms

EAs are the most widely used population-based MHs for DTI. A vast number of EA-based approaches implement a global search strategy to find near-optimal DTs, although recursive partitioning and subsequent optimization strategies also have been implemented using EAs. GA and GP are the methods most commonly used to induce DTs. Furthermore, CEA, ES, and other EA-based approaches also have been applied for DTI. A timeline of the EA-based approaches for DTI is shown in Fig. 2.12.

Genetic algorithms with linear chromosomes (LGA)

In the case of EA-based approaches for DTI, several types of chromosomes have been utilized such as binary, integer, and real-valued chromosomes, as shown in Table 2.5. A list of genetic operators used in these approaches is shown in Table 2.6.

In particular, the following procedure is commonly applied to construct a DT from a linear chromosome: First, the element in the initial location of the chromosome is used as the root node of the DT. Next, the remaining elements of the chromosome are inserted in the DT as successor nodes of those previously added so that each new level of the tree is completed before placing new nodes at the next level, in a similar way to the breadth-first search strategy. In the case of univariate DTs, the number of successor nodes of an internal node is calculated based on the domain of the attribute used in its test condition, and for a multivariate DT, each internal node has two successors nodes. This procedure is depicted in Fig. 2.13.

Axis-Parallel DTs: Kennedy *et al.* [185] introduce a global search strategy known as the Caltrop method to evolve a population of DTs represented as a set of sub-trees. Each sub-tree with three nodes (a caltrop) is encoded with a sequence of integers referring to binary attributes used as test conditions.

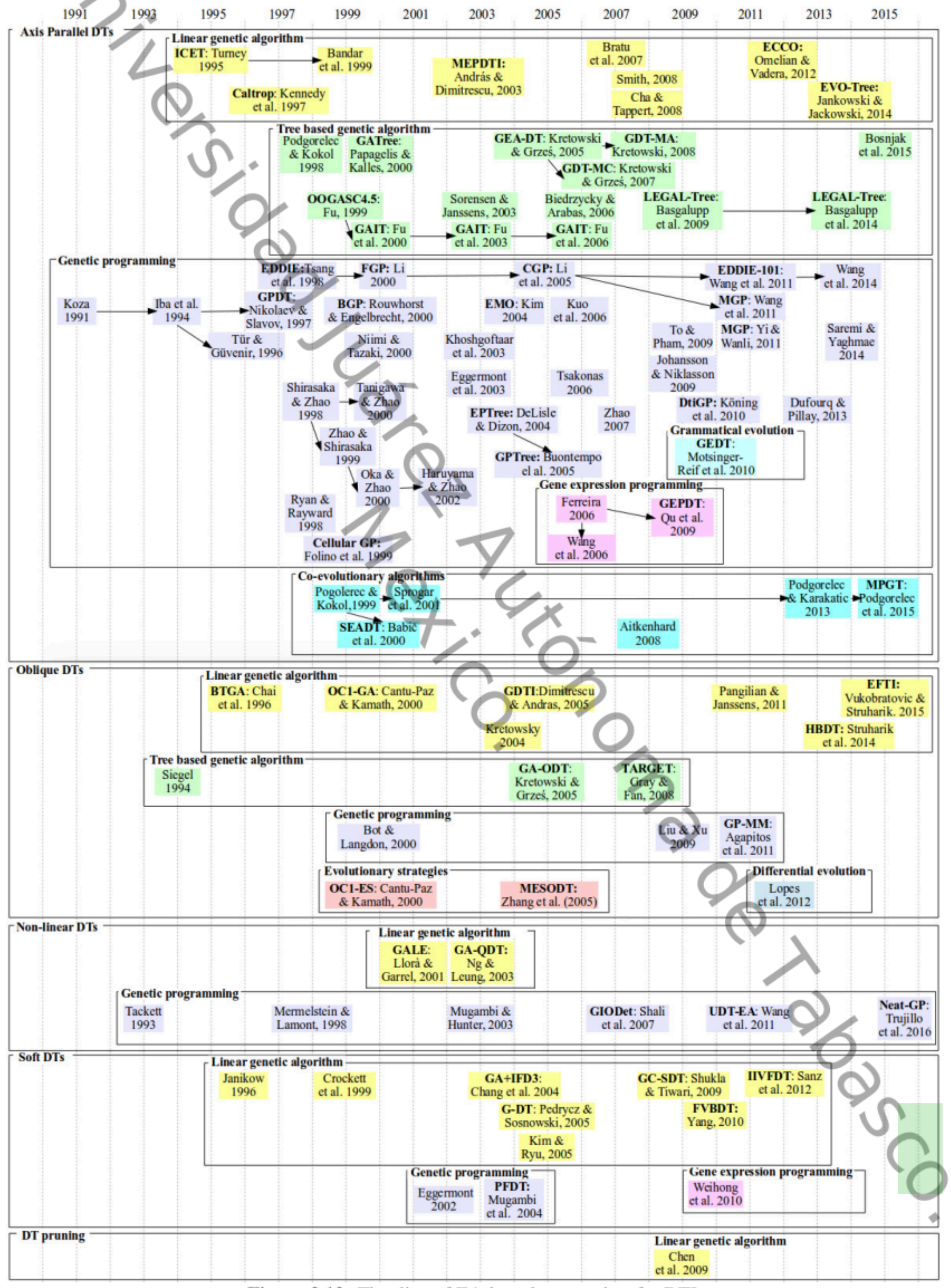


Figure 2.12: Timeline of EA-based approaches for DTI.

Table 2.5: Linear chromosome representation used by EA-based approaches for DTI.

Type of chromosome	Description
Linear chromosomes:	
ChC	A sequence of characters
BinC	A binary sequence encoding integers through the base-2 numeral system
GrayC	A binary sequence encoding integers through the Gray code [31]
IntC	An integer-valued chromosome
RealC	A real-valued chromosome
NumC	A numeric-valued chromosome encoding integers and real numbers
Linear chromosomes for oblique DTs:	
PerpH	A perpendicular hyperplane: A real-valued chromosome encoding a hyperplane which is perpendicular to the segment connecting two instances with different class labels. This hyperplane cuts the segment into two parts with diferent size
BisH	A bisector hyperplane: A perpendicular hyperplane cutting the segment into two equal parts
APH	An axis-parallel hyperplane

Table 2.6: Genetic operators used by EA-based approaches of DTI.

Acronym	Description	Acronym	Description
Selection operators:			
TouS	Tournament-based selection	RWS	Roulette-wheel-based selection
ERS	Exponential-ranking-based selection	LRS	Linear-ranking-based selection
TruS	Truncate-based selection	AssM	Assortative matting
SpeS	Specialized selection		
Crossover for linear chromosomes:			
DPX	Double-point crossover	UnX	Uniform crossover
ArX	Arithmetic crossover	SPX	Single-point crossover
MPX	Multiple-point crossover	SpeX	Specialized crossover
Mutation for linear chromosomes:			
BIM	Bit-interchange mutation	UnM	Uniform mutation
nUM	Non-uniform mutation	SpeM	Specialized mutation

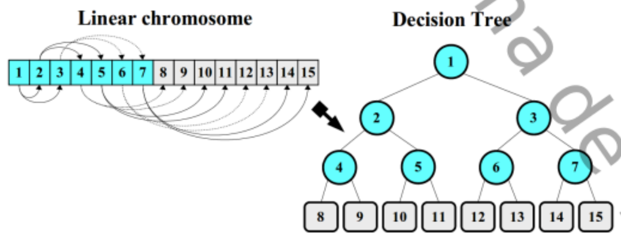


Figure 2.13: Mapping strategy to build a DT from a linear chromosome.

Furthermore, in other GA-based approaches, the chromosome encodes either the nodes of a complete DT, or the elements used to build them: attributes, threshold values of the numerical attributes, and class labels.

In the first case, Cha and Tappert [60, 61] encode both test conditions and leaf nodes in a sequence of integers. Each test condition evaluates a binary attribute represented by the index of its location in an ordered list of attributes, and each leaf node is encoded by an index randomly chosen from a list of class labels. Also, in the method described by Bandar *et al.* [12], each test condition is represented by an index identifying the attribute, either categorical or numerical, used in it. In this algorithm, both the threshold values used in the internal nodes and the class labels assigned to the leaf nodes of the DT are determined by evaluating the training set. Each induced DT is pruned using the χ^2 test to determine the statistical significance of each test condition. A test condition is replaced with a leaf node if its statistical significance is less than a threshold value. Finally, in the Evolutionary Classifier with Cost Optimization (ECCO) method [274] introduced by Omielan and Vadera, a binary chromosome encodes the set of test conditions of a complete DT through an index identifying each possible test condition, to induce multi-branch cost-sensitive DTs.

On the other case, the Evolutionary Algorithm for DTI (EVO-Tree) method, described by Jankowski and Jackowski [174], and the work of Smith [331] implement two similar approaches. Both construct two arrays to encode the elements used by the nodes of a binary DT: one identifying both attributes and class labels, and the other storing the threshold values. In Smith [331], a leaf node is represented with 0 in the first array, and class labels are stored in the second array, and in the EVO-Tree method, a leaf node is represented with a *null* value in the first array and class labels are stored in the second one.

With the aim of showing an example of the GA-based approaches previously described, a complete DT induced using a hypothetical training set with three binary attributes and two class labels is depicted in Fig. 2.14, as well as five linear chromosomes encoding it. This DT has three test conditions and four leaf nodes.

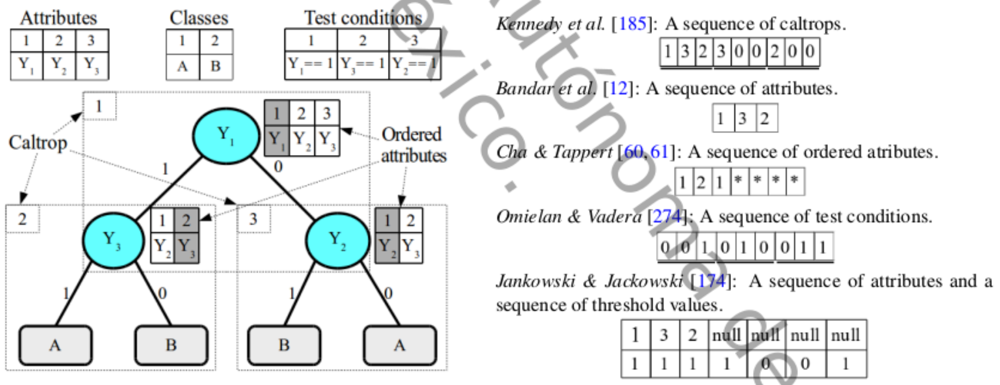

Attributes
1 2 3
Y1 Y2 Y3
Classes
1 2
A B
Test conditions
1 2 3
Y1 == 1 Y3 == 1 Y2 == 1
Caltrop
1
2
3
Ordered attributes
Y1
Y3
Y2
1
0
A
B
B
A
Kennedy et al. [185]: A sequence of caltrops.
1 3 2 3 0 0 2 0 0
Bandar et al. [12]: A sequence of attributes.
1 3 2
Cha & Tappert [60,61]: A sequence of ordered atributes.
1 2 1 * * * *
Omielan & Vadera [274]: A sequence of test conditions.
0 0 1 0 1 0 0 1 1
Jankowski & Jackowski [174]: A sequence of attributes and a sequence of threshold values.
1 3 2 null null null null
1 1 1 1 0 0 1

Figure 2.14: Linear chromosome representation of DTs.

An alternative encoding scheme named Multi-Expression Programming (MEP)[4] is applied by András and Dumitrescu [9] in the MEP-based DTI (MEPDTI) method implementing a global search strategy for DTI. In this method, the test conditions are encoded as functional symbols, and the leaf nodes are represented as terminals.

Although the GA does not encode DTs in the Inexpensive Classification with Expensive Tests (ICET) method described by Turney [361], it is applied to induce cost-sensitive DTs. This method implements a global search strategy evolving a population of biases[5]. In each generation of the ICET method,

[4] MEP [273] defines a linear chromosome composed of variable-length genes. Each gene encodes a terminal or a functional symbol. Functional symbols contain functions that take as arguments the indices of other elements.

[5] In ICET, one bias is defined as the cost of measuring an attribute.

the chromosomes are processed by the EG2 method [271] to build binary DTs. Also, Bratu *et al.* [44] modify this approach by introducing elitism in the population and by adjusting several method parameters.

Oblique DTs: GA-based approaches often use a linear chromosome to encode the hyperplane coefficients used as test condition of one oblique DT that is induced through a recursive partition strategy. Chai *et al.* [62, 63] implement the Binary Tree-Genetic Algorithm (BTGA) encoding the hyperplane coefficients in a binary chromosome. Each hyperplane in the initial population is the perpendicular bisector of two instances with different class labels randomly selected from the training set. Furthermore, the OC1-GA method described by Cantú-Paz and Kamath [58, 59] is an OC1 variant encoding the hyperplane coefficients in a real-valued chromosome. First, it obtains the axis-parallel hyperplane that best divides the training instances, which is copied to 10% of an initial population whose remaining elements are created at random. Then, the OC1-GA method evolves this population to find a near-optimal hyperplane. The induced oblique DT is pruned using the cost-complexity pruning method. In a similar approach, Kr̨etowski [203] uses dipoles[6] representing hyperplanes also encoded with real-valued chromosomes. The initial population has a set of hyperplanes splitting mixed dipoles chosen at random. Pessimistic-error pruning is applied to improve the performance of the induced DT. Also, Pangilinan and Janssens [282] describe a multi-objective GA-based method evolving a population of real-valued chromosomes. This approach uses the selection operators of the Strength Pareto EA (SPEA) algorithm [402] and the Nondominated Sorting GA (NSGA-II) method [82].

Struharik *et al.* [343] use an especial GA named HereBoy algorithm (HBA)[7] in the HereBoy for DT (HBDT) method to find near-optimal hyperplanes. The hyperplane coefficients are encoded in a fixed-length binary chromosome. The induced DT is pruned by a reduced-error-based pruning method. Vukobratovic and Struharik [373] also applies HBA in the Evolutionary Full Tree Induction (EFTI) method to induce a full oblique DT using a variable-length linear chromosome. In the EFTI method, a DT is composed of several internal nodes connected by pointers. Starting with one-node DT randomly created, this method modifies the structure of the DT during its evolutionary process using both topological mutations and hyperplane coefficient mutations.

Unlike the approaches detailed previously, where a recursive partitioning strategy is implemented, a global search in the tree space is conducted in the Generalized Decision Tree Inducer (GDTI) method described by Dumitrescu and J. András [93], in which the MEP encoding scheme is used to represent oblique-DTs.

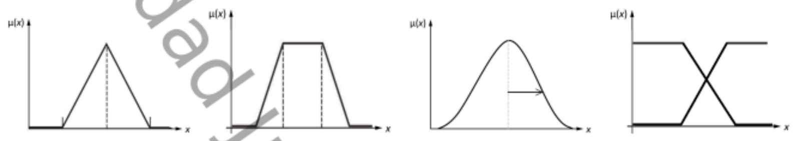
Non-linear DTs: Llorà and Garrell [229–232] and Llorà and Wilson [233] implement the Genetic and Artificial Life Environment (GALE) method, a parallel EA-based approach to evolve a population of DTs placed in a two-dimensional grid. The trees in the population can be axis-parallel, oblique, or non-linear DTs. In particular, non-linear DTs use hyper-spheres in their test conditions. The GALE method uses several specialized operators such as the recombination and the partition operators to modify each DT based on the information of its neighbors, as well as the survival operator to remove DTs with poor performance and to prune the remaining DTs. On the other hand, Ng and Leung [265, 266] introduce the GA-QDT (GA-based Quadratic Decision Tree) method implementing a recursive partitioning strategy to find a near-optimal hypersurface used as test condition of a non-linear DT. Each hypersurface is modeled as $x^T A x + b^T x > \theta$, where x is a vector of attribute values, A is a symmetric matrix, b is a

[6] A dipole is a pair of instances in the training set represented as vectors [34].

[7] HBA [219] works with a unique binary chromosome evolving using a mutation operator.

vector, and θ is the independent term in the inequality. Each chromosome represents a hypersurface by encoding the values of A, b, and θ.

Soft DTs: A GA-based approach commonly used to induce soft DTs is to evolve a population of parameters describing the membership functions associated with the possible result values of the test conditions in a soft DT. Fig. 2.15 shows several types of membership functions used to induce soft DTs through this type of approaches.



(a) Triangular (b) Trapezoidal (c) Gaussian (d) Piecewise linear

Figure 2.15: Membership functions used to induce soft DTs through GA-based approaches.

The GA for Fuzzy ID3 (GA-FID3) method, described by Chang *et al.* [65], and the work of Janikow [173] implements two similar approaches. Both encode in a real-valued chromosome the parameters of the membership functions used in the test conditions of a soft DT, which is induced through a recursive partitioning strategy. Jakinow encodes the corners of several trapezoidal functions, and the mean and the variance of the Gaussian functions, as well as the threshold values used to assign leaf nodes, are encoded in the GA-FID3 method.

The subsequent optimization strategy is implemented in several GA-based approaches to induce soft DTs. Crockett *et al.* [76] describe an algorithm to optimize the piecewise linear membership functions assigned to a DT previously induced by the ID3 algorithm. A real-valued chromosome encodes the bounds of all membership functions used by the soft DT. Pedrycz and Sosnowski [286] implements a very similar approach in the Genetically optimized fuzzy DT (G-DT) method. Starting with a DT previously induced by the C4.5 algorithm, this method evolves a population of binary chromosomes encoding the parameters of the membership functions (piecewise linear or Gaussian) defining the fuzzy regions associated with the test conditions in the soft DT. Furthermore, Sanz *et al.* [316] develop the IIVFDT (Ignorance functions based Interval-Valued Fuzzy Decision Tree with genetic tuning) method to improve the performance of a fuzzy DT previously induced with the Fuzzy-ID3 method [393]. This method encodes the parameters used to weight the ignorance degree[8] for the triangular membership functions of each test condition in a real-valued chromosome.

On the other hand, Kim and Ryu [191, 192] implement a global search strategy to induce fuzzy DTs. A chromosome in this approach encodes the parameters representing the triangular membership functions of all outcomes of the tree internal nodes.

DTI through clustering is also applied to build soft DTs with several GA-based approaches. Shukla and Tiwari [326] implement a recursive partitioning strategy in the Genetically optimized Cluster oriented Soft DTs (GC-SDT) method to find near-optimal test conditions of a soft DT. A binary chromosome encodes a set of prototypes (centroids) of the clusters representing groups of training instances (Fig. 2.16(a)). Furthermore, the Fuzzy Variable-Branch DT (FVBDT) method is implemented by Yang [390]. This method finds the near-optimal number of sub-trees connected to each test condition of a multi-branch DT. The number of occurrences of the bit 1 in a binary chromosome indicates the number of sub-trees attached to the test condition (Fig. 2.16(b)).

[8] Ignorance degree quantifies the uncertainty to assign membership values in fuzzy sets when a classifier is being trained [56].

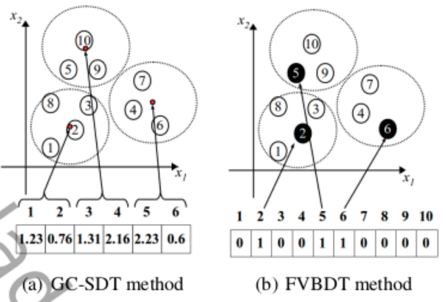


Figure 2.16: Linear chromosome representation for clusters-based DTI.

DT pruning: Chen *et al.* [66] describe a method to prune a previously induced DT by the ID3 algorithm. A binary chromosome represents the test conditions of the DT. In this chromosome, a bit 1 indicates that the test condition, and the sub-tree associated with it, will be removed.

Discussion

The components of the 36 LGA-based approaches for DTI described in the existing literature are shown in Table 2.7. Eight studies implement an aggregating FF, and the remaining studies apply one uni-objective FF. No study reports the use of a multi-objective FF in this type of methods. Eleven studies employ some splitting criterion as their fitness measure, and the test accuracy and the size are the fitness measures most commonly utilized in those implementing either a global search or one subsequent optimization strategy. Some authors describe the genetic operators applying in their works, but others only report to use some GA-library such as GENESIS [146], GENOCOP [246], and GALib [374]. Tournament-based and roulette-wheel-based selection operators have been implemented in eleven and seven studies, respectively. Furthermore, both single-point and double-point crossover, as well as the bit-interchange and the uniform mutation are the genetic operators most commonly applied by this type of GAs, although several authors create ad-hoc genetic operators to ensure the construction of feasible DTs. In particular, both the HBDT method [343] and the EFTI algorithm [373] alter its only one chromosome with a mutation operator. Finally, the initial population in these LGA-based approaches is commonly constructed as a set of chromosomes randomly created. In particular, the OC1-GA method [58] introduces several copies of the best axis-parallel hyperplane found by the OC1-AP method in its initial population.

Table 2.8 resumes 36 experimental studies in the existing literature implementing some LGA-based approach for DTI. In this table is shown that 30 studies induce DTs from several UCI datasets, and 14 methods from other datasets. 20 studies implement one k-fold CV to estimate the classifier performance, and ten studies use a hold-out method. The EVO-Tree method [174] is the only one performing a 5×2 CV. Both the Caltrop method [185] and the procedure described by Cha and Tappert [60, 61] employ the full datasets to build the DTs in their experiments. The test accuracy in 25 studies, the ME in four methods, the tree size in 17 procedures, and the MC in three algorithms have been computed to determine the performance of the LGA-based approaches. In particular, Pangilinan and Janssens [282] utilize the hypervolume as its performance measure. 15 studies report the application of a statistical test to compare the results reached by the LGA-based approaches with those produced by other classifiers. Only three studies apply a post-hoc analysis after finding statistical differences in their experimental results: the Tukey test in the HBDT method [343], and in the EFTI algorithm [373], and the Holm test in the IIVFDT method [316]. Finally,

Table 2.7: Components of LGA-based approaches for DTI.

Stra-tegy	**DT**	**FF**	**Studies**	**Fitness measure**	**Genetic operators**			**Chromosomes in the initial population**
					SEL	**Xover**	**MUT**	
RP	OB	UF	·BTGA [62, 63]	Gini	RWS	DPX	BIM	BisHs randomly created
			·OC1-GA [58, 59]	Twoing	TouS	UnX	N/A	Hyperplanes randomly created with several copies of the best APH found by the OC1-AP method
			·*Krętowski* (2009) [203]	Dipolar	RWS	DPX	SpeM	BisHs randomly created
			·*Pangilinan & Janssens* (2011) [282]	Twoing	TouS	ArX	nUM	Hyperplanes randomly created with one APH also randomly created
			·HBDT [343]	IG	N/A	N/A	BIM	One BisH passing through the centre point of a set of training instances
	NL	UF	·GA-QDT [265, 266]	Gini	RWS	DPX	nUM	Quadric hypersurfaces randomly created
	SF	UF	·*Janikow* (1996) [173]	IG		Based on GENOCOP		RealCs randomly created
			·GC-SDT [326]	SSRE	TouS	DPX	BIM	BinCs randomly created
		AF	·GA-FID3 [65]	Acc∧Size	-	-	-	RealCs randomly created
			·FVBDT [390]	ME∧Size	RWS	DPX	BIM	BisCs randomly created
GS	AP	UF	·ICET [361]	MC		Based on GENESIS		GrayCs randomly created
			·Caltrop [185]	Size	AssM	SPX	UnM	IntCs randomly created
			·*Bandar et al.* (1999) [12]	Acc	-	-	-	IntCs randomly created
			·MEPDTI [9]	Acc	TouS	DPX	SpeM	IntCs randomly created
			·*Bratu et al.* (2007) [44]	MC	RWS	MPX	UnM	GrayCs randomly created
			·*Cha & Tappert* (2008) [60, 61]	Size	-	SPX	UnM	IntCs randomly created
			·ECCO [274]	MC		Based on GENESIS		BinCs randomly created
		AF	·*Smith* (2008) [331]	ME∧Time	-	-	-	IntCs randomly created
			·EVO-Tree [174]	ME∧Size	RWS	SPX	UnM	NumCs randomly created
	OB	UF	·GDTI [93]	Acc	TouS	SpeX	SpeM	IntC randomly created
		AF	·EFTI [373]	Acc∧Size	N/A	N/A	SpeM	A hyperplane of a one-node DT
	NL	UF	·GALE [229–233]	Acc	SpeS	SPX	UnM	IntC randomly created
	SF	AF	·*Kym & Ryu* (2005) [191, 192]	Acc∧Size	RWS	SpeX	SpeM	RealC randomly created
SO	AP	AF	·*Chen et al.* (2009) [66]	ME∧Size	RWS	SPX	BIM	BinCs randomly created
SO	SF	UF	·*Crockett et al.* (1999) [76]	Acc		Based on GENESIS		RealCs randomly created
			·G-DT [286]	ME		Based on GALib		RealCs randomly created
			·IIVFDT [316]	Acc	TouS	UnX	UnM	RealCs randomly created

16 studies compare their results with those obtained by the C4.5 algorithm, and the OC1 method is used in the experiments conducted by six studies inducing oblique DTs, and in seven studies constructing non-linear DTs. In the experiments carried out by Pangilinan and Janssens [282], a procedure to induce axis-parallel DTs is utilized, but the authors do not indicate its name.

Genetic algorithms with tree-based chromosomes (TGA)

Although GA commonly uses linear chromosomes to encode candidate solutions, tree-based representation also has been used to evolve DTs. In this case, a variety of specialized crossover and mutation operators to build valid offsprings have been implemented in several studies found in the existing literature. In this thesis, one representative name is associated with each type of genetic operator to describe these operators. Table

Table 2.8: Experimental analysis reported by LGA-based approaches for DTI.

Strategy	DT	Studies	Datasets		Sampling	Performance	Statistical	Compared methods
			UCI	other	method	measures	tests	
RP	OB	·BTGA [62,63]	1	3	3-f CV	ME, Size	-	TF, PCA, FR
		·OC1-GA [58,59]	10	-	5-f CV	Acc, Size, Time	t-test	CART, OC1, OC1-ES, OC1-SA
		·*Krętowski* (2004) [203]	4	4	10-f CV	Acc, Size, Time	-	OC1, OC1-GA
		·*Pangilinan & Janssens* (2011) [282]	6	1	5-f CV	Acc, Size, HV	-	AP, OC1
		·HBDT [343]	26	-	10-f CV	Acc, Size	ANOVA, Tuk-t	CART, OC1, OC1-AP, OC1-SA, OC1-GA, OC1-ES, GALE, GaTree
	NL	·GA-QDT [265,266]	2	2	10-f CV	Acc, Time	t-test	C4.5, C5.0, OC1, OC1-GA, OC1-ES, BTGA, LMDT, NDT
	SF	·*Jakinow* (1996) [173]	-	1	HO	Acc	-	-
		·GA-FID3 [65]	5	-	-	Acc, Size	-	C4.5, RID3
		·GC-SDT [326]	6	-	5-f CV	ME, Size	t-test	C4.5, C-Fuzzy DT
		·FVBDT [390]	8	2	10-f-CV	Acc	-	MB, C-Fuzzy DT, [301], [391]
GS	AP	·ICET [361]	5	-	HO	MC, Time	-	C4.5, EG2, CS-ID3, IDX
		·Caltrop [185]	-	1	FD	Size	-	ID3
		·Bandar *et al.* (1999) [12]	2	-	HO	Acc	-	-
		·MEPDTI [9]	11	-	HO	Acc	-	C4.5, CN2, BGP
		·*Bratu et al.* (2007) [44]	3	-	HO	MC	-	C4.5, ICET, AdaBoost, MetaCost
		·*Cha & Tappert* (2008) [60,61]	-	1	FD	Size	-	-
		·*Smith* (2008) [331]	-	3	HO	Time	-	-
		·ECCO [274]	4	-	HO	MC	-	ICET
		·EVO-Tree [174]	7	-	5×2-CV	Acc, Size	ARR	C4.5, RTree, NB, MLP, SVM
	OB	·GDTI [93]	11	-	HO	Acc	-	C4.5, CN2, BGP
		·EFTI [373]	26	-	10-f CV	Acc, Size	ANOVA, Tuk-t	CART, OC1, OC1-AP, OC1-SA, OC1-GA, OC1-ES, GALE, GaTree, HDBT
	NL	·GALE [229–233]	11	-	10-f CV	Acc	Avg, t-test	C4.5, CART, OC1
	SF	·*Kym & Ryu* (2005) [191,192]	7	-	-	Acc, Size	Avg	[391], [2].
SO	AP	·*Chen et al.* (2009) [66]	4	-	HO	Acc, Size	-	ID3
	SF	·*Crockett et al.* (1999) [76]	-	2	HO	Acc	-	ID3
		·G-DT [286]	5	-	5-f CV	ME	-	C4.5
		·IIVFDT [316]	17	3	5-f CV	Acc	Avg, Fri-t, Hol-t, Wil-t	C4.5, GA+FID3, [191]

2.9 and Table 2.10 detail these types of crossover and mutation operators. Fig. 2.17 and Fig. 2.18 shown a graphical scheme of each one. Furthermore, the Table 2.11 shows the structure of the internal nodes defined in the studies with this chromosome representation.

Axis-Parallel DTs: Podgorelec and Kokol [291] introduce a GA-based approach implementing a global search strategy for DTI. Papagelis and Kalles [283,284] develop a similar approach named the GATree (Genetically Evolved Decision Trees) algorithm. Kalles and Papagelis [182] describe how the reuse of fitness values previously calculated can improve the evolutionary process speed. Furthermore, Sörensen and Janssens [333] implement an approach to induce complete binary DTs. Fu [129], Fu

Table 2.9: Crossover operators used by tree-based chromosomes.

Acronym	Operator	Description
SSX	Sub-tree swapping	Swaps a sub-tree randomly chosen of a DT and a sub-tree also randomly selected from another DT.
NSX	Node swapping	Swaps a node randomly chosen of a DT and a node also randomly chosen from another DT. The sub-trees of these nodes remain unchanged. This operator can be limited to be applied only between internal nodes, or only between leaf nodes.
BSX	Branch swapping	Swaps a branch randomly chosen of a DT and a branch also randomly selected from another DT.

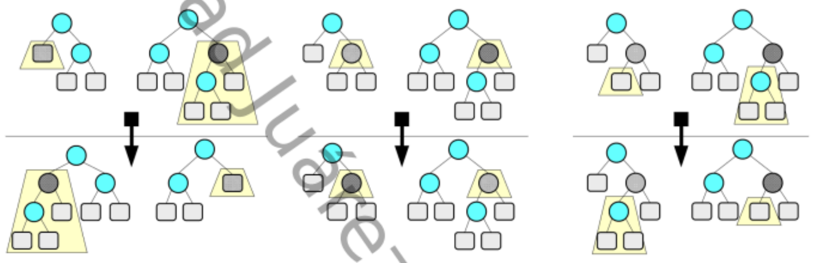


Figure 2.17: Crossover operators used by tree-based chromosomes (Adapted from [283], [333], [132], [206], and [19]).

Table 2.10: Mutation operators used by tree-based chromosomes.

Acronym	Operator	Description
NDM	Node disturbance	Modifies the elements of one node randomly chosen from one DT. This operator can be applied to the attribute or the threshold value of one univariate test condition, with the values of one multivariate test condition, or with the class label of one leaf node.
NSM	Node switching	Swaps two nodes randomly chosen from one DT. The sub-trees of the selected nodes remain unchanged. This operator can be limited to be applied only with test conditions, or just with leaf nodes.
SSM	Sub-tree switching	Swaps two sub-trees from the same DT, which are randomly chosen.
NIM	Node insertion	Adds a new randomly created node (leaf or internal node) to a DT.
NRM	Node replacement	Replaces one node randomly chosen from one DT. A sub-tree can be replaced with a new leaf node, or a leaf node can be replaced with a new sub-tree randomly created.
SRM	Sub-tree replacement	Replaces one sub-tree randomly chosen from one DT with a new sub-tree randomly created.

and Mae [134] and Fu *et al.* [130, 131] describe several versions of the GAIT method. Fu *et al.* [132] implement a probabilistic criterion in the fitness function and, with the aim of introducing diversity in population, two objectives are considered to select DTs that will be part of the next generation. Also, another GAIT version with a fitness function using several percentiles of the accuracy distribution is developed by Fu *et al.* [133].

In the Global EA for DTI (GEA-DT) method, Krętowski and Grześ [206] use the dipolar representation to encode the test conditions of a DT. Krętowski and Grześ [208] use this approach for cost-sensitive classification in the GDT-MC method, and by Krętowski [204] in the GDT-MA method, in which a greedy local search procedure to improve the performance of the induced DTs is applied. Also, in the method described by Biedrzycki and Arabas [29], the search space associated with the training set is initially computed, and one GA is applied to evolve a population of DTs into this space.

Basgalupp *et al.* [21] introduce the Lexicographic Genetic Algorithm for Learning decision Trees

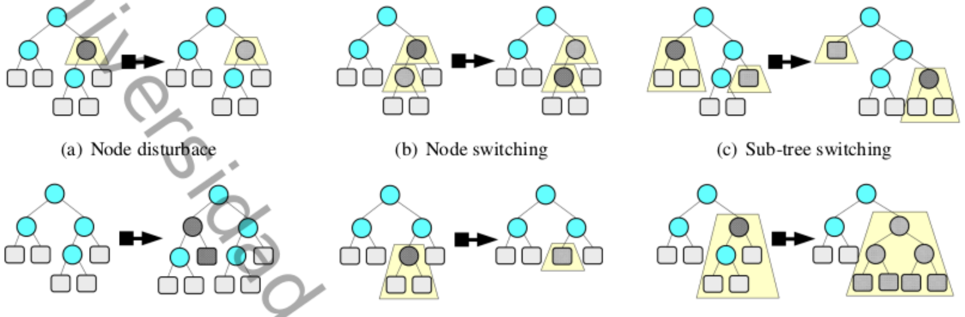


Figure 2.18: Crossover operators used by tree-based chromosomes (Adapted from [283], [333], [132], [206], and [19]).

(LEGAL-Tree), a multi-objective GA-based approach that builds DTs as a collection of decision stumps (DSs)[9]. A lexicographic approach based on tolerance thresholds is applied to select as fitness value either the accuracy or the DT size, using a set of predefined priorities. Basgalupp *et al.* [19] modify the LEGAL-Tree method to include a beam-search-based DTI method that is utilized to build the initial population, and to apply a statistical test in the lexicographic selection instead of a set of predefined tolerance thresholds.

On the other hand, to prevent the premature convergence of the algorithm, Bosnjak *et al.* [40] introduce diversity in the population utilizing a modified selection operator: One DT is selected based on its fitness value, and the other is chosen according to its level of similarity with the first.

Oblique DTs: Siegel [327] describes an approach in which the fitness value of a DT is computed with a subset of training instances. Each instance has a value indicating the number of incorrect predictions in the population. A tournament-based-selection operator is applied to build the subset of instances in a process known as competitive adaptation.

In the Global EA for oblique DTI (GEA-ODT) method described by Kretowski and Grześ [205, 207] a population of mixed DTs is evolved. The GEA-ODT method implements a global search strategy to find the structure of a DT as well as the hyperplane coefficients used in the test conditions. Furthermore, Gray and Fan [145] modify their regression tree induction approach, known as TARGET [144], to induce near-optimal oblique DTs. TARGET implements elitism and immigration (new randomly created DTs are introduced in population) in its evolutionary process.

Discussion

The components of the TGA-based approaches for DTI previously described are shown in Table 2.12. Both uni-objective and aggregating FFs have been used in ten studies each one, and a multi-objective FF that implement a lexicographic ordering criterion is utilized in the LEGAL-Tree method [21]. Test accuracy and size have been employed as fitness measures in 18 and 12 studies, respectively. Roulette-wheel-based selection is used in seven studies, and both tournament and linear-ranking-based selection operators have been implemented in five studies each one. All TGA-based approaches for DTI apply the sub-tree-swapping-based crossover operator, but both the node-swapping and the branch-swapping-based crossover operators

[9] A decision stump is a DT with one test condition and two leaf nodes

Table 2.11: Structure of the internal nodes encoded by one tree-based chromosomes.

Study	Branching type	Internal node
Axis-parallel DT:		
· GAIT [129–134]	Binary	A numerical attribute is compared with a threshold value produced by the C4.5 method.
· GATree [283, 284]	Binary	One categorical attribute is compared with only one possible value. A numerical attribute is compared with a threshold value (this value is modified by the mutation operator).
· *Sörensen & Janssens* (2003) [333]	Binary	Only binary attributes are evaluated.
· GEA-DT [206], GDT-MC [208], GDT-MA [204]	Multi-branch	A numerical attribute is compared with a threshold value (this value can be mutated). A branch is associated with each possible value of a categorical attribute.
· *Biedrzycki & Arabas* (2006) [29]	Multi-branch	A branch is associated with each possible value of each attribute.
· LEGAL-Tree [19, 21]	Multi-branch	A numerical attribute is compared with a threshold value defined using the IG criterion. A branch is associated with each possible value of a categorical attribute.
Oblique DT:		
· *Siegel* (1994) [327]	Multi-branch	Several categorical attributes can be used in one test condition, and a branch can be associated with several attribute values.
· TARGET [144], GEA-ODT [205]	Binary	A linear combination of attributes is used in one test condition.
· GDT-Mix [207]	Multi-branch	Test conditions can use one or more attributes. For a univariate test condition, if a numerical attribute is used, it is compared with a threshold value. Otherwise, a sub-tree is associated with each value of a categorical attribute. For a multivariate test condition, a linear combination of attributes is used in the test condition.

also have been used in seven and five studies, respectively. Furthermore, several mutation operators have been implemented with this type of EAs. In particular, the method described by Biedrzycki and Arabas [29] does not apply a crossover operator, and it only uses a node-insertion-based mutation to alter the initial DTs. Finally, the random creation of chromosomes is the most common scheme used to build the initial population of this type of approaches, although other procedures also have been utilized. In particular, the GAIT method [130] uses the C4.5 method with several instances randomly chosen from the training set to build its initial population, and the LEGAL-Tree method [21] generates it based on a random combination of several decision stumps previously created.

Table 2.13 shows the experimental studies reported in the existing literature implementing some TGA-based approach for DTI. From this table is observed that 15 studies induce DTs using UCI datasets, and 16 studies utilize other datasets. 12 studies implement one k-fold CV, and nine studies apply a hold-out sampling strategy to compute the classifier performance. Test accuracy is calculated for 19 TGA-based methods, 16 studies estimate the size of the induced DTs, and two algorithms determine the misclassification error. Both the technique described by Bosnjak *et al.* (2015) [40] and the LEGAL-Tree method adopt the F-measure as their performance measure, and only the GDT-MC method [208] implements a cost-sensitive classification approach. Ten studies compare their experimental results with those obtained by the C4.5 method. In particular, Podgorelec and Kokol [291] implement a traditional DTI method, without specifying its name, and the results obtained by the GDT-MA method [204] have been compared with a basic evolutionary version of it, named the GDT-AP method by its authors. Finally, nine studies describe the application of a statistical test to compare their experimental results with those obtained by other approaches, and only the LEGAL-Tree method applies one post-hoc analysis.

Table 2.12: Components of TGA-based approaches for DTI.

Strategy	DT	FF	Studies	Fitness measure	SEL	Genetic operators: Xover	Genetic operators: MUT	Chromosomes in the initial population
GS	AP	UF	·OOGASC4.5 [129]	Acc	-	SSX	NSM	DTs induced by the C4.5 method with a subset of randomly chosen training instances
			·GAIT [130–134]	Acc	RWS	SSX	SSM	DTs induced by the C4.5 method with a subset of randomly chosen training instances
			·*Sörensen & Janssens* (2003) [333]	Acc	RWS	SSX, NSX	NSM, SSM	Complete binary DTs randomly created
			·*Bosnjak et al.* (2015) [40]	Acc, Size, FM	TouS	SSX	NDM	DTs randomly created
		AF	·*Podgorelec & Kokol* (1998) [291]	ME∧Size	ERS	SSX	NDM	DTs randomly created
			·GATree [182, 283, 284]	Acc∧Size	-	SSX	NDM	DSs randomly created
			·GEA-DT [206]	Acc∧Size	LRS	SSX, NSX, BSX	NDM, NSM, NRM, SSM	DTs randomly created
			·*Biedrzycki & Arabas* (2006) [29]	ME∧Size	TouS	N/A	NIM	Empty DTs
			·GDT-MC [208]	MC∧Size	LRS	SSX, NSX, BSX,	NDM, NSM, NRM, SSM	DTs randomly created
			·GDT-MA [204]	Acc∧Size	LRS	SSX, NSX, BSX	NDM, NSM, NRM, SSM	DTs induced with 10% of the training set using several splitting criteria (IG, GR, Gini and Dipolar)
		MF	·LEGAL-Tree [19, 21]	Acc, Size	TouS	SSX	NRM	DTs created with a random combination of several DSs previously induced with 10% of the training set
	OB	UF	·*Siegel* (1994) [327]	Acc	TouS	SPX, SSX	N/A	-
			·TARGET [145]	ME	RWS	SSX, NSX	NDM, NSM	DTs randomly created
		AF	·GEA-ODT [205, 207]	Acc∧Size	LRS	SSX, NSX, BSX	NDM, NSM, NRM, SSM	DTs randomly created

Genetic programming (GP)

In DTI through GP, internal and leaf nodes can contain different elements and they can have distinct meanings. Table 2.14 describes the structure of the nodes used in the GP-based approaches for DTI. The ramped half-and-half (RH&H) criterion described by Koza [202] is the most used procedure to build an initial population of candidate solutions. RH&H creates trees having a wide variety of sizes and shapes based on a variable height that ranges between two and one maximum value previously specified. The sub-tree-swapping-based crossover operator and the sub-tree-replacement-based mutation operator are typically applied by the GP algorithm, although the node-perturbation operator is also used.

Axis-Parallel DTs: Koza [200] proposes to encode DT with S-Exps: the internal nodes as functions and the leaf nodes as terminals. These expressions are used by Iba *et al.* [171] and by Tür and Güvenir [360]. Function and terminal sets are also used to encode DTs by Tsang *et al.* [357] in the Evolutionary Dynamic Data Investment Evaluator (EDDIE) system, by Wang *et al.* in both the Memetic GP (MGP) algorithm [375] and in the EDDIE-101 approach [377], and in the work of Niimi and Tazaki [267]. In particular, the EDDIE system encodes its leaf nodes as terminals with indicators of financial forecasting, and in both the MGP algorithm and the EDDIE-101 approach the class label in a leaf node is replaced by one record of instances reaching the node, grouped by their class label. Also, the EDDIE-101 approach implements a local search to improve the threshold values of each test condition, and

Table 2.13: Experimental analysis reported by TGA-based approaches for DTI.

Stra-tegy		**DT Studies**	**Datasets UCI**	**other**	**Sampling method**	**Performance measures**	**Statistical tests**	**Compared methods**
GS	AP	· *Podgorelec & Kokol* (1998) [291]	-	1	HO	Acc, Size	-	Traditional DTI method
		· OOGASC4.5 [129]	-	1	HO	Acc, Time	-	SampleC4.5
		· GATree [283, 284]	13	5	5-f CV	Acc, Size	Avg	C4.5, 1R
		· GAIT [130, 134]	-	1	HO	Acc, Time	-	C4.5
		· *Sörensen & Janssens* (2003) [333]	-	1	-	Acc	-	-
		· GAIT [131]	-	1	HO	Acc, Time	t-test	LR
		· GAIT [132]	1	2	HO	Acc, Size, Time	t-test	C4.5
		· GEA-DT [206]	8	3	10-f CV	Acc, Size	-	C4.5
		· GAIT [133]	3	2	HO	Acc, Size	t-test	C4.5
		· *Biedrzycki & Arabas* (2006) [29]	4	2	10-f CV	ME, Size	-	ID3, J48
		· GDT-MC [208]	13	-	10-f CV	MC, Size	Avg	C5.0, CSJ48, MetaCost
		· GDT-MA [204]	15	-	10-f CV	Acc, Size	-	C4.5, GDT-AP
		· LEGAL-Tree [21]	6	-	10-f CV	Acc, Size	t-test	J48
		· GATree [182]	12	-	5-f CV	Acc, Size, Time	-	J48
		· LEGAL-Tree [19]	16	-	10-f CV	Acc, FM, Size, Time	Fri-t, Nem-t	J48, CART, GALE
		· *Bosnjak et al.* (2015) [40]	20	-	HO	Acc, Size, FM	KW-t	GATree
	OB	· *Siegel* (1994) [327]	-	1	HO	Acc	-	-
		· GEA-ODT [205, 207]	10	5	10-f CV	Acc, Size	-	C4.5, OC1
		· TARGET [145]	3	2	10-f CV	ME, Size	-	CART, RPART, QUEST, CRUISE, RF

Niimi and Tazaki [267] construct the initial population of DTs utilizing the Apriori algorithm [4].

A simple scheme to represent DTs is applied in the Evolutionary Multi-objective optimization (EMO) approach described by Kim [188, 189]. EMO is based on the multi-objective GA (MOGA) non-dominated rank method introduced by Fonseca and Fleming (1993) [119]. Furthermore, in the multi-objective GP (MOGP) method detailed by Wang *et al.* [376], a population of DTs evolves to find Pareto-optimal solutions. This method implements four multi-objective methods based on the NSGA-II method [82], the multi-objective evolutionary algorithms based on decomposition (MOEA/D) method [397], the approximation-guided evolutionary multi-objective (AGEMOA) method [48], and the multi-objective selection based on dominated hypervolume (SMS-EMOA) method [27], respectively.

Some GP-based methods introduce specialized operators to build feasible offsprings. In the GP for DT (GPDT) method described by Nikolaev and Slavov [268, 330], a mutation operator implements a depth-first search strategy to alter each visited node based on a mutation probability, and Ryan and Rayward-Smith [314] applies a modified crossover operator in which each offspring is refined using the training set. Furthermore, Kuo *et al.* [210–213] uses one operator to eliminate redundant sub-trees, and other to remove subsumed sub-trees, and Dufourq and Pillay [92] use an encapsulation operator to preserve sub-trees in the population. Finally, a migration operator is applied by Folino *et al.* [117] to periodically swap DTs placed in the cells of a bidimensional grid in the Cellular GP method, and by To and Pham [352] to exchange chromosomes between sub-populations evolving in an island-based parallel GP.

Several modifications in GP-based approaches for DTI have been implemented with the aim to avoid and to improve the GP performance. König *et al.* [197] introduce an adaptive fitness function, to include diversity in a population of DTs and for controlling the DT size, in the DT Injection GP (DTiGP) method. First, a fitness value of a DT induced with a subset of instances randomly chosen from the training set is designated as a reference value. Next, the evolutionary process begins and, at even intervals, the fitness value of the best DT in the population is compared with the reference value. If its fitness value is less than the reference value, the fitness function is modified to allow larger DTs. Furthermore, Yi and Wanli [392] implement the multiage GP (MGP) method to evolve groups of DTs according to their number of leaf nodes (ages). The MGP method first divides the population of DTs using their ages and then evolves each group independently. Finally, these evolved groups are combined to build a new population that will be used in the next iteration of the method. This division allows to limit the selection pressure in a particular area (group) and tries to prevent that genetic operators destroy the continuity of the evolutionary process.

Other GP-based methods for DTI build their trees without using the function and terminal sets. Shirasaka *et al.* [325] define a DT as a list of nodes each one with references to one predecessor node and two successor nodes. To induce more compact DTs, Zhao and Shirasaka [400] evaluate several alternatives such as controlling the DT size in the selection process and deleting redundant nodes and sub-trees. Oka and Zhao [272] apply the C4.5 method with segments of the training set to build the initial population. Tanigawa and Zhao [348] employ GP to induce small sub-trees that combine to build a complete DT. Furthermore, Haruyama and Zhao [152] implement three multi-objective methods based on the vector evaluated GA (VEGA) method [319], the Pareto ranking algorithm [143], and the Niched-Pareto GA (NPGA) method [164], respectively. Furthermore, DeLisle and Dixon [84] in the Evolutionary Programming Tree (EPTree) as well as Buontempo *et al.* [53] and Wang *et al.* [380] encode a DT as a linked structure. Rouwhorst and Engelbrecht [311] and Engelbrecht *et al.* [101] implement the Building Block approach for GP (BGP) in which a DT is encoded with *building blocks* representing test conditions. Starting with a population of decision stumps, the BGP method adds new nodes in each one. BGP also applies one operator to prune the evolved DTs. A very similar encoding scheme called *full atomic representation* is described by Eggermont *et al.* [99, 100] in which a population of DTs evolves through a multi-layered fitness function consisting of two ranked fitness measures. The authors first utilize an information-theory-based splitting criterion [99], and then they develop a refined atomic representation including minimum and maximum values for numerical attributes [100]. The bounds of each numerical attribute in the training set are determined with a k-means clustering method [337].

Additionally, discretization of real-valued attributes also has been implemented in several GP-based approaches for DTI: The GP Evolved Intervals (GPEI) method described by Dufourq and Pillay [91] includes an adaptive discretization with both fixed and varying number of intervals. Also, Saremi and Yaghmaee [317] implement specialized operators modifying threshold values used to divide the attribute values.

On the other hand, GGP and TGP have been implemented in several DTI approaches. The Financial Genetic Programming (FGP) method described by Li [220] and by Tsang and Li [356] represents DTs through a BNF grammar. In the Constrained GP (CGP) method, Li *et al.* [222] implements cost-sensitive classification using the same representation. Tsakonas [355] uses a similar grammar to describe DTs in his Grammar-guided GP (G^3P) method. Moreover, Johansson and Niklasson [177] and Johansson *et al.* [178] find near-optimal DTs in a GGP-based method using an oracle data[10], to improve the performance of their induced DTs. Also, Johansson *et al.* [176] describe several strategies to select a candidate solution in a population of DTs.

Khoshgoftaar *et al.* [187] and Khoshgoftaar and Liu [186] implement a TGP in a multi-objective

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

12

method to evolve a population of DTs. Khoshgoftaar *et al.* [187] use a method based on the Constraint Oriented Transformation approach [102] in which an objective is initially chosen to be minimized and the other objectives are transformed into constraints. Khoshgoftaar and Liu [186] apply a NSGA-based method [336] to optimize MCs, resource availability, and DT size. Crossover and mutation operators are applied considering the constraints impossed by data types. Furthermore, Zhao [399] uses the TGP for DTI in a MOGA-based approach to implementing an application allowing for the decision maker to specify partial preferences on conflicting objectives such as false negative versus false positive, sensitivity versus specificity, and recall versus precision.

Oblique DTs: Bot and Langdon (2000) [41,42] apply TGP to encode the hyperplanes used as test conditions in an oblique DT. They analyze the use of one limited error fitness [140] to reduce the evolution time, as well as the implementation of two multi-objective methods: the NPGA method, and the Pareto scoring with fitness sharing algorithm [143]. Furthermore, Liu and Xu (2009) [228] apply GP to induce DTs from multiclass datasets. The chromosomes are grouped in sub-ensembles evolving independently. Each sub-ensemble tries to solve a two-class problem from the dataset. The better DTs in the sub-ensembles are combined to build a final multiclass DT. Several methods to solve the problems arising in the fusion of sub-ensembles are implemented in this work, as well as a greedy algorithm to keep high diversity in the sub-ensembles. Finally, a GGP-based method to find near-optimal hyperplanes is implemented by Agapitos *et al.* (2011) [3] in the GP hybridized with Margin Maximisation (GP-MM) method. This method also uses a (1+1)-ES to maximize the margins associated with the hyperplane.

Non-linear DTs: Shali *et al.* [323] in the GP for Induction of Oblique Decision Trees (GIODeT) method, and Marmelstein and Lamont [241] describe 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.

On the other hand, a global search of non-linear DTs also has been implemented in several GP-based approaches. The representation scheme introduced by Koza [200] is used by Tackett (1993) [345]. Furthermore, the Unconstrained Decision Trees EA (UDT-EA) method implemented by Wang *et al.* [377] induces non-linear DTs with hypersurfaces as test conditions, and Mugambi and Hunter [255] use TGP in a multi-objective approach that evolves a population of DTs to find Pareto optimum values. Finally, Trujillo *et al.* [354] adapt the NeuroEvolution of Augmenting Topologies (NEAT) algorithm [329] in the neat-GP method to maintain the distribution of the size of the trees in the population close to one uniform distribution.

2

Soft DTs: Eggermont [98] 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.* [256] use SGP in the Polynomial-Fuzzy DTs (PFDT) method to find near-optimal hypersurfaces of a non-linear DT. The PFDT method is based on a multi-objective GP [309] that evolves polynomials representing the hypersurfaces, as well as sigmoid and bell-shaped membership functions associated with the numerical attributes of the dataset.

Discussion

Table 2.15 shows the components of GP-based approaches for DTI. 43 studies in the existing literature implement a standard GP, eight studies describe the construction of DTs with one grammatical-based GP, and the strongly-typed GP is applied in seven studies. In this Table can be observed that 25 studies implement an uni-objective FF, 14 use an aggregating FF, and the remaining studies evaluate a multi-objective FF.

Table 2.14: Structure of internal and leaf nodes used in GP-based approaches for DTI.

Study	**Internal node**	**Leaf node**
Axis-parallel DT:		
·*Koza* (1991) [200]	An attribute as a function with several arguments	A class label
·*Iba et al.* (1994) [171], *Tür and Güvenir* (1996) [360]	An attribute as a boolean function	A class label
·*Nikolaev & Slavov* (1997) [268], *Slavov & Nikolaev* (1998) [330]	A categorical attribute or a discretized numerical attribute	A class label
·*Ryan & Rayward-Smith* (1998) [314]	A categorical attribute	A class label
·EDDIE [357], FGP [220, 356]	$\{$*if-then-else*$, \wedge, \vee, \neg, >, <\}$	An attribute, a class label or a numerical value
·*Kuo et al.* (2006, 2007, 2008) [210–213]	$\{$*if-then*, *if-then-else*$, \wedge, \vee, \neg, >, <, \geq, \leq\}$	A class label or an attribute
·*Khoshgoftaar et al.* (2003) [187], *Khoshgoftaar & Liu* (2007) [186]	$\{$*if*$, <\}$, or a class label	An attribute or a numerical value
·*Zhao* (2007) [399]		An attribute, a class label, or a numerical value
·*Johansson & Niklasson* (2009) [176, 177], *Johansson et al.* (2011) [178], DTiGP [197]	$\{$*if*$, >, <, =\}$	An attribute, a class label, or a numerical value
·MGP [375], EDDIE-101 [377], MOGP [376]	$\{$*if-then-else*$, \wedge, \vee, \neg, >, <, =\}$	1 representing *positive* and 0 representing *negative*.
·*Shirasaka et al.* (1998) [325], *Zhao & Shirasaka* (1999) [400], *Oka & Zhao* (2000) [272], *Tanigawa & Zhao* (2000) [348], *Haruyama & Zhao* (2002) [152]	Each node is a 7-tuple $\{t,\ label,\ P,\ R,\ C,\ size\}$ where t is the node number, *label* is the class label of a leaf node. P is a pointer to the parent. L and R are the pointers to the left and the right children. C is a set of counters. For an internal node, $n = C[0]$ and $a = C[1]$, and a decision is made based on the comparison ($attribute_n < a$). For a leaf node, $C[i]$ is the number of training samples of the i-th class, which are classified to this node. The class label assigned is determined by majority voting. *size* is the size of node when it is considered as a sub-tree.	
·BGP [101, 311]	A building block $\langle a, o, a \vert t\rangle$ where a is an attribute, t is a threshold value, and $o \in \{=, \neq, <, \leq, >, \geq\}$ is a mathematical operator.	
·*Niimi & Tazaki* (2000) [267]	A conditional sentence.	An attribute or a class label
·*Eggermont et al.* (2003, 2004) [99, 100]	A categorical attribute, or a numerical attribute compared with a threshold value	A class label
·EPTree [84], GPTree [53, 380], *To & Pham* (2009) [352], EMO [188, 189]	A numerical attribute compared with a threshold value	A class label
Oblique DT:		
·*Bot & Langdon* (2000) [41, 42]	A label indicating the number of attributes in the linear combination (1, 2 or 3)	A class label, a numerical value or an integer representing an attribute
·*Liu & Xu* (2009) [228]	$\{>, \leq, *, -, +, \max, \min\}$	An attribute or a numerical value
Non-linear DTs:		
·*Tackett* (1993) [345]	$\{$ *if-then-else*$, +, -, *, \%\}$	An attribute or a numerical value
·*Marmelstein & Lamont* (1998) [241]	$\{+, -, \times, \div, \leq\}$	An attribute or a numerical value
·GIODeT [323]	$\{+, -, *, /, \ln, \surd, \leq, \wedge, \vee\}$	An attribute or a numerical value
·UDT-EA [377]	$\{$*if-then-else*$, +, -, *, \wedge, \vee, \neg, >, <, \geq, \leq, =, \leq\}$	An attribute, a class label or a numerical value
·neat-GP [354]	$\{+, -, \times, \div, \sin, \cos, \exp, \surd, x^y, \vert x \vert,$ *if* $\}$	An attribute
Soft DTs:		
·fuzzy GP [98]	A fuzzified numerical attribute, or a categorical attribute	A class label

The test accuracy and the DT size are the fitness measure most commonly applied to these methods: Test accuracy in 32 studies, and the DT size in 28 works. Also, the misclassification cost is assessed in 12 studies. The tournament-based selection is the prominent selection operator for this type of methods since it is applied in 38 studies. Furthermore, sub-tree swapping and sub-tree replacement are the crossover and mutation operators most commonly implemented in these approaches. Other mutation operators such as node disturbance and node replacement also have been applied to alter the structure of one induced DT. The random creation of the initial population and the ramped half-and-half method are applied to generate the initial candidate solutions in 25 and 23 studies, respectively. Some studies apply alternative strategies to create the initial population: to use the C4.5 method with a collection of instances randomly chosen from the training set [272], to create decision stumps [311], and to map DTs from the rules created with the Apriori algorithm [267].

Table 2.16 shows the elements of the experimental studies carried out by the GP-based approaches for DTI previously described. In this table is shown that 33 studies induce DTs from several UCI datasets and 31 studies use datasets from other sources. 24 studies implement one k-fold CV, and 30 studies apply a hold-out sampling strategy. Shirasaka *et al.* [325], To and Pham [352] and Zhao [399] use the full datasets to compute the performance of their algorithms. The method described by Liu and Xu [228] is the only one implementing the LOOCV as its sampling method. Test accuracy, size, and ME are adopted as performance measures in 38, 18 and 11 studies, respectively. The MGP method [375] and the MOGP algorithm [376], as well as the study described by Zhao [399], estimate the AUC value. Also, MC, Sen, Spe, Fid, and the ROC curve analysis have been utilized as performance measures in several GP-based approaches for DTI. Furthermore, 11 studies carried out a statistical test to compare their experimental results with those obtained from other methods, and four studies implement some post-hoc analysis: the Nemenyi test is used in the studies described by Johansson and Niklasson [177], Johansson *et al.* [176] and Johansson *et al.* [178], and the Bonferroni-Dunn test in the studies of Johansson and Niklasson [177], Johansson *et al.* [178], and Trujillo *et al.* [354]. Finally, the results produced by the C4.5 method and the J48 algorithm have been contrasted with those obtained by 24 and six studies, respectively. The results of the NB method are compared with those obtained by six GP-based approaches, and the results of the SVM method with those reported in two studies. Also, the results of several ANN-based methods are confronted with the results of four GP-based methods, and the results obtained by the kNN algorithm with those calculated by the CGP method [222]. In particular, the MGP method implemented by Wang *et al.* [375] is compared with the GGP (G-mean GP) and EGP (Entropy GP) methods. These methods are variants of the MGP method developed by the same authors.

Other EA-based approaches for DTI

Evolutionary strategies: Cantú-Paz and Kamath [58, 59] develop the OC1-ES algorithm, and Zhang *et al.* [396] introduce the Multimembered ES Oblique DT (MESODT) method to obtain a near-optimal hyperplane using the $(1+1)$-ES and the (μ,λ)-ES, respectively. The OC1-ES algorithm modifies the best hyperplane found by the OC1-AP method, and the MESODT method creates an initial population based on a hyperplane constructed by an ANN-based approach described by Unnikrishnan and Venugopal [363]. In both methods, an oblique DT is induced in a recursive partitioning strategy, and it is pruned with the cost-complexity pruning method.

Co-evolutionary algorithms: Several versions of a competitive-CEA-based method are described by Podgorelec and Kokol [292–294], Babič *et al.* [11], and Sprogar *et al.* [335] applying a tree-based chromosome representation with GA to build axis-parallel DTs using three independent populations competing to find a near-optimal DTs. One population is a single DT induced by the C4.5 method, and the

Table 2.15: Components of GP-based approaches for DTI.

Strategy	DT	FF	MH	Studies	Fitness measure	Genetic operators SEL	Xover	MUT	Chromosomes in the initial population
RP	NL	AF	GP	·*Marmelstein & Lamont* (1998) [241]	ME∧Size	TouS	SSX	SRM	S-Exps randomly created
				·GIODeT [323]	GR∧Size	TouS	SSX	SRM	ExpTs randomly created
GS	AP	UF	GP	·*Koza* (1991) [200]	Acc	TouS	SSX	SRM	-
				·*Iba et al.* (1994) [171]	MDL	TouS	SSX	NRM, SRM	S-Exps randomly created
				·GPDT [268, 330]	MDL	RWS	SSX	SpeM	DTs randomly created
				·EDDIE [357]	Acc	TouS	BSX	SRM	-
				·*Shirasaka et al.* (1998) [325]	Acc	TruS	SSX	NDM	-
				·Cellular GP [117]	ME	SpeS	SSX	SRM	DTs randomly created
				·*Zhao & Shirasaka* (1999) [400]	Acc	TruS	SSX	NDM	DTs randomly created
				·*Oka & Zhao* (2000) [272]	Acc	TruS	SSX	NDM	DTs created using C4.5
				·*Tanigawa & Zhao* (2000) [348]	Acc	TruS	SSX	NDM	DTs randomly created
				·BGP [101, 311]	Acc	TouS	SSX	NDM	DSs randomly created
				·*Niimi & Tazaki* (2000) [267]	Acc	TouS	SSX	SRM	DTs created using Apriori
				·EPTree [84]	MDL	TouS	SSX	NDM, SRM	DTs randomly created
				·GPTree [53, 380]	Acc	TouS	SSX	NDM, SRM	DTs randomly created
				·*To & Pham* (2009) [352]	Acc	TruS	SSX	N/A	DTs randomly created
				·GPEI [91, 92]	Acc	TouS	SSX	SRM	RH&H criterion
			GGP	·FGP [220, 356]	Acc	TouS	SSX	SRM	RH&H criterion
				·*Johansson et al.* (2010) [176]	Acc	RWS	SSX	SRM	RH&H criterion
		AF	GP	·*Tür & Güvenir* (1996) [360]	Acc∧Size	-	SSX	SRM	DTs randomly created
				·*Ryan & Rayward-Smith* (1998) [314]	ME∧Size	-	SpeX	N/A	DTs created using C4.5
				·*Kuo et al.* (2006–2008) [210–213]	Acc∧Size	RWS	SSX	SRM	DTs randomly created
				·DTiGP [197]	ME∧Size	RWS	SSX	SRM	RH&H criterion
				·MGP [392]	Acc∧Size	-	-	-	DTs randomly created
				·MGP [375]	IG∧Size	TouS	SSX	SpeM	RH&H criterion
				·EDDIE101 [377]	IG∧Size	TouS	SSX	SRM	RH&H criterion
				·*Saremi & Yaghmaee* (2014) [317]	Acc∧Size	TouS	SSX	NDM, SRM	DTs randomly created
			GGP	·CGP [222]	Acc∧FPR	TouS	SSX	SRM	RH&H criterion
				·G^{3}P [355]	Acc∧Size	TouS	SSX	SRM	DTs randomly created
				·*Johansson & Niklasson* (2009) [177], *Johansson et al.* (2011) [178]	Acc∧Size	RWS	SSX	SRM	RH&H criterion
		MF	GP	·*Haruyama & Zhao* (2002) [152]	Acc, Size	TouS	SSX	NDM	DTs randomly created
				·*Eggermont et al.* (2003) [99, 100]	ME, Size	TouS	SSX	SRM	RH&H criterion
				·EMO [188, 189]	ME, Size	TouS	SSX	NDM, SRM, NRM	DTs randomly created
				·MOGP [376]	Sen, FPR	TouS	SSX	SpeM	RH&H criterion
			TGP	·*Khoshgoftaar et al.* (2003) [187], *Khoshgoftaar & Liu* (2007) [186]	ME, Size	TouS	SSX	SRM	RH&H criterion
				·*Zhao* (2007) [399]	FPR, FNR, Sen, Spe, P	TouS	SSX	SRM	DTs randomly created
	OB	UF	GP	·*Liu & Xu* (2009) [228]	Acc	RWS	SSX	SRM	RH&H criterion
			GGP	·GP-MM [3]	Acc	TouS	N/A	SRM	RH&H criterion
		AF	TGP	·*Bot & Langdon* (2000) [41]	Acc∧Size	TouS	SSX	SRM	RH&H criterion
		MF	TGP	·*Bot & Langdon* (2000) [42]	ME, Size	TouS	SSX	SRM	RH&H criterion
	NL	UF	GP	·neat-GP [354]	ME	TouS	SSX	SRM	RH&H criterion
		AF	GP	·UDT-EA [377]	IG∧Size	TouS	SSX	SpeM	RH&H criterion
		MF	GP	·*Tackett* (1993) [345]	ME	RWS	SSX	SRM	S-Exps randomly created
			TGP	·*Mugambi & Hunter* (2003) [255]	Sen, Spe, Size	TouS	SSX	SRM	DTs randomly created
SF	MF	GP		·Fuzzy-GP [98]	Acc, Size	TouS	SSX	SRM	RH&H criterion
		TGP		·PFDT [256]	Sen, Spe, Size	TouS	SSX	SRM	DTs randomly created

Table 2.16: Experimental evaluation reported by GP-based approaches for DTI.

Strategy	DTMH		Studies	Datasets UCI	Datasets other	Sampling method	Performance measures	Statistical tests	Compared methods
RP	NL	GP	· *Marmelstein & Lamont* (1998) [241]	2	1	HO	ME	-	RBF-NN, MLP, GP
			· GIODeT [323]	19	-	5-f CV	Acc, Size	Avg	C4.5
GS	AP	GP	· *Koza* (1991) [200]	-	-	-	-	-	-
			· *Iba et al.* (1994) [171]	-	2	HO	Acc	-	ANN
			· *Tür & Güvenir* (1996) [360]	-	1	HO	Acc, Time	-	-
			· GPDT [268, 330]	1	11	HO	Acc, Size	-	C4.5
			· EDDIE [357]	-	2	HO	Acc	-	-
			· *Shirasaka et al.* (1998) [325]	-	1	FD	Acc, Size	-	-
			· *Ryan & Rayward-Smith* (1998) [314]	6	-	HO	ME, Size	-	C4.5
			· Cellular GP [117]	5	-	HO	ME	-	C4.5
			· *Zhao & Shirasaka* (1999) [400]	-	1	HO	Acc, Size	-	-
			· *Oka & Zhao* (2000) [272]	-	1	HO	Acc, Size	-	C4.5, [325]
			· *Tanigawa & Zhao* (2000) [348]	-	1	HO	Acc, Size	-	-
			· BGP [101, 311]	4	-	HO	Acc	t-test	C4.5, CN2
			· *Niimi & Tazaki* (2000) [267]	1	-	HO	Acc, Size	-	GP
			· *Haruyama & Zhao* (2002) [152]	3	1	HO	Acc, Size	-	[325]
			· *Eggermont et al.* (2003) [99]	5	-	10-f CV	ME	-	C4.5
			· *Eggermont et al.* (2004) [100]	6	-	10-f CV	ME	-	C4.5
			· EPTree [84]	-	2	10-f CV	Acc, Size	-	CART
			· EMO [188, 189]	8	1	10-f CV	ME	-	C4.5
			· GPTree [53, 380]	-	2	HO	Acc, Size	-	C5.0
			· *Kuo et al.* (2006–2008) [210–213]	-	1	HO	Acc	-	C5.0
			· *To & Pham* (2009) [352]	-	1	FD	Sen, Spe	-	SVM, LR, LDA
			· DTiGP [197]	18	-	4-f CV	Acc	Avg, WTL	J48, jaDTi, GP
			· MGP [392]	12	-	10-f CV	Acc, Time	-	C4.5, DTiGP, GP
			· MGP [375]	10	-	5-f CV	AUC	WTL, Wil-t	C4.5, FGP, GGP, EGP
			· EDDIE101 [377]	5	3	HO	Acc	Wil-t	J48, REPTree, RF, UDT-EA
			· GPEI [91, 92]	5	-	10-f CV	Acc	-	ID3-S, C4.5
			· *Saremi & Yaghmaee* (2014) [317]	6	-	HO	Acc, Size	-	C4.5
			· MOGP [376]	27	-	5-f CV	AUC	WTL, Wil-t	C4.5, NB, FGP, GGP, EGP, PRIE, 4 MOGPs
		GGP	· FGP [220, 356]	-	2	3-f CV	Acc	-	C4.5
			· CGP [222]	3	-	10-f CV	MC	-	C4.5, PART, kNN, NB
			· G^3P [355]	6	-	10-f CV, HO	ME, Size	-	C4.5, ID3, NB
			· *Johansson et al.* (2010) [176]	25	-	4-f CV	Acc	Fri-t, Nem-t	J48
			· *Johansson & Niklasson* (2009) [177], *Johansson et al.* (2011) [178]	26	-	4-f CV	Acc, Fid	Fri-t, BD-t, Nem-t	J48
		TGP	· *Khoshgoftaar et al.* (2003) [187], *Khoshgoftaar et al.* (2007) [186]	-	1	HO	ME	-	GP
			· *Zhao* (2007) [399]	13	-	FD	AUC	-	C4.5, BP-NN, SVM
	OB	GP	· *Liu & Xu* (2009) [228]	-	5	10-f CV, HO LOOCV	Acc	-	CART, RF, DF
		GGP	· GP-MM [3]	5	-	10-f CV	Acc	-	C4.5, GP, SVM, NB
		TGP	· *Bot & Langdon* (2000) [41, 42]	4	-	10-f CV	Acc, Size	-	C5.0, OC1, M5'
	NL	GP	· *Tackett* (1993) [345]	-	1	HO	Acc	-	BP-NN
			· UDT-EA [377]	5	3	HO	Acc	Wil-t	J48, REPTree, RF
			· neat-GP [354]	5	-	HO	ME, Size	Fri-t, BD-t	GP, FlatOE
		TGP	· *Mugambi & Hunter* (2003) [255]	-	1	HO	ROC	-	RBF-NN
	SF	GP	· Fuzzy-GP [98]	5	-	10-f CV	ME	-	C4.5, ESIA, CEFR-Miner
		TGP	PFDT [256]	-	2	HO	ROC	-	C4.5

others evolve using a weighted sum of the misclassification errors as their local fitness functions. The best chromosome from the main population competes with the best chromosome from the others as shown in Fig. 2.19. When it becomes dominant over the others, the local fitness function is adjusted in each population. A global fitness function is used to determine the dominance of one population over the others. Furthermore, Aitkenhead [7] implements competition between one axis-parallel DT and the training set. The algorithm mutates the nodes of the DT while updating the training set size and the DT depth. Initially, the training set contains only two sets of values randomly chosen from the dataset, and once DT evolves and its fitness value becomes higher than a threshold value, the training set size is increased. This process continues until the entire dataset is used to evaluate the DT.

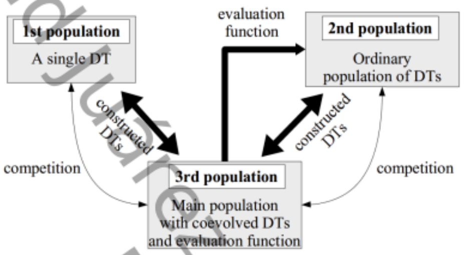


Figure 2.19: A competitive CEA method using one TGA proposed by Podgorelec and Kokol [292].

On the other hand, the multi-population genetic algorithm for DTI (MPGT) described by Podgorelec and Karakatic [289] and by Podgorelec *et al.* [290] is a cooperative CEA using two sub-populations of chromosomes to find near-optimal axis-parallel DTs. After a predefined number of generations, an exchange of DTs between the populations occurs according to one migrate rate. The first sub-population uses the training accuracy as its fitness function and the second one determines the balanced single-class accuracies as its fitness value.

Differential evolution: Lopes *et al.* [238] and Freitas *et al.* [124] use DE to implement a global search of a near-optimal oblique DT in the Perceptron Decision Tree (PDT) method. The coefficients of all hyperplanes used in a complete oblique DT are encoded in a matrix-based chromosome, and the independent term of the hyperplanes, as well as the class label of the leaf nodes, are stored in two vectors. In each DE iteration, mutation parameters are randomly altered as well as a group of new DTs randomly created replaces the worst chromosomes in the population. Furthermore, one special treatment for the leaf nodes is defined.

Grammatical evolution: Motsinger-Reif *et al.* [254] define an appropriate grammar to map axis-parallel DTs from binary strings used to model one gene-gene interaction [175].

Gene expression programming: Ferreira [111] and Wang *et al.* [378] use GEP to conduct a global search of near-optimal axis-parallel DTs. In these methods, the attributes of the dataset are functions, the class labels are terminals, and the *Dc* domain is used to represent the thresholds values used with numerical attributes. Furthermore, the GEP decision tree (GEPDT) method described by Qu *et al.* [300] considers the range of the values of each numerical attribute to compute the threshold values used in the test conditions. On the other hand, Weihong *et al.* [382] implement a GEP for DTI including a fuzzification process of the numerical attributes. Several symmetrical triangular membership functions are associated with each test condition of an axis-parallel DT previously induced with the GEPDT method.

Discussion

Table 2.17 shows the components of these approaches. Ten studies implement an uni-objective FF, and the remaining apply an aggregating FF. No study utilizes a multi-objective FF in its evolutionary process. Test accuracy is the fitness measure most commonly utilized by these methods. Furthermore, CEA-based approaches manipulate tree structures, and the others represent their chromosomes as sequences of values. Seven studies realize one linear-ranking-based selection, and four studies perform a roulette-wheel-based selection. Both the sub-tree-swapping-based crossover and the node-disturbance-based mutation are used 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 carried out with the majority of these approaches. In particular, Zhang *et al.* [396] to create a set of hyperplanes through an ANN-based approach. In the work of Aitkenhead (2008) [7], a randomly generated decision stump is used as the initial candidate solution in his CEA-based method.

Table 2.17: Components of other EA-based approaches for DTI.

Stra-tegy	DT	FF	MH	Studies	Repr. scheme	Fitness measure	Variation operators			Initial solution
							SEL	Xover	MUT	
GS	AP	UF	CEA	*Aitkenhead* (2008) [7]	Tree	Acc	N/A	N/A	NDM	A DS randomly created
			GEP	*Ferreira* (2006) [111], *Wang et al.* (2006) [378], GEPDT [300]	Linear	Acc	RWS	SPX, DPX, SpeX	UnM, SpeM	ChCs randomly created
		AF	CEA	*Podgorelec et al.* (1999–2002) [292–294], *Babič et al.* (2000) [11], *Sprogar* (2001) [335]	Tree	Acc∧Size	LRS	SSX	NDM	DTs randomly created
				MPGT [289, 290]	Tree	Acc∧Size	LRS	SSX	NDM	DTs randomly created
			GE	GEDT [254]	Linear	Sen, Spe	TouS	SPX	BIM	DTs randomly created
	OB	UF	ES	OC1-ES [58, 59]	Linear	Twoing	N/A	N/A	SpeM	The best APH found by the OC1-AP method
				MESODT [396]	Linear	IG, DLS	SpeS	SpeX	SpeM	Hyperplanes created using an ANN
			DE	PDT [124, 238]	Linear	ME	SpeS	SpeX	SpeM	RealCs randomly created
	SF	UF	GEP	*Weihong et al.* (2010) [382]	Linear	Acc	RWS	SPX, DPX, SpeX	UnM, SpeM	ChCs randomly created

In Table 2.18 is shown the 19 experimental studies reported by the existing literature implementing other EA-based approaches for DTI. This table shows that 12 methods induce DTs with several UCI datasets and eight studies with datasets from other sources. k-fold CV and hold-out have been applied as sampling methods in eight and nine studies, respectively. Also, the full datasets have been used in the works of Aitkenhead [7] and Wang *et al.* [378]. Size and test accuracy have been assessed by five and 17 studies, respectively. Also, Sen and Spe have been applied in four studies, and ME is evaluated by the GEDT method [254]. Six studies implement a statistical test of their experimental results, and one post-hoc analysis is carried out in the MPGT method [289]. Eleven studies compare their experimental results with those obtained by the C4.5 method and by the C5.0 algorithm. In particular, the results of the MPGT method are compared with those from a single population GA-based method implemented by the authors.

Table 2.18: Experimental analysis reported using other EA-based approaches for DTI.

Strategy	DT	MH	Studies	Datasets UCI	Datasets other	Sampling method	Performance measures	Statistical tests	Compared methods
GS	AP	CEA	·*Podgorelec et al.* (1999-2002) [292–294], *Sprogar* (2001) [335]	-	1	HO	Acc, Sen, Spe	-	C5.0, DF
			·*Babič et al.* (2000) [11]	-	1	HO	Acc, Sen, Spe	-	C5.0, MtDeciT
			·*Aitkenhead* (2008) [7]	2	-	FD	Acc	-	C4.5, GALE, BP-NN
			·MPGT [289, 290]	10	-	5-f CV	Acc, FM, Size	Fri-t, Nem-t	C4.5, CART, GT
		GEP	·*Ferreira* (2006) [111]	2	-	HO	Acc	-	-
			·*Wang et al.* (2006) [378]	2	-	FD	Acc	-	-
			·GEPDT [300]	8	-	5-f CV	Acc	-	C4.5
		GE	·GEDT [254]	-	40	10-f CV	ME	-	C4.5
	OB	ES	·OC1-ES [58, 59]	10	3	5-f CV	Acc, Size, Time	t-test	CART, OC1, OC1-GA, OC1-SA
			·MESODT [396]	2	2	HO	Acc, Size	-	C5.0, OC1, OC1-ES, APDT
		DE	·PDT [124, 238]	8	-	HO	Acc	KW-t	J48, BFTree, RTree, RBF-NN, MLP, NB, IB1
	SF	GEP	·*Weihong et al.* (2010) [382]	2	-	5-f CV	Acc	-	GEPDT

2.5.3 Swarm-intelligence-based methods

SI methods such as ACO and PSO have been applied for DTI. All SI-based approaches described in the existing literature induce only axis-parallel DTs. A timeline of these methods is shown in Fig. 2.20.

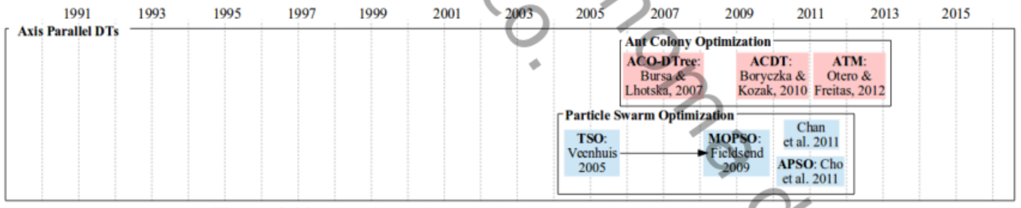


Figure 2.20: Timeline of swarm-intelligence-based methods for DTI.

Ant colony optimization: Bursa and Lhotska [54, 55] describe the ACO-DTree method as a global search strategy in which each artificial ant constructs an axis-parallel DT utilizing 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 the ants representing better solutions can deposit pheromone in the matrix. The induced DTs are pruned by applying a penalty value for unused nodes. A similar approach known as the ACDT (Ant Colony algorithm for constructing DTs) method is described by Boryczka and Kozak [36–39]. The ACDT method uses a combination of the splitting criterion and the pheromone values to select the attributes which will be used to build a DT. The induced DT is refined by applying the error-based pruning procedure. Also, an adaptive discretization of the numerical attributes is implemented by Boryczka and Kozak [37].

Furthermore, the Ant-Tree-Miner (ATM) method is described by Otero *et al.* [278] implements an iterative process in which each artificial ant creates a new DT based on a combination of the splitting criterion and the 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. The induced DTs are pruned by applying the error-based pruning procedure.

Particle swarm optimization: Veenhuis *et al.* [368] introduce a PSO-based approach to induce axis-parallel DTs. In this method, known as Tree Swarm Optimization (TSO) method, one DT is a particle moving in the solution space. The DT is represented as a sequence of nodes (test conditions and leaf nodes) traversed in breadth-first order. Each node has a *symbols vector* grouping all possible test conditions and the class labels. The symbol that will be used in the node is the one with the maximum value in the real-valued vector representing the particle in the swarm. If a numerical attribute is selected, the first element in this vector is taken as its associated threshold value. Furthermore, Fieldsend [114] implement a multi-objective PSO (MOPSO) for DTI based on the TSO and the Pareto dominance described by Alvarez-Benitez *et al.* [8]. In this method, instead of using one real-valued vector, a matrix in which each row represents a discrete symbol (an attribute or a class label) and each column is a node of the DT is applied. An additional column in the matrix is used to represent the threshold values.

On the other hand, Chan *et al.* [64] implement a method to find the test conditions with numerical attributes used in a DT induced employing a recursive partitioning strategy. Each particle represents one univariate test condition with a numerical attribute. Additionally, Cho *et al.* [67] implement a subsequent optimization strategy to improve the threshold values of the test conditions of a DT previously induced by the CART method. Each particle in the swarm encodes the threshold values of all test conditions used in the induced DT.

Discussion

In Table 2.19 are shown the components of the SI-based methods for DTI. Six studies evaluate an aggregating FF and four studies one uni-objective FF. Only the MOPSO method [114] applies a multi-objective FF. Test accuracy and ME have been used as fitness measures in six and four studies, respectively. Furthermore, both a matrix representation and a sequence of values have been applied to encode candidate solutions in eight and three studies, respectively. In particular, the ACO-based approaches use the pheromone matrix to represent a DT. Also, the MOPSO method utilizes a matrix to encode a complete oblique DT. Finally, the random generation of candidate solutions to create the initial population is the strategy applied in all studies implementing these MHs.

Table 2.20 resumes 11 experimental studies in the existing literature implementing SI-based methods for DTI. In this table is shown that all studies build DTs with UCI datasets, and four methods employ another type of datasets. Six studies implement a hold-out sampling procedure, and one k-fold CV is applied in the ATM method [278] and the APSO algorithm [67], as well as in the procedure described by Chan *et al.* [64]. Both the TSO method [368] and the MOPSO algorithm [114] use the full dataset to obtain their performance values. Test accuracy, size, and ME have been adopted as performance measures in eight, six and three studies, respectively. The ATM method is the only one reporting statistical tests using the Wilcoxon and the Friedman tests, as well as the Hommel post-hoc analysis to compare its results. Several studies compare their experimental results with those obtained by the C4.5 method and the CART algorithm. In particular,

Table 2.19: Components of SI-based approaches for DTI.

Strategy	DT	FF	MH	Studies	Repr. scheme	Fitness measure	Initial solution
GS	AP	UF	ACO	·ACO-DTree [54, 55]	Matrix	ME	A pheromone matrix randomly created
			PSO	·TSO [368]	Linear	ME	Particles randomly created
				·APSO [67]	Linear	Acc	Particles randomly created
		AF	ACO	·ACDT [36–39]	Matrix	Acc∧Size	A pheromone matrix randomly created
				·ATM [278]	Matrix	GR∧Pheromone	A pheromone matrix randomly created
			PSO	·*Chan, et al.* (2011) [64]	Linear	GR∧Acc	Particles randomly created
		MF	PSO	·MOPSO [114]	Matrix	ME	Particles randomly created

the results of the ACO-DTree method [54] are compared with those produced by an EA implemented by the authors.

Table 2.20: Experimental analysis reported using SI-based approaches for DTI.

Strategy	DT	MH	Studies	Datasets UCI	Datasets other	Sampling method	Performance measures	Statistical tests	Compared methods
GS	AP	ACO	·ACO-DTree [54, 55]	1	2	HO	ME	-	RA
			·ACDT [36–39]	7	-	HO	Acc, Size, Time	-	CART, Ant-Miner
			·ATM [278]	22	-	10-f CV	Acc, Size	Fri-t, Hom-t, Wil-t	C4.5, CART, aCDT
		PSO	·TSO [368]	1	-	FD	Acc	-	C4.5, GP
			·MOPSO [114]	5	-	FD	ME	-	C4.5, GP, TSO
			·APSO [67]	1	1	5-f CV	Acc	-	CART
			·*Chan, et al.* (2011) [64]	4	1	10-f CV	Acc, Size	-	C5.0, CART, QUEST, CHAID

2.5.4 Hyperheuristic-based approaches to build DTI methods

All HH-based approaches utilized to create DTI methods described in the existing literature use one EA. The timeline of the HH-based methods for DTI is shown in figure 2.21.

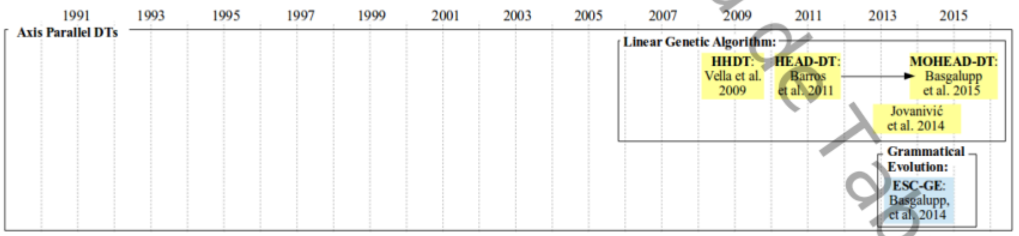


Figure 2.21: Timeline of HH-based approaches to build DTI methods.

Genetic algorithms with linear chromosomes: Vella *et al.* [369] describe a GA-based HH to build DTI algorithms. An individual in this method represents a list of rules to select the most appropriate splitting criterion based on the entropy degree of the attributes in a dataset. 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.* [13, 15, 16] describe the Hyper-heuristic Evolutionary Algorithm for Designing DT algorithms (HEAD-DT) in which four components of a DTI method are encoded in a linear chromosome: the split criterion, the stopping rule, the procedure to deal with missing values, and the pruning approach. Moreover, Basgalupp *et al.* [20] introduces a multi-objective version of the HEAD-DT procedure known as the MOHEAD-DT method allowing to choose between the Pareto dominance approach and the lexicographic analysis. The MOHEAD-DT method proposes several modifications in the fitness calculation, the selection operator, and the procedure to select the best method in the population. Furthermore, Jovanović *et al.* [181] implements a similar approach in which five components of a DTI process are encoded: the procedure to remove of insignificant attributes, the split criterion, the split evaluation method, the stop criterion, and the pruning approach. These components are obtained from several DTI methods such as ID3, C4.5, CART, and CHAID.

Grammatical evolution: Basgalupp *et al.* [18] 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 criterion for a DTI method. Each chromosome in the population represents a split criterion that is incorporated into a DTI algorithm.

Discussion

In Table 2.21 are shown the components of the HH-based approaches to building DTI methods. The HHDT method [369], the HEAD-DT algorithm [13], and the work of Jovanovic *et al.* [181] evaluate an uni-objective FF, the ESC-GE procedure [18] implements an aggregating FF, and the MOHEAD-DT method [20] applies a multi-objective FF. Test accuracy is the fitness measure most commonly evaluated by these HH-based 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 approaches. Finally, tournament-based selection, single-point crossover, and uniform mutation are the genetic operators most commonly used by these methods.

Table 2.21: Components of HH-based approaches to build DTI methods.

Strategy	DT	FF	MH	Studies	Repr. scheme	Fitness measure	Genetic operators SEL	Xover	MUT	Initial solution
GS	AP	UF	LGA	·HHDT [369]	Linear	Acc	TouS	SPX	UnM	NumC randomly created
				·HEAD-DT [13, 15, 16]	Linear	Acc	TouS	SPX	UnM	NumC randomly created
				·*Jovanović et al.* (2014) [181]	Linear	Acc	RWS	UnX	SpeX	ChC randomly created
		AF	GE	·ESC-GE [18]	Linear	P∧Sen	TouS	SPX	UnM	BinC randomly created
		MF	LGA	·MOHEAD-DT [20]	Linear	FM, Size	TouS	SPX	UnM	IntC randomly created

Finally, Table 2.22 resumes the experimental studies reported by the existing literature in which an HH is implemented to build DTI methods. In this Table is shown that four methods use UCI datasets, and the ESC-GE method [18] utilizes another type of datasets. All HH-based approaches to build DTI methods implement the k-fold CV, and also they calculate the test accuracy of the DTI method as a performance measure. F-measure and size are also evaluated in two and three studies, respectively. Three studies implement a statistical test to compare their results with those reported by other approaches. The results obtained by the C4.5 method have been used to analyze those obtained by the DTI methods constructed through these procedures. In particular, the experimental results of the HHDT method [369] are compared with those obtained by several variants of the ID3 method.

Table 2.22: Experimental evaluation reported by hyperheuristics.

Strategy	DT	MH	Studies	Datasets UCI	Datasets other	Sampling method	Performance measures	Statistical tests	Compared methods
GS	AP	LGA	· HHDT [369]	12	-	9-f CV	Acc	-	ID3
			· HEAD-DT [13, 15, 16]	20	-	10-f CV	Acc, FM, Size	Fri-t, Nem-t	C4.5, CART
			· *Jovanović et al.* (2014) [181]	16	-	10-f CV	Acc	-	C4.5, REPTree, ADTree, LMT
			· MOHEAD-DT [20]	20	-	10-f CV	FM, Size	Fri-t, Nem-t	C4.5, CART, HEAD-DT
		GE	· ESC-GE [18]	-	20	5-f CV	Acc, FM, Size	Fri-t, Nem-t	J48

2.5.5 Comparative analysis

In the following paragraphs, a comparative analysis of the different MH-based approaches for DTI is presented. First, a general summary of the classification of the methods described in this chapter is presented, followed by a comparative analysis of both the components of the algorithms and the elements considered in their experimental studies. An annual summary of the MH-based approaches for DTI is displayed in Fig. 2.22.

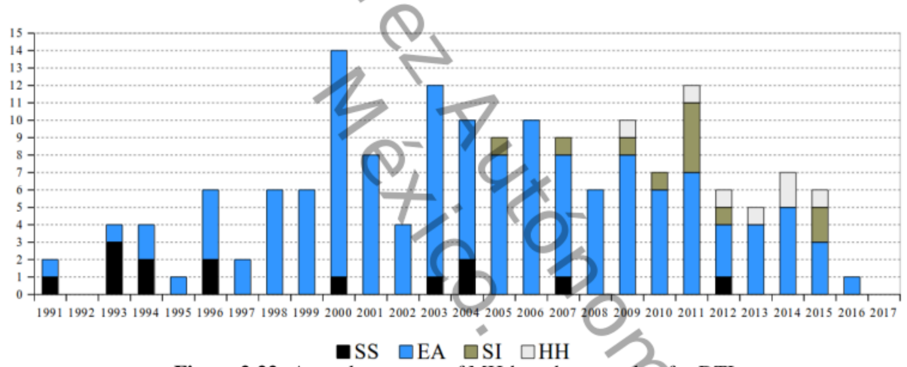


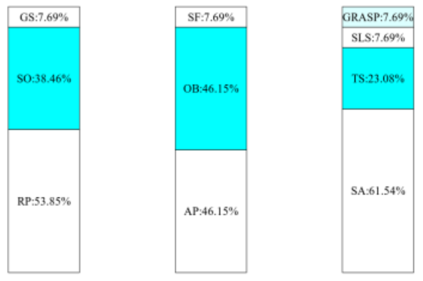
Figure 2.22: Annual summary of MH-based approaches for DTI.

Classification of MH-based approaches for DTI

SS-based methods for DTI: In Table 2.23 the classification of the SS-based approaches for DTI is detailed. Fig. 2.23 illustrates the proportion of these studies according to the type of strategy implemented, by the type of induced DT, and by the type of MH. Fig. 2.23(a) shows that recursive partitioning is the strategy most commonly used with these methods (53.85%). The SS-based MHs improves a single candidate solution in each step of its iterative process, and it is well situated to replace a standard splitting criterion by an intelligent search procedure to obtain a better separation of the training instances. This procedure provides better exploitation of the search space identifying areas with promissory solutions, and also it allows to escape from a local optimum. On the other hand, the subsequent optimization strategy (38.46%) is mainly applied to improve the performance of a DT previously induced by altering its test conditions or by including membership functions values of a soft DT. Fig. 2.23(b) shows that these methods have been applied to build both axis-parallel DTs and oblique DTs in the same proportion (46.15%). Finally, in Fig. 2.23(c) is shown that only four types of SS-based MHs have been used for DTI, highlighting the implementation of those based on SA (61.54%).

Table 2.23: Classification of SS-based MHs for DTI.

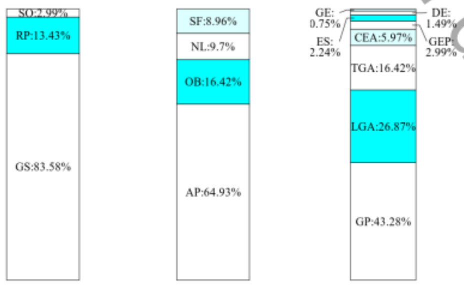
Strategy	DT	MH	Num studies	Studies
RP	AP	SA	1	*Ahmed & Rahman* (2004) [6]
		GRASP	1	*Pacheco et al.* (2012) [279]
	OB	SLS	1	OC1 [260, 261]
		SA	2	SADT [154]; OC1-SA [59]
		TS	2	LDTS [224]; LDSDT$_{TS}$ [277]
GS	AP	SA	1	GCP/SA [118]
SO	AP	SA	3	*Bucy & Diesposti* (1991) [51, 52]; SACS [240]
	OB	TS	1	EPTS [26]
	SF	SA	1	*Dvořák & Savický* (2007) [96]



(a) Strategy type (b) DT type (c) MH type

Figure 2.23: Percentages of studies implementing SS-based MHs for DTI.

EA-based methods for DTI: In Table 2.24 the classification of these methods is shown. Both GA and GP are the population-based approaches most commonly applied to implement DTI methods. They encode their candidate solutions through different structures: GA regularly utilizes linear chromosomes, and the standard GP evolves tree structures. Following the definitions and interpretations provided in the existing literature, two types of GA-based approaches for DTI are described in this review: LGA and TGA. LGA represents the standard GA encoding chromosomes with fixed-length linear sequences of values, and TGA uses tree-like structures for representing candidate solutions. There exists a disagreement about whether to consider a TGA as a GA or a GP. Since a TGA modifies a population of trees in its evolutionary process, it could be regarded as a GP. Nevertheless, since Koza [201] points out that the chromosomes in a GP are generated by combining the elements from both function and terminal sets, then a TGA should not be considered a GP because they do not define any set to represent their candidate solutions. In contrast, some authors point out that implement a GP-based approach, but they build their trees without using the previously referred sets. In this thesis, the analyzed studies are classified according to the definition provided in them.



(a) Strategy type (b) DT type (c) MH type

Figure 2.24: Percentages of studies implementing EA-based approaches for DTI.

Fig. 2.24(a) shows that the global search (83.58%) is the strategy most commonly implemented to build DTs with EA-based approaches. EAs evolve populations of chromosomes and use intelligent

Table 2.24: Classification of EA-based approaches for DTI.

Strategy	DT	MH	Num studies	Studies
RP	OB	ES	3	OC1-ES [58, 59]; MESODT [396]
		LGA	7	BTGA [62, 63]; OC1-GA [58, 59]; *Krętowski* (2004) [203]; *Pangilinan & Janssens* (2011) [282]; HBDT [343]
	NL	LGA	2	GA-QDT [265, 266]
		GP	2	*Marmelstein & Lamont* (1998) [241]; GIODeT [323]
	SF	LGA	4	*Janikow* (1996) [173]; GA-FID3 [65]; GC-SDT [326]; FVBDT [390]
GS	AP	LGA	10	ICET [361]; Caltrop [185]; Bandar *et al.* (1999) [12]; MEPDTI [9]; *Bratu et al.* (2007) [44]; *Smith* (2008) [331]; *Cha & Tappert* (2008) [60, 61]; ECCO [274]; EVO-Tree [174]
		TGA	18	*Podgorelec & Kokol* (1998) [291]; OOGASC4.5 [129]; GATree [182, 283, 284]; GAIT [130–134]; *Sörensen & Janssens* (2003) [333]; GEA-DT [206]; *Biedrzycki & Arabas* (2006) [29]; GDT-MC [208]; GDT-MA [204]; LEGAL-Tree [19, 21]; *Bosnjak et al.* (2015) [40]
		GP	46	*Koza* (1991) [200]; *Iba et al.* (1994) [171]; *Tür & Güvenir* (1996) [360]; GPDT [268, 330]; *Ryan & Rayward-Smith* (1998) [314]; EDDIE [357]; *Shirasaka et al.* (1998) [325]; Cellular GP [117]; *Zhao & Shirasaka* (1999) [400]; *Oka & Zhao* (2000) [272]; *Tanigawa & Zhao* (2000) [348]; *Niimi & Tazaki* (2000) [267]; FGP [220, 356]; BGP [101, 311]; *Haruyama & Zhao* (2002) [152]; *Khoshgoftaar et al.* (2003, 2007) [186, 187]; *Eggermont et al.* (2003, 2004) [99, 100]; EMO [188, 189]; EPTree [84]; GPTree [53, 380]; CGP [222]; G^{3}P [355]; *Kuo et al.* (2006–2008) [210–213]; *Zhao* (2007) [399]; *To & Pham* (2009) [352]; *Johansson & Niklasson* (2009) [177], *Johansson et al.* (2010, 2011) [176, 178]; DTiGP [197]; MGP [392]; MGP [375]; EDDIE-101 [377]; GPEI [91, 92]; MOGP [376]; *Saremi & Yaghmaee* (2014) [317]
		CEA	8	*Podgorelec et al.* (1999-2002) [292–294], *Babič et al.* (2000) [11], *Sprogar* (2001) [335]; *Aitkenhead* (2008) [7]; MPGT [289, 290]
		GE	1	GEDT [254]
		GEP	3	*Ferreira* (2006) [111]; *Wang et al.* (2006) [378]; GEPDT [300]
	OB	LGA	2	GDTI [93]; EFTI [373]
		TGA	4	*Siegel* (1994) [327]; GEA-ODT [205, 207]; TARGET [145]
		GP	4	*Bot & Langdon* (2000) [41, 42]; *Liu & Xu* (2009) [228]; GP-MM [3]
		DE	2	PDT [124, 238]
	NL	LGA	5	GALE [229–233]
		GP	4	*Tackett* (1993) [345]; *Mugambi & Hunter* (2003) [255]; UDT-EA [377]; Neat-GP [354]
	SF	LGA	2	*Kym & Ryu* (2005) [191, 192]
		GP	2	Fuzzy-GP [98]; PFDT [256]
		GEP	1	*Weihong et al.* (2010) [382]
SO	AP	LGA	1	*Chen et al.* (2009) [66]
	SF	LGA	3	*Crockett et al.* (1999) [76]; G-DT [286]; IIVFDT [316]

search procedures by applying a controlled interaction of their exploration and exploitation skills. These characteristics allow them to avoid the problems of traditional DTI methods. On the other hand, the recursive partitioning strategy is implemented in several EA-based approaches (13.43%). In this case, an EA is run as many times as internal nodes are required to build a DT. In Table 2.24 is shown that this strategy is commonly applied for inducing multivariate DTs by determining the best combination of attributes used by the test condition of one tree internal node. Finally, EA-based methods implementing a subsequent optimization strategy have been used to modify a previously induced DT through two alternatives: 1) for adding the membership functions in each internal node of a soft DT, and 2) for pruning a DT.

Fig. 2.24(b) shows that EA-based approaches are commonly utilized to induce both axis-parallel DTs

(64.93%) and oblique DTs (16.42%), and to build non-linear DTs and soft-DTs to a lesser extent. In Table 2.24 is shown that non-linear DTs are constructed by LGA-based and GP-based methods. On the other hand, soft DTs are induced by GA-based approaches since the parameters describing the membership functions associated with the possible outcomes of the test conditions in a soft DT can be encoded using linear chromosomes. Furthermore, clustering-based methods also have been utilized to induce soft DTs, either to determine the number of branches of each tree internal node or to fuzzify the values of the attributes in a dataset.

In Fig. 2.24(c) is shown that GP is the most common type of MH utilized to implement DTI methods (43.28%). Nevertheless, if LGA and TGA are grouped, they represent the 43.29% of these approaches. In contrast with the use of these MHs, other EAs such as ES, CEA, GE, DE, and GEP have been less applied for constructing DTs. They represent the 13.43% of all these methods, in comparison with the 86.57% of studies describing the application of GA or GP for DTI.

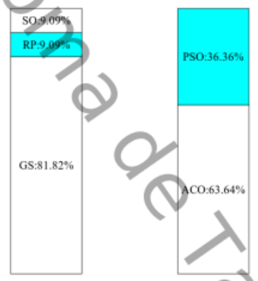
SI-based methods for DTI: SI-based methods have been sparsely applied for DTI with only 11 studies in the existing literature implementing this type of MHs. Table 2.25 shows the classification of these methods. In Fig. 2.25(a) is shown that global search is the most common strategy implemented for DTI with these methods (81.82%). In Table 2.25 is shown that the axis-parallel DT is the only one type of induced DT through SI-based methods. Fig. 2.25(b) shows that two types of SI methods have been applied to build DTs: ACO and PSO. ACO is the most used method to build DTs since it denotes a problem with a graph in which a set of artificial ants walk with the aim of finding a near-optimal solution. This scheme allows expressing a DT employing a pheromone matrix. The lack of development of SI-based approaches for DTI may be since this type of MHs is more recent than EAs. A challenge for the implementation of SI-based methods for DTI such as PSO is the definition of a scheme for mapping the DT structure from a real-valued vector representation.

Table 2.25: Classification of SI-based MHs for DTI.

Strategy	DT	MH	Num studies	Studies
RP	AP	PSO	1	*Chan et al. (2011)* [64]
GS	AP	ACO	7	ACO-DTree [54, 55]; ACDT [36–39]; ATM [278]
	AP	PSO	2	TSO [368]; MOPSO [114]
SO	AP	PSO	1	APSO [67]

Table 2.26: Classification of HH-based approaches to build DTI methods.

DT	MH	Num studies	Studies
AP	LGA	6	HHDT [369]; HEAD-DT [13, 15, 16]; *Jovanović et al.* (2014) [181]; MOHEAD-DT [20]
	GE	1	ESC-GE [18]



(a) Strategy type (b) MH type

Figure 2.25: Percentages of studies implementing SI-based approaches for DTI.

HH-based procedures to build DTI methods: In Table 2.26 is shown the classification of these methods. In this table can be observed that LGA is the MH most commonly used to implement DTI algorithms. HHs 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. HHs are a novel area in soft computing for machine learning, and the axis-parallel DTs are the only one type of DTs that are generated for the classifiers constructed with some HH-based approach.

Table 2.27 shows the classification of MHs-based approaches for DTI, in accordance with the type of strategy implemented, and Table 2.28 shows the classification of these approaches considering the type of induced DT. In Table 2.29 and Table 2.30 are shown the summaries of each type of classification. Fig. 2.26 shows the percentages of studies in the existing literature describing MHs-based approaches for DTI based on the strategy type, the DT type, and the MH type. Fig. 2.26(a) shows that global search is the prominent strategy implemented for MHs (78.18%) and in Fig. 2.26(b) is shown that axis-parallel DTs are the most common type of induced DT with these approaches (63.03%). In Fig. 2.26(c) is observed that GA (38.78%) and GP (35.15%) are the most commonly types of MHs implemented for DTI.

Table 2.27: Classification of MH-based approaches by type of strategy applied.

MH Type	MH	RP	GS	SO	Total
SS	GRASP	1			1
	SLS	1			1
	SA	3	1	4	8
	TS	2		1	3
EA	ES	3			3
	LGA	13	25	4	42
	TGA		22		22
	GP	2	56		58
	CEA		8		8
	DE		2		2
	GE		2		2
	GEP		4		4
SI	ACO		7		7
	PSO	1	2	1	4
Total		26	129	10	165

Table 2.28: Classification of MH-based approaches for type of induced DT.

MH Type	MH	AP	OB	NL	SF	HH	Total
SS	GRASP	1					1
	SLS		1				1
	SA	5	2		1		8
	TS		3				3
EA	ES		3				3
	LGA	11	9	7	10	6	42
	TGA	18	4				22
	GP	46	4	6	2		58
	CEA	8					8
	DE		2				2
	GE	1				1	2
	GEP	3			1		4
SI	ACO	7					7
	PSO	4					4
Total		104	28	13	13	7	165

Table 2.29: Summary of MH-based approaches by type of strategy applied.

MH Type	RP	GS	SO	Total
SS	7	1	5	13
EA	18	119	4	141
SI	1	9	1	11
Total	26	129	10	165

Table 2.30: Summary of MH-based approaches for type of induced DT.

MH Type	AP	OB	NL	SF	HH	Total
SS	6	6		1		13
EA	87	22	13	12	7	141
SI	11					11
Total	104	28	13	13	7	165

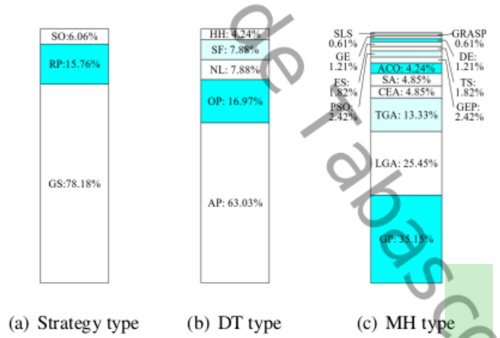


Figure 2.26: Percentages of studies implementing MH-based approaches for DTI.

Analysis of the components of MH-based methods for DTI

Fitness function and representation scheme: In Table 2.31 is shown the classification of MH-based approaches for type of fitness function, and Table 2.32 shows the classification of these approaches for type of representation scheme. Summaries of each type of classification are shown in Table 2.33 and in Table 2.34, respectively. Fig. 2.27 shows the percentages of studies in the existing literature describing MHs-based approaches for DTI. These rates are based on two constituent elements: the fitness function type, and the kind of representation scheme. Fig. 2.27(a) shows that uni-objective FF is the prominent type of fitness function in these MH-based approaches (55.15%), the aggregating FF is implemented in 56 studies (33.94%) and only 18 methods (10.91%) evaluate a multi-objective FF. Furthermore, in Fig. 2.27(b) is shown that the candidate solutions are represented with tree-like structures in 95 MH-based approaches (57.58%), with a sequence of values in 61 studies (36.97%), and with a matrix in nine studies (5.45%).

Table 2.31: Classification of MH-based approaches for type of fitness function.

MH Type	MH	UF	AF	MF	Total
SS	GRASP	1			1
	SLS	1			1
	SA	5	3		8
	TS	2	1		3
EA	ES	3			3
	LGA	33	8	1	42
	TGA	10	10	2	22
	GP	25	19	14	58
	CEA	1	7		8
	DE	2			2
	GE		2		2
	GEP	4			4
SI	ACO	2	5		7
	PSO	2	1	1	4
Total		91	56	18	165

Table 2.32: Classification of MH-based approaches for type of representation scheme.

MH Type	MH	Linear	Tree	Matrix	Total
SS	GRASP		1		1
	SLS	1			1
	SA	2	6		8
	TS	2		1	3
EA	ES	3			3
	LGA	42			42
	TGA		22		22
	GP		58		58
	CEA		8		8
	DE	2			2
	GE	2			2
	GEP	4			4
SI	ACO			7	7
	PSO	3		1	4
Total		61	95	9	165

Table 2.33: Summary of MH-based approaches for type of fitness function.

MH Type	UF	AF	MF	Total
SS	9	4		13
EA	78	46	17	141
SI	4	6	1	11
Total	91	56	18	165

Table 2.34: Summary of MH-based approaches for type of representation scheme.

MH Type	Linear	Tree	Matrix	Total
SS	5	7	1	13
EA	53	88		141
SI	3		8	11
Total	61	95	9	165

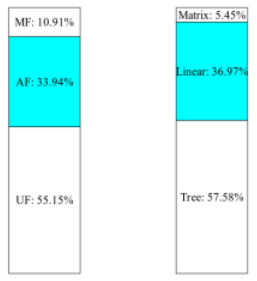
Fitness measure: Table 2.35 shows the classification of MH-based approaches for kind of fitness measure, and Table 2.36 shows the summary of this classification. Fig. 2.28 shows the percentages of the types of fitness measures used for MH-based approaches for DTI. Fig. 2.28(a) shows that twoing rule is adopted in seven studies (24.24%) and information gain is applied in six studies (20.69%). Other types of splitting criteria such as Gini index and the MDL principle have been utilized in several studies. On the other hand, in Fig. 2.28(b) is shown that test accuracy and size are the performance measures most commonly evaluated to determine the quality of the candidate solutions, with 41.71% and 31.75%, respectively. Other performance measures such as misclassification error and misclassification costs also have been used to determine the quality of a candidate solution.

Table 2.35: Classification of MH-based approaches by type of fitness measure.

MH type	**MH**	**Splitting criteria**				**Performance measures**				**Total**
		IG	**Twoing**	**Gini**	**Other**	**Acc**	**ME**	**Size**	**Other**	
SS	GRASP	1								1
	SLS	1	1	1	4					7
	SA		1		2		5	3		11
	TS	1					2			3
EA	ES	1	2		1					4
	LGA	2	3	4	2	19	5	11	5	51
	TGA					18	3	13	2	36
	GP					33	13	29	11	86
	CEA					8		7		15
	DE						2			2
	GE								4	4
	GEP					4				4
SI	ACO				1	4	2	4		11
	PSO				1	2	2			5
Total		6	7	5	11	88	34	67	22	240

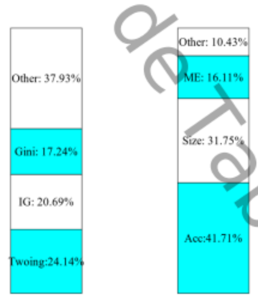
Table 2.36: Summary of MH-based approaches by type of fitness measure.

MH type	**Splitting criteria**				**Performance measures**				**Total**
	IG	**Twoing**	**Gini**	**Other**	**Acc**	**ME**	**Size**	**Other**	
SS	3	2	1	6		7	3		22
EA	3	5	4	3	72	23	60	22	202
SI				2	6	4	4		16
Total	6	7	5	11	88	34	67	22	240



(a) FF type (b) Rep. scheme type

Figure 2.27: Percentages of components used in MH-based approaches for DTI.



(a) Splitting criteria (b) Performance measure

Figure 2.28: Percentages of components used in MH-based approaches for DTI.

Genetic operators: The types of selection, crossover and mutation operators that have been used for EA-based approaches for DTI are shown in Table 2.39, Table 2.37 and Table 2.38, respectively. Since these operators have been applied alone or in combination, the total reported in each table is different. In Table 2.39 is shown that the tournament-based selection is utilized in 55 studies and the roulette-wheel-based selection is applied in 34 studies. In the case of EA-based methods using linear chromosomes, single-point crossover with 15 studies and uniform mutation with 11 studies are genetic operators most commonly implemented in these approaches. Sub-tree swapping with 82 studies and sub-tree replacement with 43 studies are the crossover and mutation operators most utilized by EA-based methods, respectively.

Table 2.37: Classification of EA-based approaches for type of crossover operator.

MH	Linear chromosome			Tree-based chromosome			Total
	SPX	DPX	Other	SSX	SpeX	Other	
LGA	10	8	4		3		25
TGA				21		12	33
GP				54	1	1	56
CEA				7			7
GEP	4	4			4		12
GE	1						1
ES					1		1
DE					2	0	2
Total	15	12	4	82	11	13	137

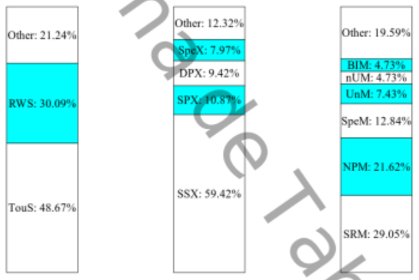
Table 2.38: Classification of EA-based approaches for type of mutation operator.

MH	Linear chromosome			Tree-based chromosome			Other	Total
	UnM	nUM	BIM	SRM	NDM	SpeM		
LGA	11	3	6			6		26
TGA					11		26	37
GP				43	13	5	3	64
CEA					8			8
GEP		4				4		8
GE			1					1
ES						2		2
DE						2		2
Total	11	7	7	43	32	19	29	148

Finally, in Fig. 2.29 is shown the proportion of genetic operators implemented in the EA-based approaches for DTI. The tournament-based selection (48.67%) and the sub-tree swapping crossover (59.42%), as well as the sub-tree replacement (29.05%) and the node disturbance (21.62%), are the genetic operators most used for this type of methods.

Table 2.39: Classification of EA-based approaches for type of selection operator.

MH	RWS	TouS	Other	Total
LGA	11	7	6	24
TGA	7	5	6	18
GP	12	42	2	56
CEA			7	7
GEP	4			4
GE		1		1
ES			1	1
DE			2	2
Total	34	55	24	113



(a) Selection type (b) Crossover type (c) Mutation type

Figure 2.29: Percentages of genetic operators used in EA-based approaches for DTI.

Analysis of the experimental studies

Sampling methods: In Table 2.40 is shown the classification of MH-based approaches by type of sampling method, and Table 2.42 shows the summary of this classification. Fig. 2.30(a) shows the percentages of each sampling methods used by these approaches. In this figure can be observed that cross-validation (49.7%) and hold-out (40.61%) are the sampling methods more utilized in the experimental studies reported by the existing literature. In table 2.40 is shown that 11 approaches use the full datasets in their experimental analysis and five studies do not report experimental results.

Performance measures: In Table 2.41 is shown the classification of MH-based approaches by performance measure, and Table 2.43 shows the summary of this classification. Fig. 2.30(b) shows the percentages of each performance measure used by these approaches. In this figure can be observed that accuracy (42.2%) and size (27.66%) are the performance measures more utilized in the experimental studies reported by the existing literature. In table 2.41 is shown that misclassification cost, misclassification error, and time also have been used in the experimental studies.

Table 2.40: Classification of MH-based approaches by type of sampling method.

MH Type	MH	CV	HO	FD	N/A	Total
SS	GRASP	1				1
	SLS	1				1
	SA	3	3	2		8
	TS	3				3
EA	ES	3				3
	LGA	26	10	3	3	42
	TGA	12	9		1	22
	GP	24	30	3	1	58
	CEA	2	6			8
	DE		2			2
	GE	2				2
	GEP	2	1	1		4
SI	ACO	1	6			7
	PSO	2		2		4
Total		82	67	11	5	165

Table 2.41: Classification of MH-based approaches by type of performance measure.

MH Type	MH	Acc	Size	ME	MC	Time	Other	Total
SS	GRASP	1	1					2
	SA	3	7	5		1		16
	SLS	1	1					2
	TS	1	2	2		2		7
EA	ES	3	3			1		7
	LGA	30	21	4	3	7	5	70
	TGA	19	16	2		7	2	46
	GP	38	18	11	1	2	9	79
	CEA	8	2				12	22
	DE	2						2
	GE	1	1	1			1	4
	GEP	4						4
SI	ACO	5	5	2		4		16
	PSO	3	1	1				5
Total		119	78	28	4	24	29	282

Table 2.42: Summary of MH-based approaches by type of sampling method.

MH Type	CV	HO	FD	N/A	Total
SS	8	3	2		13
EA	71	58	7	5	141
SI	3	6	2		11
Total	82	67	11	5	165

Table 2.43: Summary of MH-based approaches by type of sampling method.

MH Type	Acc	Size	ME	MC	Time	Other	Total
SS	6	22	7		3		38
EA	105	50	18	4	17	29	223
SI	8	6	3		4		21
Total	119	78	28	4	24	29	282

Statistical tests: In Table 2.44 is shown the classification of MH-based approaches by statistical test, and Table 2.45 shows the summary of this classification. Fig. 2.30(c) shows the percentages of each performance measure used by these approaches. In this figure can be observed that t-test (29.69%), the averaging over the datasets (23.4%) and the Friedman test (21.88%) are the statistical tests more utilized in the experimental studies reported by the existing literature. In table 2.44 is shown that Wilcoxon test also has been used in the experimental studies to compare two algorithms.

Table 2.44: Classification of MH-based approaches by type of statistical test.

MH Type	MH	Avg	WTL	t-test	ANOVA	Wil-t	Fri-t	KW-t	Total
SS	GRASP	1	1	1					3
	SLS								0
	SA	1		1					2
	TS	1			1				2
EA	LGA	8		9	2	1	5		25
	TGA	3		4			1	1	9
	GP	1	3	2		4	4		14
	ES			2					2
	CEA						2		2
	DE							2	2
	GE						1		1
	GEP								0
SI	ACO					1	1		2
	PSO								0
Total		15	4	19	3	6	14	3	64

Table 2.45: Summary of MH-based approaches by type of statistical test.

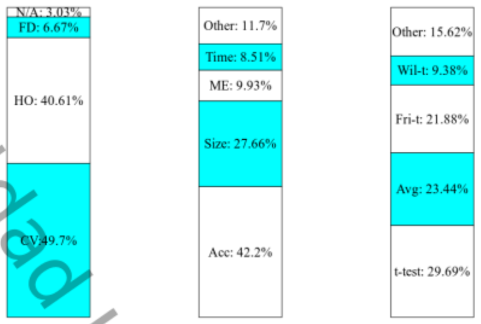
MH Type	Avg	WTL	t-test	ANOVA	Wil-t	Fri-t	KW-t	Total
SS	3	1	2	1				7
EA	12	3	17	2	5	13	3	55
SI					1	1		2
Total	15	4	19	3	6	14	3	64

2.5.6 Final remarks

DTI algorithms stand out of other machine learning techniques to build predictive models from data, since they are simple procedures producing accurate models with a high level of expressiveness and interpretability, unlike other classification techniques. Three types of strategies to implement MH-based approaches for DTI have been identified in this thesis: recursive partitioning, global search, and subsequent optimization.

The principal strategy in which MHs have been implemented for DTI is to realize a global search in the search space. On the other hand, MHs-based approaches for DTI implementing a recursive partitioning strategy have been applied to replace the traditional splitting criteria into the induction process. These approaches can solve the selection bias problem by adjusting their fitness function, and also can improve the construction of multivariate test conditions to induce oblique and non-linear DTs. However, in them, the overfitting and the instability of small changes in the training set persist since these problems are inherent to the greedy strategy. Finally, in the case of subsequent optimization, this strategy is rarely used and is applied to improve the performance of a previously induced DT with some other approach or to introduce soft conditions in the tree.

The use of MH for DTI has provided a new approach to building classifiers with better performance, especially by their ability to perform a global search in the solution space. Even though its application to induce DTs began 25 years ago, it currently presents many opportunities to study and new challenges, since there are characteristics of the MHs that must be analyzed in the context of the machine learning. Several research areas could be developed, such as the application of new single-solution based MHs or improvements to known methods that have been recently published. Another area of potential development is the application of SI-based methods for DTI, proposing schemes to allow an efficient search in the DT



(a) Sampling method (b) Performance measure (c) Statistical test

Figure 2.30: Percentages of the elements included in the experimental studies.

space using the operators defined to update the agents or particles in the swarm. Finally, the use of HH to build algorithms allowing the induction of DTs is a strategy that has been gaining interest in recent years.

Chapter 3

Decision tree induction with the differential evolution algorithm

All life evolves by the differential survival of replicating entities

Richard Dawkins, The Selfish Gene

THREE DE-based algorithms for DTI are described in this chapter. First, DE is applied to find a near-optimal hyperplane splitting a set of training instances, in a recursive partitioning strategy to induce one oblique DT. Next, two procedures to conduct a global search to find a near-optimal DT are proposed: one to build oblique DTs, and the other to induce axis-parallel DT.

3.1 OC1-DE method to induce an oblique decision tree

6

In this thesis, a method to induce an oblique DT using `DE/rand/1/bin` in a recursive partitioning strategy is introduced. This method, nam 6 OC1-DE, is similar to the OC1 system [261] and its GA-based variant [59] but applies the DE algorithm to finding a near-optimal hyperplane at each internal node of an oblique DT. Since the task of finding a near-optimal hyperplane with real-valued coefficients is an optimization problem in a continuous space, DE operators can be applied without any modification, and the OC1-DE method should induce one better oblique DT.

3.1.1 Recursive partitiong strategy to induce oblique decision trees

6 6

When a recursive partitioning strategy is implemented to induce one oblique DT, the classification method starts finding the hyperplane that best splits the training instances into two subsets. This hyperplane will be used as test condition of a new internal node that is added in the DT. This procedure is applied for each instances subset previously created until a leaf node is created. Hyperplane quality is estimated through a splitting criterion measuring the impurity of a partition or other discriminant value. Finally, a pruning process is carried out to reduce the tree overfitting and to improve its predictive power. The splitting criterion can be applied exhaustively to find the best hyperplane of all possible partitions of the set, or perform a search guided by a metaheuristic. In particular, the OC1-DE method uses the DE algorithm for this purpose. In the next paragraphs, the OC1 variants first are described and then the OC1-DE method is detailed.

3.1.2 OC1 variants

The OC1 system implements a two-step process to search a better hyperplane dividing the instances set. These steps are delineated in the Algorithm 2. First, OC1 finds the best axis-parallel hyperplane h^0 splitting the instances set ϕ. Next, it applies the following perturbation schemes:

3

- *Sequential perturbation*: This is a deterministic rule adjusting the hyperplane coefficients, taking one at a time and looking for its optimal value.
- *Random vector perturbation*: When the sequential perturbation reaches a local optimum, a random vector is added to the current hyperplane with the aim of looking elsewhere in the solutions space.

Finally, the OC1 system returns as the best hyperplane to the one selected between the best-perturbed hyperplane h and the best axis-parallel hyperplane h^0. This algorithm requires two parameters to control the iterative process: The number of restarting steps, and the number of random perturbations.

Algorithm 2 Hyperplane selection proposed in the OC1 system (based on [261]).

```
function OC1($a,\phi,R,J$)
    Input: The set of attributes ($a$), the instances set ($\phi$), the number of restarting steps ($R$), and the
           number of random perturbation ($J$).
    Output: The best hyperplane ($h^{\text{best}}$).

    $h^0 \leftarrow$ BESTAXISPARALLELHYPERPLANE($a,\phi$)
    for each $i \in \{1,\dots,R\}$ do
        $h \leftarrow \begin{cases} \text{A random hyperplane} & \text{if } i \neq 1, \\ h^0 & \text{otherwise} \end{cases}$
(A):    repeat
            $h^p \leftarrow$ SEQUENTIALPERTURBATION($h$)        ▷ Perturb each of coefficients of $h$ in sequence
            if $f(h^p,\phi)$ is better than $f(h,\phi)$ then
                $h \leftarrow h^p$
            end if
        until $f(h^p,\phi)$ is not better than $f(h,\phi)$
        for each $j \in \{1,\dots,J\}$ do
            $h^p \leftarrow$ RANDOMVECTORPERTURBATION($h$)        ▷ Random vector perturbation of $h$
            if $f(h^p,\phi)$ is better than $f(h,\phi)$ then
                $h \leftarrow h^p$
                go to (A)
            end if
        end for
        $h^{\text{best}} \leftarrow \begin{cases} h & \text{if } f(h,\phi) \text{ is better than } f(h^0,\phi), \\ h^0 & \text{otherwise} \end{cases}$
    end for
    return $h^{\text{best}}$
end function
```

Based on the OC1 system, Cantú-Paz and Kamath [59] implement the OC1-GA algorithm in which replace the OC1 perturbation schemes by one GA. This method evolves a population of real-valued chromosomes to find a near-optimal hyperplane evaluating its quality through the twoing rule. The OC1-GA algorithm uses a pairwise-tournament-based selection operator, and one uniform crossover operator, but does not use a mutation operator.

The OC1-GA algorithm returns as the best hyperplane to the one selected between the best hyperplane in the last population and the best axis-parallel hyperplane. This algorithm sets the population size based on the number of attributes in the training set, as well as uses a fixed number of generations. Algorithm 3 shows the structure of this GA-based approach to find near-optimal hyperplanes.

Algorithm 3 Hyperplane selection using the OC1-GA algorithm (based on [59]).

```
function OC1-GA($a, \phi$)
    Input:   The set of attributes ($a$), the instances set ($\phi$).
    Output:  The best hyperplane ($h^{\text{best}}$).

    $g \leftarrow 0$
    $h^0 \leftarrow$ BESTAXISPARALLELHYPERPLANE($a, \phi$)
    if $|\phi| < 2|a|$ then
        return $h^0$
    end if
    $X_0 \leftarrow$ Initial population of random hyperplanes with 10% of copies of $x^{\text{ap}}$.
    EVALUATE($X_0$)
    for each $g \in \{1, \ldots, 25\}$ do
        $X \leftarrow$ TOURNAMENTSELECTION($X_{g-1}$)
        $X_g \leftarrow$ UNIFORMCROSSOVER($X$)
        EVALUATE($X_g$)
    end for
    $x^{\text{best}} \leftarrow$ The best chromosome in $X_g$
    $h^{\text{best}} \leftarrow \begin{cases} x^{\text{best}} & \text{if } f(x^{\text{best}}, \phi) \text{ is better than } f(h^0, \phi) \\ h^0 & \text{otherwise} \end{cases}$
    return $h^{\text{best}}$
end function
```

3.1.3 OC1-DE method

The OC1-DE method implemented to find a near-optimal hyperplane is shown in Algorithm 4. This approach is very similar to the OC1-GA algorithm: the axis-parallel hyperplane that best splits a set of training instances is first obtained, and it is inserted in an initial population of hyperplanes randomly created. Then, this population is evolved through several generations using the DE operators. Finally, the OC1-DE algorithm returns the hyperplane selected between the best axis-parallel hyperplane and the best oblique hyperplane in the last population of the OC1-DE algorithm.

3.2 DE-ODT method to find a near-optimal oblique decision tree

8

The DE-ODT method implements a global search strategy with the aim of constructing more accurate oblique DTs, and also to overcome the inherent problems of the recursive partitioning strategy. This method evolves a population of oblique DTs encoded in fixed-length real-valued vectors.

Algorithm 4 Hyperplane selection using the OC1-DE algorithm.

```
function OC1-DE($a, \phi$, CR, F, NP)
    Input: The set of attributes ($a$), the instances set ($\phi$), the crossover rate (CR), the scale factor (F),
           and the population size (NP).
    Output: The best hyperplane ($h^{\text{best}}$).

    $g \leftarrow 0$
    $h^0 \leftarrow \text{BestAxisParallelHyperplane}(a, \phi)$
    $X_0 \leftarrow$ Initial population of random hyperplanes and one copy of $h^0$.
    while stop condition is not fullfilled do
        $g \leftarrow g + 1$
        $X_g \leftarrow \varnothing$
        for each $i \in \{1, \ldots, \text{NP}\}$ do
            $x^i \leftarrow$ Target vector from $X_{g-1}$
            $v^i \leftarrow$ Mutated vector generated using (2.4)
            $u^i \leftarrow$ Trial vector constructed using (2.5)
            $X_g \leftarrow X_g \cup \begin{cases} \{(i, u^i)\} & \text{if } f(u^i, \phi) \text{ is better than } f(x^i, \phi) \\ \{(i, x^i)\} & \text{otherwise} \end{cases}$
        end for
    end while
    $x^{\text{best}} \leftarrow$ The best individual in $X_g$
    $h^{\text{best}} \leftarrow \begin{cases} x^{\text{best}} & \text{if } f(x^{\text{best}}, \phi) \text{ is better than } f(h^0, \phi) \\ h^0 & \text{otherwise} \end{cases}$
    return $h^{\text{best}}$
end function
```

3.2.1 Linear representation of oblique decision trees

In the DE-ODT method, each candidate solution encodes only the internal nodes of a complete binary oblique DT stored in a fixed-length real-valued vector (Fig. 3.1). This vector represents the set of hyperplanes used as test conditions of the oblique DT. The main advantage of the linear representation is that it is utilized to encode candidate solutions in several EAs such as GA, DE and ES and they can be implemented for DTI without any modification. Nevertheless, since these EAs use a fixed-length representation, it is necessary to define a priori this length, and this can limit the performance of the induced DTs. In the DE-ODT method, the size of the real-valued vector is determined using both the number of attributes and the number of class labels of the training set whose model is induced.

Since each internal node of an oblique DT has a hyperplane as its test condition, the size of the real-valued vector x^i used to encode each i-th candidate solution in the population is fixed as $n_e(d+1)$, where n_e is the estimated number of internal nodes of a complete binary oblique DT, and d is the number of attributes in the dataset. Considering that: 1) an oblique DTs is more compact than its univariate version, and that 2) the DT size is related to the structure of the training set, the DE-ODT method determines the value of n_e based on both the number of attributes and the number of class labels in it. If the number of internal nodes of a complete binary DT with height H is $2^H - 1$, and the number of leaf nodes of the same DT is 2^H, two heights can be obtained as follows:

$$H_i = \lceil \log_2(d+1) \rceil, \tag{3.1}$$

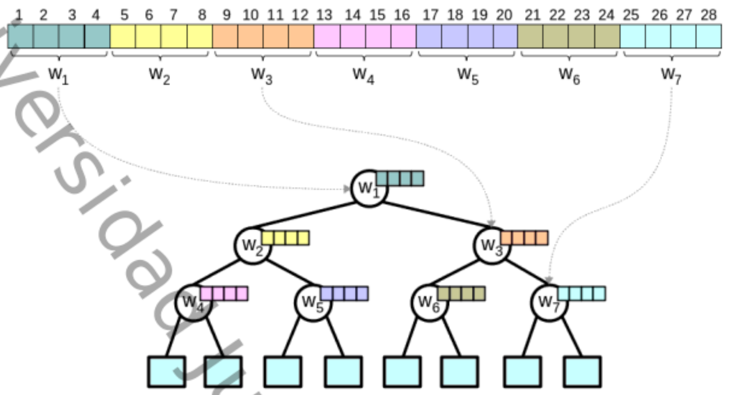


Figure 3.1: Linear encoding scheme for the internal nodes of a complete binary oblique tree.

and

$$H_l = \lceil \log_2(s) \rceil. \tag{3.2}$$

Using these equations, n_e is determined as follows:

$$n_e = 2^{\max(H_i, H_l)-1} - 1, \tag{3.3}$$

and, the size of the real-valued vector representing a sequence of n_e hyperplanes for a training set with d attributes is computed as follows:

$$n = n_e(d+1). \tag{3.4}$$

As an example, if a hypothetical dataset with three numerical attributes and three class labels is used to induce an oblique DT, then $d = 3$ and $s = 3$. In this case, $H_i = \lceil \log_2(4) + 1 \rceil = 3$ and $H_l = \lceil \log_2(3) + 1 \rceil = 3$. Finally, $n_e = 2^{\max\{3,3\}} - 1 = 7$. This implies that the oblique DT could have seven internal nodes. Finally, one individual representing a candidate solution in the evolutionary process has 28 real-valued parameters.

3.2.2 Induction of feasible oblique decision trees

The DE-ODT method implements one three-stages procedure to map an oblique DT from a real-valued individual of the population. First, x^i is used to build the vector w^i which encodes a sequence of candidate nodes of an oblique DT. Each node contains one hyperplane utilized to divide the training instances. Next, w^i is traversed to create a partial tree pT^i composed only of internal nodes. Finally, to complete the DT, a set of leaf nodes are added in pT^i using the training set. This procedure allows inducing feasible oblique DTs with a different number of nodes, although they are represented with a fixed-length parameters vector. Fig. 3.2 shows a graphical representation of this procedure.

1. **Hyperplanes construction:** Vector x^i is used to build the vector w^i representing the sequence of candidate internal nodes of a partial DT. Since the values of x^i represent the hyperplane coefficients contained in these nodes, the following criterion applies: Values $\{x^i_1, \ldots, x^i_{d+1}\}$ are assigned to the hyperplane h^1, the values $\{x^i_{d+2}, \ldots, x^i_{2d+2}\}$ are assigned to the hyperplane h^2, and so on. For each $j \in \{1, \ldots, n_e\}$, the coefficients of h^j are designed as follows:

$$h^j_k = \left\{ x^i_{(j-1)(d+1)+k} : k \in \{1, \ldots, d+1\} \right\}. \tag{3.5}$$

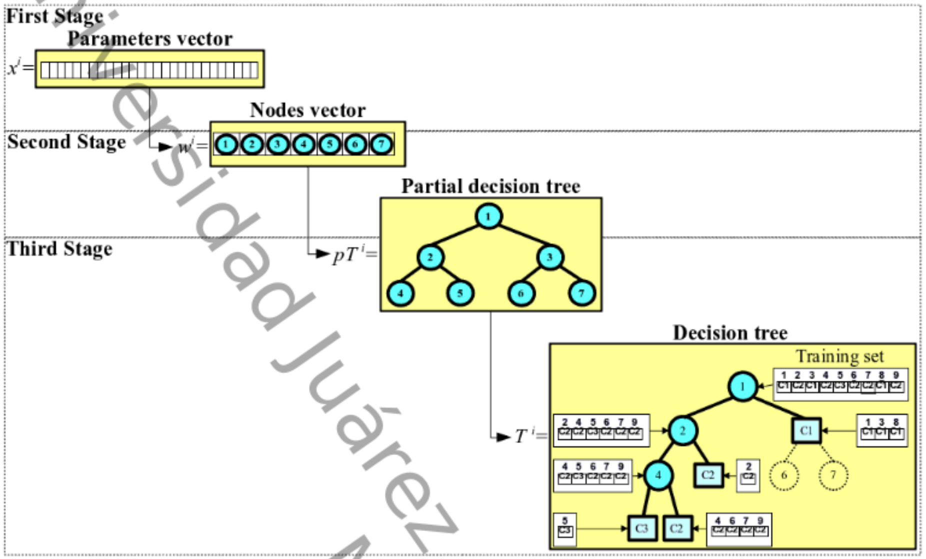


Figure 3.2: Three stages procedure to map an oblique DT from x^i.

These hyperplanes are assigned to the elements of w^i: h^1 is assigned to w^i_1, h^2 is assigned to w^i_2, an so on. The Algorithm 5 outlines the process to build w^i from x^i. In Fig. 3.3 is shown an example of the construction of a set of hyperplanes which are assigned to the seven nodes of one candidate solution x^i for the hypothetical dataset previously described. Once w^i is completed, it is used to create a partial DT with only internal nodes.

Algorithm 5 Algorithm to build w^i from x^i.

```
function OBLIQUENVCONSTRUCTION($x^i, d$)
    Input: The real-valued parameter vector ($x^i$), and the number of attributes ($d$) in the training set.
    Output: The nodes vector ($w^i$)
    .
    for each $j \in \{1,\dots,n_e\}$ do
        for each $k \in \{1,\dots,d+1\}$ do
            $h^j_k \leftarrow x^i_{(j-1)(d+1)+k}$
        end for
        $w^i_j \leftarrow (h^j)$
    end for
    return $w^i$                                    ▷ Vector of internal nodes
end function
```

2. **Partial oblique decision tree construction:** A straightforward procedure is applied to construct the partial DT from w^i: First, the element in the initial location of w^i is used as the root node of pT^i. Next, the remaining elements of w^i are inserted in pT^i as successor nodes of those previously added so that each new level of the tree is completed before placing new nodes at the next level, in a similar way to the breadth-first search strategy. Since a hyperplane divides the training instances into two subsets,

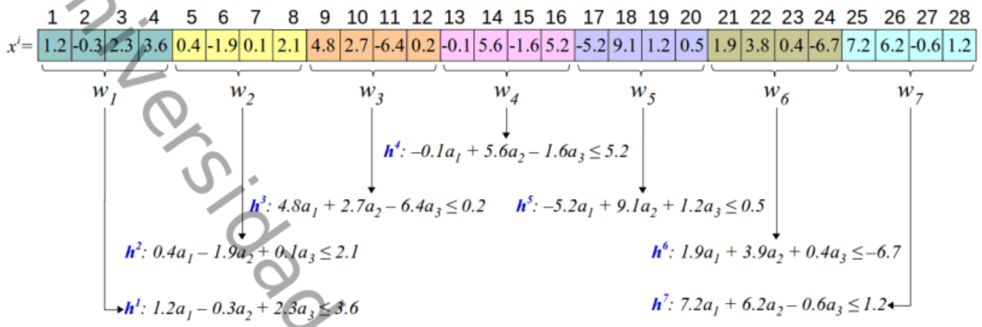

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
x^i = 1.2 -0.3 2.3 3.6 0.4 -1.9 0.1 2.1 4.8 2.7 -6.4 0.2 -0.1 5.6 -1.6 5.2 -5.2 9.1 1.2 0.5 1.9 3.8 0.4 -6.7 7.2 6.2 -0.6 1.2
w1 w2 w3 w4 w5 w6 w7
h4: −0.1a1 + 5.6a2 − 1.6a3 ≤ 5.2
h3: 4.8a1 + 2.7a2 − 6.4a3 ≤ 0.2
h5: −5.2a1 + 9.1a2 + 1.2a3 ≤ 0.5
h2: 0.4a1 − 1.9a2 + 0.1a3 ≤ 2.1
h6: 1.9a1 + 3.9a2 + 0.4a3 ≤ −6.7
h1: 1.2a1 − 0.3a2 + 2.3a3 ≤ 3.6
h7: 7.2a1 + 6.2a2 − 0.6a3 ≤ 1.2

Figure 3.3: Construction of a set of hyperplanes from x^i.

each internal node has assigned two successor nodes.

Algorithm 6 shows the steps applied to create pT^i from w^i. In this algorithm can be observed that V is the set of valid internal nodes of pT^i, and E is the set of edges representing each possible outcome of each test condition in pT^i.

Algorithm 6 Construction of pT^i from w^i.

```
function OBLIQUEDTCONSTRUCTION($w^i$)
    Input: The nodes vector ($w^i$).
    Output: The partial DT with only internal nodes ($pT^i$).
    $V \leftarrow \{w_1^i\}$                              ▷ Set of internal nodes of $pT^i$
    $E \leftarrow \varnothing$          ▷ Set of edges of $pT^i$ used to define the successor nodes
    $k \leftarrow 1$
    for each $j \in \{1,\ldots,n_e\}$ do
        $l \leftarrow 0$
        while $k \leq n_e \wedge l < 2$ do
            $k \leftarrow k+1$
            $l \leftarrow l+1$
            $V \leftarrow V \cup \{w_k^i\}$
            $E \leftarrow E \cup \{(w_j^i, w_k^i)\}$
        end while
    end for
    $pT^i \leftarrow (G(V,E), w_1^i)$
    return $pT^i$                              ▷ partial DT with only internal nodes
end function
```

In Fig. 3.4 is shown an example of the construction of a partial oblique DT from w^i. In this figure can be observed that w_1^i is selected as the tree root node, w_2^i and w_3^i are placed as the successor nodes of w_1^i, w_4^i and w_5^i are designed as the successor nodes of w_2^i, and so on. The partial oblique DT constructed for this example is $pT^i = (G(V,E), w_1^i)$, where V and E are formed as follows:

$$
\begin{aligned}
V &= \{w_1^i, w_2^i, w_3^i, w_4^i, w_5^i, w_6^i, w_7^i\} \\
E &= \{(w_1^i, w_2^i), (w_1^i, w_3^i), (w_2^i, w_4^i), (w_2^i, w_5^i), (w_3^i, w_6^i), (w_3^i, w_7^i)\}
\end{aligned} \tag{3.6}
$$

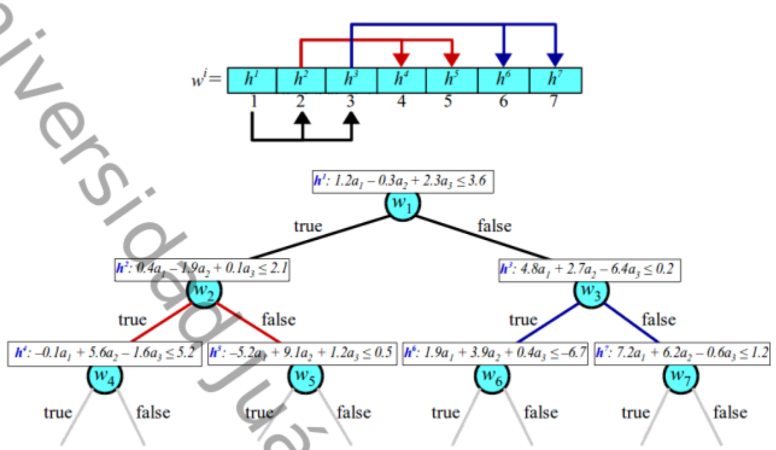

$w^i =$
h^1 h^2 h^3 h^4 h^5 h^6 h^7
1 2 3 4 5 6 7
h^1: $1.2a_1 - 0.3a_2 + 2.3a_3 \leq 3.6$
h^2: $0.4a_1 - 1.9a_2 + 0.1a_3 \leq 2.1$
h^3: $4.8a_1 + 2.7a_2 - 6.4a_3 \leq 0.2$
h^4: $-0.1a_1 + 5.6a_2 - 1.6a_3 \leq 5.2$
h^5: $-5.2a_1 + 9.1a_2 + 1.2a_3 \leq 0.5$
h^6: $1.9a_1 + 3.9a_2 + 0.4a_3 \leq -6.7$
h^7: $7.2a_1 + 6.2a_2 - 0.6a_3 \leq 1.2$
w_1 w_2 w_3 w_4 w_5 w_6 w_7
true
false

Figure 3.4: Construction of a partial oblique DT pT^i.

3. **Decision tree completion:** In the final stage of this mapping scheme, several leaf nodes are added in pT^i by evaluating the training set. One instance set ϕ is assigned to one internal node ω (ι to the root node), and by evaluating each element in ϕ with the hyperplane associated to ω, two instances subsets are created and assigned to the successor nodes of ω. This assignment is repeated for each node in pT^i. Two cases should be considered:

   (a) If ω is located at the end of a branch of pT^i, then two nodes are created, and they are designated as successor nodes of ω. Each instances subset is assigned to each created node, and each one is labeled as a leaf node using as its class label the one that has the highest number of occurrences in the instances subset assigned to it.

   (b) If the number of instances assigned to ω is less than one previously defined threshold value τ, or if all instances assigned to it belong to the same class, then ω is labeled as a leaf node. The majority class ζ of these instances is designed as the class label of the leaf node, and its successor nodes are removed, if they exist.

   Algorithm 7 summarizes the process to complete the oblique DT from pT^i. This procedure uses a first-in-first-out (FIFO) queue to assign the instances of the training set in each node of the DT. In this algorithm can be observed that V' is the set of both internal nodes and leaf nodes of T', each one associated with an instances subset. Fig. 3.5 shows an example of the tree completion through this mapping scheme. In this figure is shown that all the instances assigned to w_3 and w_5 have the same class label, so they are designed as leaf nodes, and the successor nodes of w_3 are removed from the tree. On the other hand, since w_4 is the ending node of a branch, its instances set is split using its hyperplane, the instances subsets produced are assigned to two new leaf nodes, and their majority classes are designated as their class labels. In this figure can be observed that this tree has three internal nodes and four leaf nodes.

3.2.3 General structure of the DE-ODT method

Algorithm 8 shows the structure of the DE-ODT method proposed in this thesis. This procedure requires to identify the training set used to induce an oblique DT, as well as the three control parameters applied by

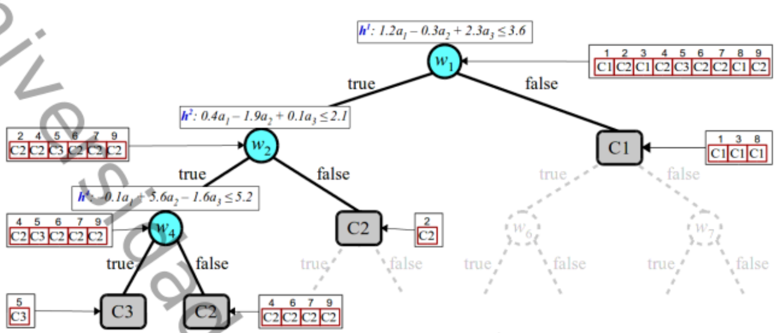

h^1: $1.2a_1 - 0.3a_2 + 2.3a_3 \leq 3.6$
h^2: $0.4a_1 - 1.9a_2 + 0.1a_3 \leq 2.1$
h^4: $-0.1a_1 + 5.6a_2 - 1.6a_3 \leq 5.2$
w_1
w_2
w_4
w_6
w_7
true
false
C1
C2
C3

Figure 3.5: Completion of an oblique DT using pT^i and the training instances.

the DE algorithm, and the threshold value τ used to determine if a node is labeled as a leaf node. First, the DE-ODT method uses the READTRAININGSET method to get the attributes vector a, the vector of class labels c, and the instances set ι. Next, the value of d and n are computed. Then, the DE algorithm evolves a population of real-valued individuals encoding oblique DTs. DE selects the best candidate solution x^{best} in the last population as the result of its evolutionary process. Finally, a near-optimal oblique DT is constructed applying the procedures described in the previous paragraphs. In particular, the F parameter in the DE-ODT method gradually decreases as the evolutionary process progresses. This decrement allows more exploration of the search space at the beginning of the evolutionary process, and with the passage of the generations, it is tried to make one better exploitation of promising areas of this space [79].

Since the DE-ODT method uses an a priori definition of the size of the real-valued vector, it is possible that some leaf nodes in the DT do not meet the following conditions: the size of its instances subset is less than τ, or all instances in the subset belong to the same class. In this case, the DE-ODT method applies the OBLIQUETREEGROWING procedure to replace this node with a sub-tree whose leaf nodes fulfill these conditions. This method implements a recursively partitioning strategy guided by some splitting criterion. However, it is desirable that this refinement is used only when the estimated number of nodes n_e does not permit to build a DT with acceptable accuracy. The Algorithm 9 shows the procedure to refine the best T^i constructed with the DE-ODT method. In this work, the twoing rule is used for the OBLIQUETREEGROWING procedure as the splitting criterion.

3.3 DE-ADT method to build axis-parallel decision trees

Unlike the previous sections, where two approaches to induce oblique DTs are described, the DE-ADT$_{\text{SPV}}$ method to build axis-parallel DTs is presented in this section. The DE-ADT$_{\text{SPV}}$ method implements a global search strategy where DE evolves a population of real-valued vectors encoding both the attributes and the threshold values associated with the numerical attributes evaluated in the internal nodes of a DT. The size of the real-valued vector is estimated a priori according to the characteristics of the dataset whose classification model is constructed, and a scheme to map a feasible axis-parallel DT from this vector is applied, using both the SPV rule and the training instances. A detailed description of the DE-ADT$_{\text{SPV}}$ elements is provided in the following paragraphs.

Algorithm 7 Completion of an oblique DT using both pT^i and the training instances.

```
function ObliqueDTCompletion($pT^i, \iota, \tau$)
    Input: The partial DT ($pT^i$), the training set ($\iota$), and the threshold value used to assign a leaf node ($\tau$).
    Output: The DT ($T^i$) mapped from an individual in the population.
    $h \leftarrow$ Hyperplane assigned to the root node of $pT^i$
    $\omega' \leftarrow (h, \iota)$                    ▷ The training set is assigned to the root node
    $V' \leftarrow \{\omega'\}$
    $E' \leftarrow \varnothing$
    $Q \leftarrow$ Empty queue                          ▷ FIFO queue used to traverse the DT
    Enqueue($Q, \omega'$)
    while $Q$ is not empty do
        $\omega \leftarrow$ Dequeue($Q$)
        $h \leftarrow$ Hyperplane assigned to $\omega$
        $\phi \leftarrow$ Instances set assigned to $\omega$
        $\Phi \leftarrow$ InstanceSetPartition($\phi, h$)    ▷ Instance subsets obtained by classifying $\phi$ using $h$
        for each $j \in \{1,2\}$ do
            $\zeta \leftarrow$ MajorityClass($\phi^j$)      ▷ Majority class label of the instances in $\phi^j$
            $\psi \leftarrow$ The number of instances in $\phi^j$ with $\zeta$ as class label
            if $(\mathcal{N}^+(\omega) \neq \varnothing \wedge |\phi^j| \neq \psi \wedge |\phi^j| > \tau)$ then
                $h \leftarrow$ Hyperplane of $\mathcal{N}_j^+(\omega)$    ▷ The hyperplane the $j$-th successor node of $\omega$
                $\omega_j \leftarrow (h, \phi^j)$          ▷ Internal node with the $j$-th instance subset of $\phi$
                Enqueue($Q, \omega_j$)
            else
                $\omega_j \leftarrow (\zeta, \phi^j)$      ▷ Leaf node with $\zeta$ as class label
            end if
            $V' \leftarrow V' \cup \{(\omega_j)\}$
            $E' \leftarrow E' \cup \{(\omega, \omega_j)\}$
        end for
    end while
    $T^i \leftarrow (G(V', E'), \omega')$
    return $T^i$                                        ▷ DT with both internal and leaf nodes
end function
```

3.3.1 Linear encoding scheme of candidate solutions

The linear encoding scheme proposed in this work associates each parameter of an individual with each attribute and with each threshold value used in the test conditions of an axis-parallel DT. Each test condition of an axis-parallel DT evaluates only one attribute to divide the training set. If one categorical attribute is evaluated, the training set is split into as many subsets as values there are in the domain of the attribute. On the other hand, if the evaluated attribute has numerical values, a threshold value is used to split the training set into two subsets, and the DTI method must determine a suitable threshold value optimizing some splitting criterion. Therefore, if the vectors y^i and z^i are used to represent the sequence of attributes and the sequence of threshold values, respectively, an individual x^i in the population is the concatenation of y^i followed by z^i, i.e., $x^i = y^i \frown z^i$, as is shown in Fig. 3.6. Furthermore, if n_y and n_z are the numbers of elements in y^i and z^i, respectively, the size of x^i is $n = n_y + n_z$.

Algorithm 8 General structure of DE-ODT method.

```
procedure DE-ODT(trainingSet, CR, F, NP, $\tau$)
    Input: The training set (trainingSet), the DE parameters (CR, F and NP), and the threshold value
           used to assign a leaf node ($\tau$).

    $(a, c, \iota) \leftarrow$ READTRAININGSET(trainingSet)
    $d \leftarrow |a|$                                              ▷ Number of attributes
    $n_e \leftarrow$ Number of estimated internal nodes computed using (3.3)
    $n \leftarrow n_e(d+1)$                                         ▷ Size of parameters vector
    $x^{\text{best}} \leftarrow$ DIFFERENTIALEVOLUTION(CR, F, NP, $n$)
    $w \leftarrow$ OBLIQUENVCONSTRUCTION($x^{\text{best}}$)         ▷ Nodes vector
    $pT \leftarrow$ OBLIQUEDTCONSTRUCTION($w$)                      ▷ DT with only internal nodes
    $T \leftarrow$ OBLIQUEDTCOMPLETION($pT, \iota, \tau$)           ▷ Complete DT
    $T \leftarrow$ OBLIQUEDTREFINEMENT($T, \tau$)                   ▷ Refined DT
    $T \leftarrow$ OBLIQUEDTPRUNING($T, \tau$)                      ▷ Pruning DT
end procedure
```

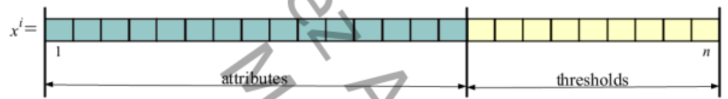


Figure 3.6: The structure of one individual to encode a sequence of attributes and a sequence of threshold values.

The DE-ADT$_{\text{SPV}}$ method uses the same scheme described in the DE-ODT procedure to estimate the height of a complete binary DT. The DE-ADT$_{\text{SPV}}$ method applies the estimated heights defined in equations 3.1 and 3.2 to calculate the size of y^i as follows:

$$n_y = 2^{\max\{H_i, H_l\}} - 1. \tag{3.7}$$

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

$$p_j = j \bmod d. \tag{3.8}$$

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

$$d_r = \left|\left\{k \in \{1, \ldots, d\} : D(a_k) \subseteq \mathbb{R}\right\}\right|, \tag{3.9}$$

n_z is obtained using the following equation:

$$n_z = d_r \left\lfloor \frac{n_y}{d} \right\rfloor + \left|\left\{k \in \{1, \ldots, d\} : D(a_k) \subseteq \mathbb{R} \wedge d \left\lfloor \frac{n_y}{d} \right\rfloor + k \leq n_y\right\}\right|. \tag{3.10}$$

The first term of (3.10) refers the number of numerical attributes used when a is entirely associated with y^i, and the second one represents the number of these attributes used when a is partially associated with y^i.

As an example, if a hypothetical dataset with four attributes (two categorical and two numerical attributes) and three class labels is used to induce a DT, then $d = 4$ and $s = 3$. In this case, $h_i = \lceil \log_2(5) + 1 \rceil =$

Algorithm 9 Refinement of a DT.

```
function ObliqueDTRefinement($T^i, \tau$)
    Input: The DT ($T^i$) and the threshold value used to assign a leaf node ($\tau$).
    Output: The refined DT ($T^i$).

    $\omega' \leftarrow$ Root node of $T^i$
    $V' \leftarrow$ Nodes of $T^i$
    $E' \leftarrow$ Edges of $T^i$
    $Q \leftarrow$ Empty queue
    Enqueue($Q, \omega'$)
    while $Q$ is not empty do
        $\omega \leftarrow$ Dequeue($Q$)
        if $\left|\mathbb{N}^+(\omega \in V)\right| \neq \varnothing$ then
            for each $j \in \{1,2\}$ do
                $\omega_j \leftarrow \mathbb{N}_j^+(\omega \in V')$                    ▷ $j$-th successor node of $\omega$.
                Enqueue($Q, \omega_j$)
            end for
        else                                                          ▷ The node is labeled as a leaf node
            $\phi \leftarrow$ Instances set in $\omega$
            $\zeta \leftarrow$ MajorityClass($\phi$)                         ▷ $\zeta$ is the majoritatian class label in $\phi$
            $\psi \leftarrow$ The number of instances in $\phi$ with $\zeta$ as class label
            if $(|\phi| \neq \psi \wedge |\phi| > \tau)$ then
                $(V', E') \leftarrow (V', E') \cup$ ObliqueTreeGrowing($\omega, \tau$)    ▷ $\omega$ is replaced by a sub-tree
            end if
        end if
    end while
    $T^i \leftarrow \left(G\left(V', E'\right), \omega'\right)$
    return $T^i$
end function
```

4 and $h_l = \lceil \log_2(3) + 1 \rceil = 3$. Finally, $n_y = 2^{\max\{4,3\}} - 1 = 15$. This implies that four attributes are associated with 15 elements of y^i. Fig. 3.7 shows an example applying the attributes mapping scheme with the hypothetical dataset, in which the positions of the attributes vector (a_1, a_2, a_3, a_4) are associated three times in complete form and once in partial form in y^i through p. Since y^i has assigned seven numerical attributes, $n_z = 7$ and x^i represents a sequence of 15 attributes, followed by a sequence of seven threshold values. In Fig. 3.8 is shown as x^i is completed with the threshold values.

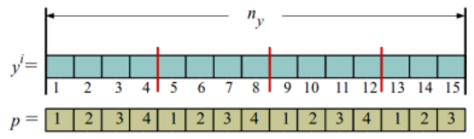


Figure 3.7: An example of the application of the attributes mapping schema.

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

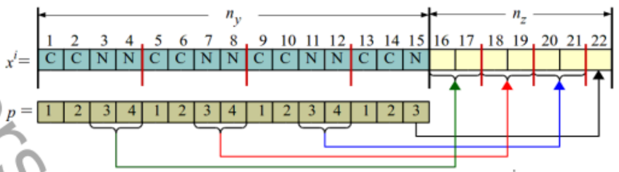


Figure 3.8: An example of the construction of x^i.

3.3.2 Induction of feasible axis-parallel decision trees

The DE-ADT$_{\text{SPV}}$ method uses one procedure similar to that described for the DE-ODT method, to map an axis-parallel DT from an individual in the population. Fig. 3.2 shows a graphical representation of this procedure.

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

$$o^i_j = \min\left\{k \in \{1,\dots,n_y\} \setminus \{o^i_1,\dots,o^i_{j-1}\} : y^i_k = \min\left\{y^i_l : l \in \{1,\dots,n_y\} \setminus \{o^i_1,\dots,o^i_{j-1}\}\right\}\right\}. \quad (3.11)$$

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

$$q = d_r \left\lfloor \frac{j}{d} \right\rfloor + \left| \left\{k \in \{1,\dots,d\} : D(a_k) \subseteq \mathbb{R} \wedge k \le p_{o^i_j}\right\} \right|, \quad (3.12)$$

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

$$t^i_q = \min\{D(a_{p_{o^i_j}})\} + \frac{(z^i_q - x^{\min}_j)\left(\max\{D(a_{p_{o^i_j}})\} - \min\{D(a_{p_{o^i_j}})\}\right)}{x^{\max}_j - x^{\min}_j}. \quad (3.13)$$

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

$$w^i_j = \begin{cases} \left(a_{p_{o^i_j}}, t^i_q\right) & \text{if } D(a_{p_{o^i_j}}) \subseteq \mathbb{R}, \\ \left(a_{p_{o^i_j}}\right) & \text{otherwise.} \end{cases} \quad (3.14)$$

Algorithm 10 outlines the process to build w^i from x^i. Once w^i is completed, it is used to create a partial DT with only internal nodes.

Fig. 3.9 shows an example of the application of SPV rule to build o^i with the positions of y^i using the hypothetical dataset. According to this rule, the smallest value of y^i is located at position 13, so this

Algorithm 10 Algorithm to build w^i from x^i.

```
function NVConstruction($x^i$)
    Input: The real-valued parameter vector ($x^i$).
    Output: The nodes vector ($w^i$)
    .
    $y^i \leftarrow (x^i_1, x^i_2, \ldots, x^i_{n_y})$                                      ▷ Sequence of attributes
    $z^i \leftarrow (x^i_{n_y+1}, x^i_{n_y+2}, \ldots, x^i_{n_y+n_z})$                      ▷ Sequence of threshold values
    for each $j \in \{1, \ldots, n_y\}$ do
        $o^i_j \leftarrow$ $j$-th minor element of $y^i$ determined using (3.11)
    end for
    for each $j \in \{1, \ldots, n_y\}$ do
        $k \leftarrow o^i_j$
        if $D(a_{p_k}) \subseteq \mathbb{R}$ then
            $q \leftarrow$ Location in $z^i$ of the threshold value associated with $a_{p_k}$ computed by (3.12)
            $t^i_q \leftarrow$ Threshold value of $a_{p_k}$ obtained using (3.13)
            $w^i_j \leftarrow (a_{p_k}, t^i_q)$                              ▷ Internal node using a numerical attribute
        else
            $w^i_j \leftarrow (a_{p_k})$                                     ▷ Internal node using a categorical attribute
        end if
    end for
    return $w^i$                                                             ▷ Vector of internal nodes
end function
```

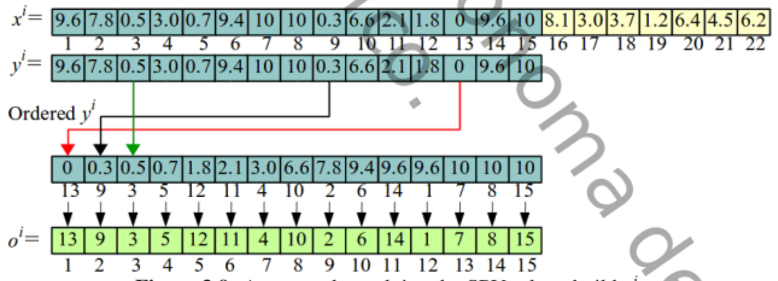


Figure 3.9: An example applying the SPV rule to build o^i.

position is assigned as the first element of o^i, the second smallest value of y^i is located at position 9, so this position is assigned as the second element of o^i, and so on. Fig. 3.10 shows an example of the construction of t^i from z^i. The domains of attributes in the hypothetical dataset are: $a_1 = \{\alpha, \beta\}$, $a_2 = \{\gamma, \kappa, \rho\}$, $a_3 = [1.5, 3.8]$, and $a_4 = [0.6, 9.8]$. In accordance with Eq. (3.12) and Eq. (3.13), p is used to associate each threshold value with its corresponding numerical attribute. Finally, in Fig. 3.11 is shown an example of the construction of w^i from both o^i and t^i. In this figure is observed that the first two positions in w^i are related with the categorical attribute a_1, and the numerical attribute a_3 along with the first threshold value in t^i are located in the third position of w^i.

2. **Partial decision tree construction:** The construction of a partial DT from w^i is very similar to the one described for the DE-ODT method, but several considerations must be applied when an internal

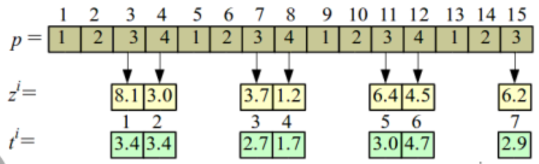


Figure 3.10: An example of the construction of t^i.

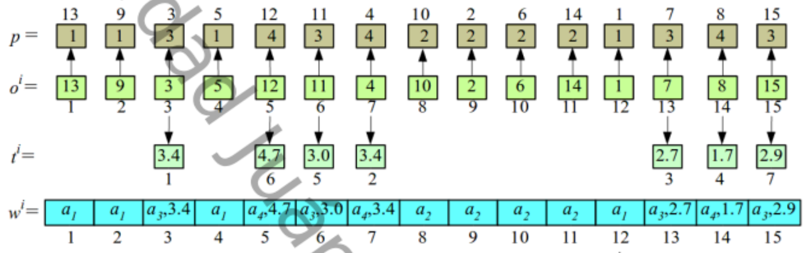


Figure 3.11: An example of the construction of w^i.

node use only one attribute in its test condition. Since the evaluated attribute can be numerical or categorical, the number of successor nodes b of an internal node is calculated based on the domain of the attribute used in its test condition, as follows:

$$b = \begin{cases} 2 & \text{if } D(\alpha) \subseteq \mathbb{R}, \\ |D(\alpha)| & \text{otherwise,} \end{cases} \tag{3.15}$$

where α is the attribute assigned in w^i.

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

R_1: A categorical attribute can only be evaluated once in each branch of the tree.

R_2: A numerical attribute can be evaluated several times in the same branch of the tree if and only if it uses coherent threshold values.

R_3: The successor nodes of one internal node with two branches of the tree cannot use the same categorical attribute.

Therefore, when an element in w^i does not satisfy the previous rules, it is not used to create an internal node, and the procedure continues analyzing the next item in it. Algorithm 11 shows the steps applied to create pT^i from w^i. In this algorithm can be observed that V is the set of valid internal nodes of pT^i, and E is the set of edges representing each possible outcome of each test condition in pT^i.

Fig. 3.12 shows the pT^i constructed from w^i in which is observed that only seven elements in w^i have been used to build the internal nodes of pT^i. The elements w^i_2, w^i_4 and w^i_{12} do not satisfy the rule 1 due to they encode internal nodes evaluating the categorical attribute a_1, but it is evaluated in the root node of pT^i, and none of its branches must re-evaluate this attribute. Furthermore, the elements w^i_6, w^i_{14} and w^i_{15} do not satisfy the rule 2. The internal node encoded by w^i_6 must be a successor node of that encoded by w^i_3 but, as the test condition of w^i_6 ($a_3 > 3.0$) is not congruent with that of w^i_3 ($a_3 > 3.4$),

Algorithm 11 Construction of pT^i from w^i.

```
function DTCONSTRUCTION($w^i$)
    Input: The nodes vector ($w^i$).
    Output: The partial DT with only feasible internal nodes ($pT^i$).
    $V \leftarrow \{w_1^i\}$                                  ▷ Set of internal nodes of $pT^i$
    $E \leftarrow \varnothing$                    ▷ Set of edges of $pT^i$ used to define the successor nodes
    for each $j \in \{1,\ldots,n_y\}$ do
        if $w_j^i \in V$ then
            $b \leftarrow$ Number of possible successor nodes of $w_j^i$ using (3.15)
            $k \leftarrow j+1$
            while $k \leq n_y \wedge \left|\mathcal{N}^+(w_j^i \in V)\right| < b$ do     ▷ Insertion of valid successor nodes
                $\alpha \leftarrow$ Attribute assigned in $w_k^i$
                if $\alpha$ satisfies $R_1$ and $R_2 \wedge w_k^i$ satisfies $R_3$ then
                    $V \leftarrow V \cup \{w_k^i\}$
                    $E \leftarrow E \cup \{(w_j^i, w_k^i)\}$
                end if
                $k \leftarrow k+1$
            end while
        end if
    end for
    $pT^i \leftarrow (G(V,E), w_1^i)$
    return $pT^i$                                ▷ partial DT with only internal nodes
end function
```

the procedure does not use this node. The same reasoning applies to elements w_{14}^i and w_{15}^i since they can not be used to construct successor nodes of the internal node encoded by w_6^i. Finally, the elements w_{10}^i and w_{11}^i are not used in the construction process due to they encode internal nodes that do not satisfy the rule 3. The internal node encoded by w_{10}^i cannot be assigned as a successor node of the one encoded by w_5^i, due to it evaluates the same categorical attribute used by an internal node previously assigned as successor node of that node. The same reasoning applies to the internal node encoded by the element w_{11}^i.

3. **Decision tree completion:** Algorithm 12 summarizes the process to complete the DT from pT^i. Fig. 3.13 shows an example of the T^i completion through this mapping scheme using the hypothetical dataset. In this figure is shown that the nodes five and eight have assigned instance sets with the same class label so that they are labeled as leaf nodes, and the successor nodes of the node number five are removed from pT^i. On the other hand, the node number seven is the ending node of a branch so that its instance set is split using its test condition. The instance subsets produced are assigned to two new leaf nodes, and the majority class is assigned as the class label of each one of them. In this figure can be observed that T^i has three internal nodes and four leaf nodes.

3.3.3 General structure of the DE-ADT$_{\text{SPV}}$ method

Algorithm 13 shows the structure of the DE-ADT$_{\text{SPV}}$ method proposed in this work. First, the DE-ADT$_{\text{SPV}}$ method uses the READTRAININGSET method to get the attributes vector a, the vector of class labels c, and the instances set ι. Next, the values of d and s are computed, as well as the values of n_y, n_z, and n used to

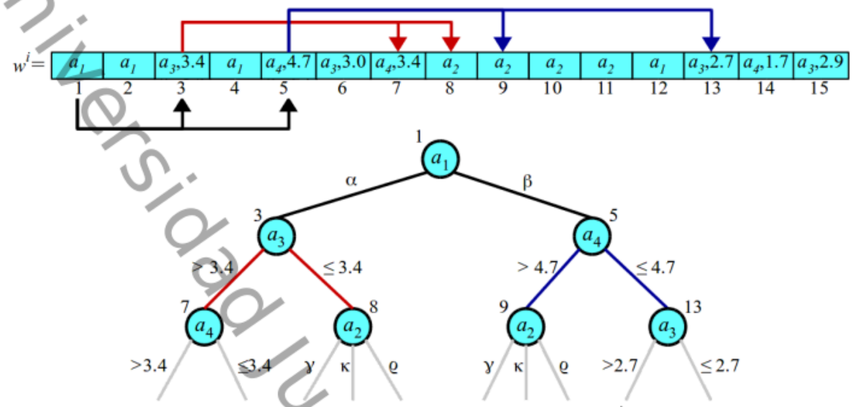


Figure 3.12: Parcial DT constructed from w^i.

build the initial population of candidate solutions. Then, the DE algorithm evolves this population to obtain the best candidate solution x^{best}. Finally, a near-optimal DT is constructed applying the procedures described in the previous paragraphs. The DE-ADT$_{\text{SPV}}$ method refines the DT by replacing non-optimal leaf nodes with sub-trees. Finally, the oblique DT is pruned to reduce the possible overfitting generated by applying this refinement.

As in the DE-ODT method, the F parameter gradually decreases as the evolutionary process implemented in the in the DE-ADT$_{\text{SPV}}$ method progresses. Also, the parameters in a vector representing an axis-parallel DT can be constrained or not. Unconstrained parameters are associated with the sequence of attributes, and constrained elements represent the threshold values used to build the test conditions with numerical attributes. When a parameter value violates a constraint, it is adjusted to the midpoint between its previous value and the boundary-value of the violated constraint as follows:

$$u_j^i \leftarrow \begin{cases} \frac{1}{2}\left(x_j^i + x_j^{\max}\right) & \text{if } u_j^i > x_j^{\max}, \\ \frac{1}{2}\left(x_j^i + x_j^{\min}\right) & \text{if } u_j^i < x_j^{\min}, \\ u_j^i & \text{otherwise.} \end{cases} \tag{3.16}$$

This mechanism to handle constraints allows asymptotically approach the space boundaries [276].

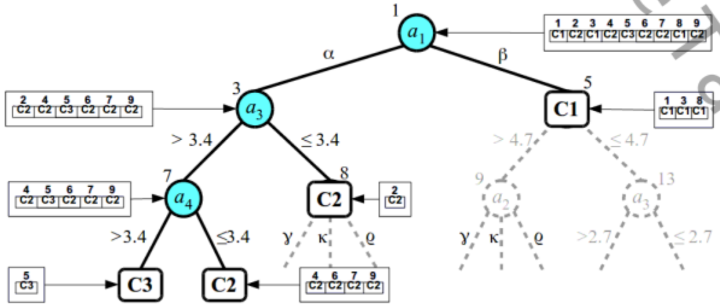


Figure 3.13: DT completed using the training set.

Algorithm 12 Completion of a DT using both pT^i and the training set.

```
function DTCOMPLETION($pT^i, \iota, \tau$)
    Input: The partial DT ($pT^i$), the training set ($\iota$), and the threshold value used to assign a leaf node ($\tau$).
    Output: The DT ($T^i$) mapped from an individual in the population.
    $\alpha \leftarrow$ Attribute asigned to the root node of $pT^i$
    $\omega' \leftarrow (\alpha, \iota)$                          ▷ The training set is assigned to the root node
    $V' \leftarrow \{\omega'\}$
    $E' \leftarrow \varnothing$
    $Q \leftarrow$ Empty queue                                   ▷ FIFO queue used to traverse the DT
    ENQUEUE($Q, \omega'$)
    while $Q$ is not empty do
        $\omega \leftarrow$ DEQUEUE($Q$)
        $\alpha \leftarrow$ Attribute asigned to $\omega$
        $\phi \leftarrow$ Instances set assigned in $\omega$
        $\varphi \leftarrow |\mathcal{N}^+(\omega \in V)|$       ▷ Number of succesor nodes assigned to $\omega$ in $pT^i$
        $b \leftarrow$ Number of successor nodes of $\omega$ using (3.15).
        $\Phi \leftarrow$ INSTANCESETPARTITION($\phi, \alpha$)   ▷ Instance subsets obtained by classifying $\phi$ using $\alpha$
        for each $j \in \{1, \dots, b\}$ do
            $\zeta \leftarrow$ MAJORITYCLASS($\phi^j$)           ▷ Majority class label of $j$-th instances set in $\Phi$
            $\psi \leftarrow$ The number of instances in $\phi^j$ with $\zeta$ as class label
            if ($j \leq \varphi \wedge |\phi^j| \neq \psi \wedge |\Phi^j| > \tau$) then
                $\alpha \leftarrow$ The attribute used by $\mathcal{N}_j^+(\omega)$
                $\omega_j \leftarrow (\alpha, \phi^j)$           ▷ Internal node with the $j$-th instance subset of $\Phi$
                ENQUEUE($Q, \omega_j$)
            else
                $\omega_j \leftarrow (\zeta, \phi^j)$            ▷ Leaf node with $\zeta$ as class label
            end if
            $V' \leftarrow V' \cup \{(\omega_j)\}$
            $E' \leftarrow E' \cup \{(\omega, \omega_j)\}$
        end for
    end while
    $T^i \leftarrow (G(V', E'), \omega')$
    return $T^i$                                               ▷ DT with both internal and leaf nodes
end function
```

3.4 Final remarks

The use of DE to induce oblique DTs is a natural approach to apply this MH since it evolves the hyperplane coefficients in the case of implementing the strategy of recursive partitioning, and of a set of hyperplanes when it comes to conducting a global search to find one near-optimal oblique trees. On the other hand, inducing axis-parallel DTs is a more difficult task since it is necessary to define a scheme to map the DT elements (attributes, thresholds for numerical attributes, and class labels) within a real-valued vector. Since the assignment of attributes is a problem of ordering values, it is possible to apply some technique to map a real-valued vector within an ordered sequence of integers.

Algorithm 13 General structure of the DE-ADT$_{\text{SPV}}$ method.

```
procedure DE-ADT_SPV(trainingSet, CR, F, NP, $\tau$)
    Input: The training set (trainingSet), the DE parameters (CR, F and NP), and the threshold value used to assign a leaf node ($\tau$).
    $(a, c, \iota) \leftarrow$ READTRAININGSET(trainingSet)
    $d \leftarrow |a|$                                                          ▷ Number of attributes
    $s \leftarrow |c|$                                                          ▷ Number of class labels
    $n_y \leftarrow$ Number of estimated internal nodes computed using (3.7)
    for each $j \in \{1, \ldots, n_y\}$ do
        $p_j \leftarrow$ Position of an attribute in $a$ using (3.8)
    end for
    $n_z \leftarrow$ Number of threshold values computed using (3.10)
    $n \leftarrow n_y + n_z$                                                    ▷ Size of parameters vector
    $x^{\text{best}} \leftarrow$ DIFFERENTIALEVOLUTION(CR, F, NP)
    $w \leftarrow$ NVCONSTRUCTION($x^{\text{best}}$)                            ▷ Nodes vector
    $pT \leftarrow$ DTCONSTRUCTION($w$)                                         ▷ DT with only internal nodes
    $T \leftarrow$ DTCOMPLETION($pT, \iota, \tau$)                              ▷ Complete DT
    $T \leftarrow$ DTREFINEMENT($T, \tau$)                                      ▷ Refined DT
    $T \leftarrow$ DTPRUNING($T, \tau$)                                         ▷ Pruned DT
end procedure
```

Chapter 4

Experimental study

The ideas of science germinate in a matrix of established knowledge gained by experiment; they are not lonesome thoughts, born in a rarified realm where no researcher has ever gone before

Seth Shostak

IN this chapter the experimental study carried out to analyze the performance of the three DE-based methods for DTI implemented in this thesis is detailed. First, a description of the datasets used in this study as well as the definition of the parameters of each method is given. Then, both the model validation technique used in the experiments and the statistical tests applied to evaluate the results obtained are outlined. Finally, a discussion about the performance of the DE-based methods is provided.

4.1 Experimental setup

4

The three DE-based methods are implemented in the Java language using the JMetal library [95]. The parameters used in the experiments are described in Table 4.1. The mutation scale factor is linearly decreased from 1.0 to 0.3 as the evolutionary process progresses, and the crossover rate is fixed in 0.9. At the beginning of the evolutionary process, if a high mutation scale factor is used, the mutated vector can be located in different areas of the search space, allowing a better exploration. As the process progresses, the target vectors approach each other, so that a reduced factor allows better exploitation of promising areas in the search space. A high crossover rate is used to allow a 1 great diversity of the trial vectors so that a better competition can be made in search of a better solution. Furthermore, following the suggestion of Storn and Price [340], the population size is adjusted to $5n$, but with 250 and 500 individuals as lower and upper bound, respectively. These bounds are used to ensure that the population is not so small as not to allow a reasonable exploration of the search space and is not so large as to impact the runtime of the algorithm. The fitness functions used in the DE-ODT method and the DE-ADT$_{\text{SPV}}$ algorithm compute the training accuracy of each DT in population, but the twoing rule is used as fitness measure in the OC1-DE method. Concerning the number of iterations of the process, the OC1-DE method uses a smaller number of generations than that those implementing a global search. This is because when DE is used to find a better hyperplane splitting the training instances, using a higher number of iterations promotes overfitting of the obtained solution, which is a near-optimal local solution, but affecting the predictive performance of the induced DT.

On the other hand, since the best DT obtained by the methods conducting a global search is refined with a procedure implementing a recursive partitioning strategy, it must be pruned to reduce the possible overfitting generated to applying this refinement. In these methods, and in the OC1-DE algorithm, the Error-Based Pruning (EBP) approach [303] is then applied since it produces DTs with an improved accuracy using only the training set [47]. Finally, the DE-based methods need to define a threshold value to determine whether a node should be labeled as one leaf node.

Table 4.1: Parameters used in the experiments conducted with the DE-based methods.

Parameter	**OC1-DE**	**DE-ODT, DE-ADT**
Number of generations	20	200
Fitness value	Twoing rule	Training accuracy
Mutation scale factor	$[0.3, 1.0]$	
Crossover rate	0.9	
Population size	$250 \leq 5n \leq 500$	
Pruning method	Error-based pruning	
Threshold value used to label leaf nodes	2 instances	

A benchmark of 28 datasets chosen from the UCI machine learning repository [225] is used to carry out the experimental study. These datasets have been selected as their attributes are numerical, categorical, or a combination of them, also their instances are classified into two or more classes, and most of them are imbalanced datasets. Table 4.2 shows the description of these datasets. Since oblique DTI methods seek a linear combination of attributes to construct hyperplanes, datasets with numerical attributes are only used for the experimental study of both OC1-DE and DE-ODT methods.

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

According to the previous paragraph, the DE-based methods are run for each iteration of the 10-fold CV procedure. Since the evolutionary process in the DE-based methods conducting a global search in the tree space use the training accuracy of each DT as its fitness value, the DTs in the final population are overfitted to the training set, so the DT with the best training accuracy would have a decreased test accuracy. In this work, with the aim of mitigating the effects of this overfitting, a subset of instances of the dataset is used to determine an independent accuracy for each DT in the final population and to select the best one. This value is referred in this work as the *selection accuracy*, so the DT with the best selection accuracy in the final population is used to calculate the test accuracy of the fold. To implement this strategy, 20% of the instances in the dataset are used to compute the selection accuracy, and the remaining are used in the CV procedure. Fig. 4.1 depicts this cross-validation scheme.

The CV procedure applied to estimate the test accuracy of the classifier constructed by the DE-based methods is similar to the one proposed by Murthy *et al.* [260]: For each fold, the selection accuracy (training accuracy for the OC1-DE method) of each DT in the population is calculated, and the DT with the best selection accuracy is used to compute the number of test instances correctly classified. The ratio between the

Table 4.2: Description of datasets used in the experiments.

Dataset	Instances	Numerical attributes	Categorical attributes	Classes	Class distribution
Datasets with only numerical attributes:					
glass	214	9	0	7	70|76|17|0|13|9|29
pima-diabetes	768	8	0	2	500|268
balance-scale	625	4	0	3	288|49|288
heart-statlog	270	13	0	2	150|120
iris	150	4	0	3	50|50|50
australian	690	14	0	2	307|383
ionosphere	351	34	0	2	126|225
wine	178	13	0	3	59|71|48
sonar	208	60	0	2	97|111
vehicle	846	18	0	4	212|217|218|199
liver-disorder	345	6	0	2	145|200
page-blocks	5473	10	0	5	4913|329|28|88|115
blood-t	748	4	0	2	570|178
breast-tissue-6	106	9	0	6	22|21|14|15|16|18
movement-libras	360	90	0	15	24 instances per class
parkinsons	195	22	0	2	48|147
seeds	210	6	0	3	70 instances per class
segment	2310	19	0	7	330 instances per class
ecoli	336	7	0	8	143|77|52|35|20|5|2|2
spambase	4601	57	0	2	1813|2788
Datasets with only categorical attributes:					
car	1728	0	6	4	1210|384|69|65
molecular-p	106	0	57	2	53|53
tic-tac-toe	958	0	9	2	332|626
Datasets with numerical and categorical attributes:					
lymph	148	3	15	4	2|81|61|4
credit-g	1000	7	13	2	700|300
cmc	1473	2	7	3	629|333|511
haberman	306	2	1	2	225|81
dermatology	366	1	33	6	112|61|72|49|52|20

correct classifications of all folds and the number of training instances is taken as the overall test accuracy of the classifier. Furthermore, the DT size is defined as the average number of leaf nodes of the DTs constructed by all folds.

In this study, the Friedman test [128] is applied to carry out a statistical analysis of the results produced by the DE-based methods when comparing them with those obtained by other classification methods. This non-parametric statistical test evaluates the statistical significance of the experimental results through computing the p-value without making any assumptions about the distribution of the analyzed data. This p-value is used to accept or to reject the null hypothesis H_0 of the experiment which holds that the performance of the compared algorithms does not present significant differences. If the p-value does not exceed a predefined significance level, H_0 is rejected, and the Bergmann-Hommel (BH) post-hoc test [161] is conducted to detect the differences between all existing pairs of algorithms. These statistical tests are applied using the *scmamp* R library [57].

The results obtained with the DE-based methods are compared with those achieved by a group of MH-based approaches for DTI and by several supervised learning methods available on the WEKA data mining software [149]. First, the accuracy of the DTs achieved by these methods are compared with those reported

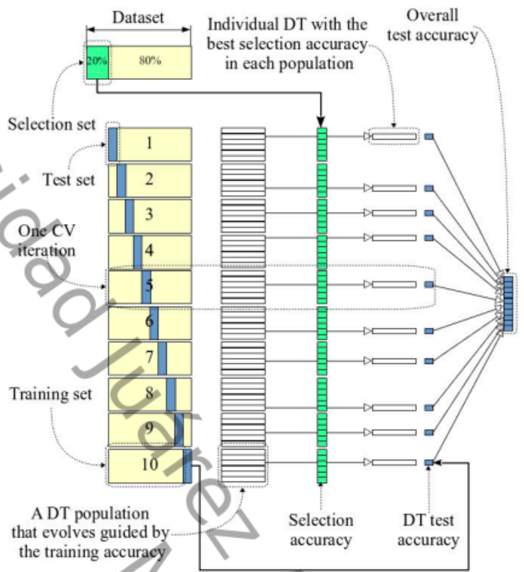


Figure 4.1: Adapted cross-validation procedure to determine the overall test accuracy of each dataset in the experimental study.

by the following MH-based algorithms:

- EFTI [373]: The experimental results in this study report a comparison of the EFTI method with several algorithms such as OC1, OC1-SA, OC1-GA, OC1-ES, GaTree, and HDBT.

- LEGAL-Tree [19]: The experimental results in this study report a comparison of two LEGAL-Tree versions: LA using the lexicographic analysis and PA using the Pareto dominance approach. This study also reports the results achieved by the GALE method.

These studies were selected since they apply the same sampling method that the one used in this thesis, and also since they report the better performance of the MH-based approaches for DTI.

Next, the accuracy and size of the DTs got by these methods are compared with those obtained by the following DTI methods:

- J48 [387]: It is a Java implementation of the C4.5 algorithm.

- sCART (SimpleCART) [46]: This is a Java implementation of the CART method.

Then, the accuracy of the DTs constructed with the DE-based methods are compared with those achieved using the following classification methods:

- NB (Naïve Bayes) [179]: This is a probabilistic classifier based on the Bayes theorem.

- MLP (Multilayer Perceptron) [258]: MLP is a feed-forward artificial neural network (FF-ANN) applying backpropagation to classify instances. The MLP has one or more hidden layers of nodes using sigmoid functions.

- RBF-NN (Radial Basis Function Neural Network) [120]: This is also an FF-ANN using a set of Gaussian radial basis functions (RBF) in its hidden layer.
- RF (Random Forest) [45]: It is an ensemble learning method constructing a multitude of DTs. RF uses a voting scheme to predict the class membership of new unclassified instances.

Finally, both height and size of the induced DT with the DE-based methods implementing a global search strategy are analyzed to evaluate the advantages of implementing the proposed scheme. The number of refinements of non-optimal leaf nodes is also assessed, due to the desire that the number of branches inserted in the evolved DT be reduced.

4.2 Results of the oblique decision trees induction

In this section, the results of DE-based methods inducing oblique DT are described and compared with those reported by other MH-based approaches that also produce this type of DTs.

4.2.1 Comparison with other MH-based approaches for DTI

In Table 4.3 and Figure 4.2 are shown the average accuracies of the DTs induced by several MH-based approaches for DTI as well as those achieved by the DE-based methods inducing oblique DTs. In Table 4.3, the best result for each dataset is highlighted with bold numbers, and the numbers in parentheses refer to the ranking reached by each method for each dataset. The last row in this table indicates the average ranking of each method. It is observed that the DE-based methods produce better results than those generated by the other MH-based approaches for DTI. In particular, the DE-ODT method obtains the best results from this experiments, as it yields higher average accuracies than those got by the compared algorithms in six of the eleven datasets.

Table 4.3: Average accuracies obtained by other MH-based approaches for DTI.

Dataset	**OC1**	**OC1-SA**	**OC1-GA**	**OC1-ES**	**HBDT**	**GATree**	**EFTI**	**DE-ODT**	**OC1-DE**
glass	62.04 (8)	64.32 (4)	63.91 (5)	62.52 (6)	62.51 (7)	52.00 (9)	70.82 (2)	68.97 (3)	**71.31** (1)
diabetes	73.03 (7)	74.35 (3)	71.47 (9)	74.28 (4)	71.51 (8)	73.58 (5)	74.94 (2)	**75.79** (1)	73.37 (6)
balance-scale	71.58 (9)	73.79 (5)	73.31 (6)	72.98 (7)	72.10 (8)	74.54 (4)	87.85 (3)	91.97 (2)	**93.92** (1)
heart-statlog	76.30 (5)	74.96 (7)	68.67 (9)	76.00 (6)	79.85 (3)	79.70 (4)	**81.28** (1)	81.11 (2)	74.11 (8)
iris	95.60 (4.5)	93.47 (9)	94.93 (6)	94.40 (7)	95.60 (4.5)	96.27 (3)	94.13 (8)	**97.17** (1)	96.73 (2)
australian	83.63 (7)	85.02 (4)	76.49 (9)	84.81 (5)	82.52 (8)	85.13 (3)	84.51 (6)	**85.61** (1)	85.20 (2)
ionosphere	88.26 (5.5)	89.29 (4)	90.10 (3)	88.26 (5.5)	88.24 (7)	84.86 (9)	86.39 (8)	**92.28** (1)	91.11 (2)
sonar	70.39 (8)	72.51 (6)	73.13 (5)	70.50 (7)	75.96 (3)	52.20 (9)	74.64 (4)	**79.34** (1)	77.65 (2)
vehicle	68.16 (6)	68.06 (7)	66.30 (8)	69.18 (4)	**74.51** (1)	60.76 (9)	68.75 (5)	71.33 (3)	72.32 (2)
liver-disorders	67.23 (5)	64.60 (8)	67.24 (4)	66.18 (6)	65.96 (7)	62.71 (9)	70.36 (2)	**71.16** (1)	67.63 (3)
page-blocks	97.05 (3)	97.00 (4.5)	96.61 (7)	**97.08** (1)	97.00 (4.5)	92.34 (9)	93.16 (8)	97.07 (2)	96.88 (6)
Average ranking	6.18	5.59	6.45	5.32	5.55	6.64	4.45	**1.64**	3.18

A statistical test of the experimental results is conducted to evaluate the performance of the DE-based methods. First, the Friedman test is run, and its resulting statistic value is 32.128 for nine methods and eleven datasets, which has a p-value of 8.837×10^{-5}. When evaluating this p-value with a significance level of 5%, H_0 is rejected. Next, the BH post-hoc test is applied to find all the possible hypotheses which cannot be rejected. In Table 4.4 is shown both the average rank (AR) of the results yielded by each method and the p-values computed by comparing the average accuracies achieved by the DE-based procedures versus

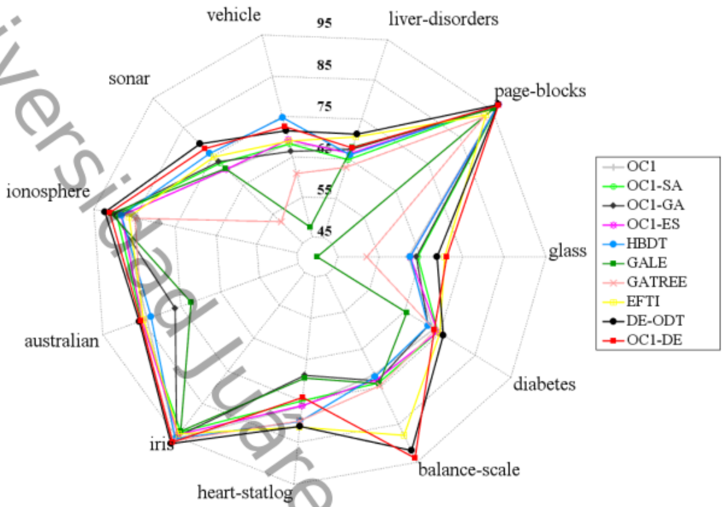


Figure 4.2: Graphical comparison of the average accuracies obtained by other MH-based approaches for DTI.

those obtained by the other MH-based approaches. The p-values highlighted with bold numbers indicate that H_0 is rejected for this pair of methods since they show different performance. Unadjusted p-values are calculated with the average ranks of the two methods being compared, as is described by Demšar in [85]. These values are used by the BH post-hoc test to compute the corresponding adjusted p-values. Table 4.4 shows that the DE-ODT method has a better performance than the other MH-based approaches since it has the lowest average rank (1.64) and its results are statistically different from six of the seven methods.

Table 4.4: p-values for multiple comparisons among othe MH-based approches for DTI and the DE-based methods.

Method	AR	OC1-DE		DE-ODT	
		Unadjusted	BH	Unadjusted	BH
OC1	6.10	1.0197e-02	1.6316e-01	9.9218e-05	**2.1828e-03**
OC1-SA	5.55	3.9110e-02	6.2576e-01	7.0797e-04	**1.5575e-02**
OC1-GA	6.45	5.0693e-03	1.0645e-01	3.6904e-05	**1.0333e-03**
OC1-ES	5.27	6.7328e-02	1.0000e+00	1.6164e-03	**3.5562e-02**
HBDT	5.45	4.2960e-02	6.8736e-01	8.1530e-04	**1.0179e-02**
GATree	6.37	3.0934e-03	8.6616e-02	1.8543e-05	**6.6756e-04**
EFTI	4.45	2.7575e-01	1.0000e+00	1.5806e-02	2.8452e-01
DE-OC1	1.64	-	-	1.8568e-01	1.0000e+00
DE-ODT	3.18	1.8568e.01	1.0000e+00	-	-

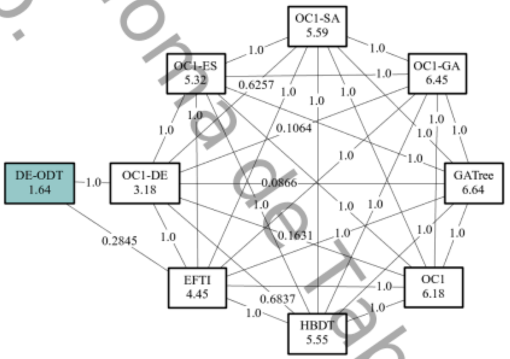


Figure 4.3: p-values graph of the MH-based approaches for DTI and the DE-based methods.

Figure 4.3 shows a graph where the nodes represent the compared methods and the edges joining two nodes indicate that the performance of these methods does not present significant differences. The values shown in the edges are the p-values computed by the BH post-hoc test. This figure is based on that obtained using the *scmamp* library, and in it is observed that the DE-based methods are not statistically different between them and that the DE-ODT method is statistically different with the other MH-based approaches. These statistical results indicate that the DE-ODT method is the better DTI method to build oblique DTs.

4.2.2 Comparison with DTI methods

In Table 4.5 and Figure 4.4 are shown the average accuracies of the DTs induced by the DTI algorithms as well as those achieved by the DE-based methods to induce oblique DTs. In this Table is observed that the DE-based methods produce better results than those generated by the other DTI algorithms, and also that the DE-ODT method induces oblique DTs with better accuracy than the trees built by the other DTI algorithms in twelve of the twenty datasets.

Table 4.5: Average accuracies obtained by the DTI algorithms and the DE-based methods.

Dataset	**J48**	**sCART**	**OC1-DE**	**DE-ODT**
glass	67.62 (4)	71.26 (2)	**71.31** (1)	68.97 (3)
diabetes	74.49 (3)	74.56 (2)	73.37 (4)	**75.79** (1)
balance-scale	77.82 (4)	78.74 (3)	**93.92** (1)	91.97 (2)
heart-statlog	78.15 (2)	78.07 (3)	74.11 (4)	**81.11** (1)
iris	94.73 (3)	94.20 (4)	96.73 (2)	**97.17** (1)
australian	84.35 (4)	85.19 (2.5)	85.19 (2.5)	**85.61** (1)
ionosphere	89.74 (3)	88.86 (4)	91.11 (2)	**92.28** (1)
wine	**93.20** (1)	89.49 (4)	92.58 (2)	91.88 (3)
sonar	73.61 (3)	70.67 (4)	77.65 (2)	**79.34** (1)
vehicle	72.28 (2)	69.91 (4)	**72.32** (1)	71.33 (3)
liver-disorders	65.83 (4)	66.64 (3)	67.63 (2)	**71.16** (1)
page-blocks	96.99 (2)	96.76 (4)	96.88 (3)	**97.07** (1)
blood-t	78.20 (2)	77.86 (3)	76.35 (4)	**78.70** (1)
breast-tissue-6	34.81 (3)	32.45 (4)	34.91 (2)	**38.85** (1)
movement-libras	69.31 (2)	65.64 (3)	**75.11** (1)	55.63 (4)
parkinsons	84.72 (4)	86.31 (3)	**87.95** (1)	86.43 (2)
seeds	90.90 (3.5)	90.90 (3.5)	**93.76** (1)	91.79 (2)
segment	**96.79** (1)	95.83 (3)	95.93 (2)	94.78 (4)
ecoli	82.83 (4)	83.15 (3)	83.51 (2)	**84.72** (1)
spambase	92.68 (2)	92.35 (3)	92.19 (4)	**93.94** (1)
Average ranking	2.825	3.250	2.175	1.750

First, the Friedman test is run, and its resulting statistic value is 27.661 for four methods and 20 datasets, which has a p-value of 4.24×10^{-5}. H_0 is then rejected, and the BH post-hoc test is applied. Table 4.6 shows the results of these tests, and Figure 4.5 shows the graph corresponding to these p-values. Table 4.6 shows that the DE-ODT method has a better performance than the other DTI methods since it has the lowest average rank (1.750) and its results are statistically different from these methods. These statistical results indicate that the DE-ODT method is the better DTI method to build oblique DT.

On the other hand, the average sizes of the DTs constructed by the DE-based algorithms and also of those induced by the J48 and the sCART methods are shown in Table 4.7 and in Figure 4.6. These results indicate that the DE-ODT method produces the most compact DTs. Also, it is observed that the size of the DTs built for the OC1-DE method has less complexity than those yielded by the J48 method.

4.2.3 Comparison with other classification methods

In Table 4.8 and Figure 4.7 are shown the average accuracies got by several classification methods as well as those obtained by the DE-based methods. In this Table can be observed that the RF algorithm and the MLP method construct more accurate classifiers than the others, and also that the DE-based procedures induce DTs with better accuracy than the models built by both the RBF-NN algorithm and the NB method.

The Friedman statistics computed by analyzing the results got by these six methods with 20 datasets is

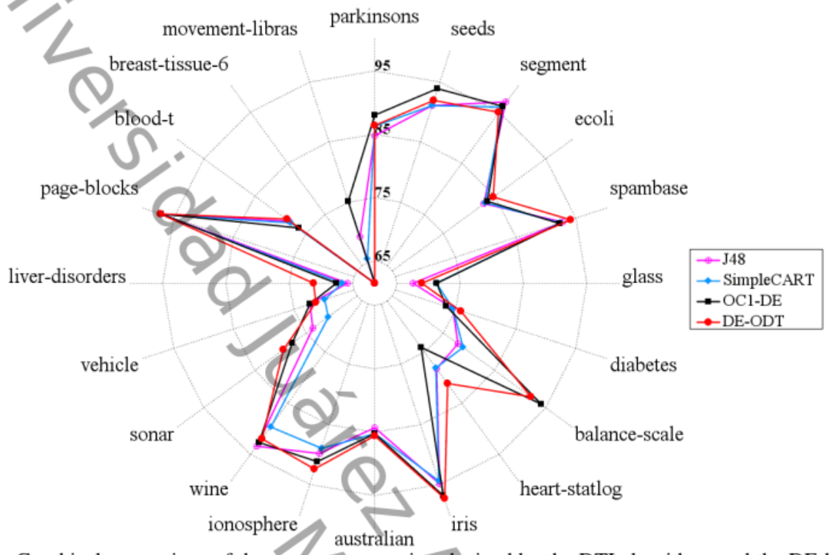


Figure 4.4: Graphical comparison of the average accuracies obtained by the DTI algorithms and the DE-based methods.

Table 4.6: p-values for multiple comparisons among DTI algorithms and the DE-based methods.

Method	AR	OC1-DE		DE-ODT	
		Unadjusted	BH	Unadjusted	BH
J48	2.825	1.1134e-01	1.1134e-01	8.4584e-03	**2.5375e-02**
sCART	3.250	8.4584e-03	**2.5375e-02**	2.3856e-04	**1.4131e-03**
OC1-DE	2.175	-	-	2.9786e-01	5.9572e-01
DE-ODT	1.750	2.9786e-01	5.9572e-01	-	-

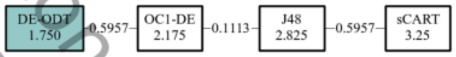


Figure 4.5: p-values graph of the DTI algorithms and the DE-based methods.

27.661, and the corresponding p-value is 4.24×10^{-5} so that H_0 is rejected. The BH post-hoc test is then applied to find all possible hypotheses that can not be refused. Table 4.9 shows the results of these tests, and Figure 4.8 shows the graph corresponding to these p-values.

The p-values obtained by the BH post-hoc test point out that the RF method is statistically different only with the RBF-NN algorithm and the NB method, and that both the MLP method and the DE-ODT procedure are statistically different with the NB method. The comparison between the remaining pairs of algorithms indicates that they have a similar performance. The RF method is the best ranked in this comparison, and the AR of the DE-ODT procedure places it as the third best classification method.

4.3 Results of the DE-ADT method

In this section, the results of the DE-ADT$_{\text{SPV}}$ variants are described and compared with those got by other classification methods. Two DE-ADT$_{\text{SPV}}$ variants are evaluated in this experimental study:

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

Table 4.7: Average DT sizes obtained by the DTI methods.

Dataset	J48		sCART		OC1-DE		DE-ODT	
glass	23.58	(4)	**8.00**	(1)	21.61	(3)	11.08	(2)
diabetes	22.20	(3)	**3.00**	(1)	41.55	(4)	14.77	(2)
balance-scale	41.60	(4)	13.00	(2)	15.24	(3)	**5.01**	(1)
heart-statlog	17.82	(4)	16.00	(2)	17.43	(3)	**7.23**	(1)
iris	4.64	(3)	5.00	(4)	**3.00**	(1)	3.37	(2)
australian	25.75	(4)	**5.00**	(1)	21.90	(3)	15.64	(2)
ionosphere	13.87	(4)	**3.00**	(1)	7.20	(2)	7.73	(3)
wine	5.30	(3)	5.00	(2)	5.48	(4)	**4.71**	(1)
sonar	14.45	(4)	10.00	(2)	10.24	(3)	**6.13**	(1)
vehicle	69.50	(3)	80.00	(4)	56.74	(2)	**44.25**	(1)
liver-disorders	25.51	(4)	**3.00**	(1)	22.65	(3)	6.60	(2)
page-blocks	42.91	(4)	**22.00**	(1)	38.70	(3)	24.56	(2)
blood-t	**6.50**	(1)	10.00	(3)	22.39	(4)	8.46	(2)
breast-tissue-6	22.45	(4)	**8.00**	(1)	14.09	(3)	8.97	(2)
movement-libras	47.52	(4)	30.00	(3)	**27.46**	(1)	29.07	(2)
parkinsons	10.24	(4)	7.00	(2)	7.11	(3)	**4.85**	(1)
seeds	7.42	(4)	6.00	(3)	4.78	(2)	**3.17**	(1)
segment	41.21	(4)	41.00	(3)	30.53	(2)	**27.91**	(1)
ecoli	18.59	(4)	15.00	(3)	12.57	(2)	**7.06**	(1)
spambase	103.37	(4)	75.00	(3)	74.42	(2)	**31.70**	(1)
Average ranking	3.65		2.15		2.65		1.55	

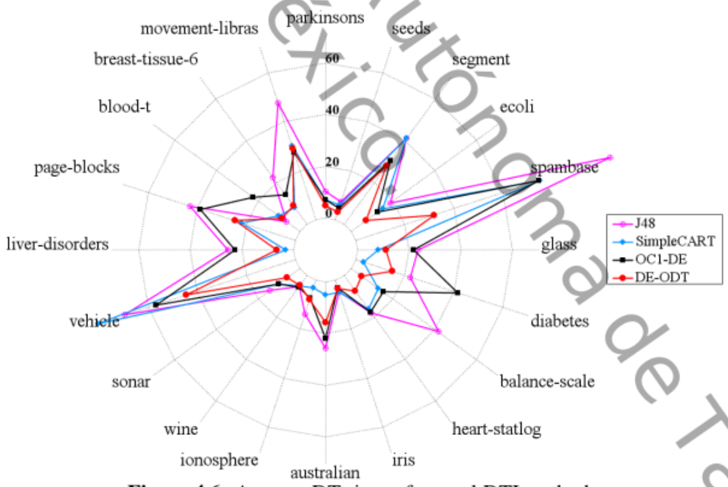


Figure 4.6: Average DT sizes of several DTI methods.

- DE-ADT$^{R}_{SPV}$: This variant returns the refined version of the DT with the best selection accuracy in the population.

4.3.1 Comparison with other MH-based approaches for DTI

In Table 4.10 is shown the average accuracies of the DTs induced by several MH-based approaches for DTI as well as those achieved by the $^{R}_{SPV}$ method. It is observed that the DE-based method produce better results

Table 4.8: Average accuracies obtained by several classification methods.

Dataset	NB		MLP		RBF-NN		RF		OC1-DE		DE-ODT	
glass	49.44	(6)	67.29	(4)	65.09	(5)	**79.95**	(1)	71.31	(2)	68.97	(3)
diabetes	75.76	(3)	74.75	(4)	74.04	(5)	**76.18**	(1)	73.37	(6)	75.79	(2)
balance-scale	90.53	(4)	90.69	(3)	86.34	(5)	81.71	(6)	**93.92**	(1)	91.97	(2)
heart-statlog	**83.59**	(1)	79.41	(5)	83.11	(2)	82.41	(3)	74.11	(6)	81.11	(4)
iris	95.53	(5)	96.93	(2)	96.00	(4)	94.73	(6)	96.73	(3)	**97.17**	(1)
australian	77.19	(6)	83.42	(4)	82.55	(5)	**86.77**	(1)	85.19	(3)	85.61	(2)
ionosphere	82.17	(6)	91.05	(5)	91.71	(3)	**93.39**	(1)	91.11	(4)	92.28	(2)
wine	97.47	(4)	**98.03**	(1.5)	97.70	(3)	**98.03**	(1.5)	92.58	(5)	91.88	(6)
sonar	67.69	(6)	81.59	(2)	72.60	(5)	**84.47**	(1)	77.65	(4)	79.34	(3)
vehicle	44.68	(6)	**81.11**	(1)	65.35	(5)	75.14	(2)	72.32	(3)	71.33	(4)
liver-disorders	54.87	(6)	68.72	(3)	65.04	(5)	**72.99**	(1)	67.63	(4)	71.16	(2)
page-blocks	90.01	(6)	96.28	(4)	94.91	(5)	**97.54**	(1)	96.88	(3)	97.07	(2)
blood-t	75.28	(5)	78.46	(2)	78.22	(3)	73.62	(6)	76.35	(4)	**78.70**	(1)
breast-tissue-6	**46.42**	(1)	35.47	(5)	41.13	(3)	45.19	(2)	34.91	(6)	38.85	(4)
movement-libras	64.14	(5)	80.50	(2)	75.50	(3)	**82.89**	(1)	75.11	(4)	55.63	(6)
parkinsons	70.10	(6)	**91.44**	(1)	81.49	(5)	91.38	(2)	87.95	(3)	86.43	(4)
seeds	90.52	(6)	**95.24**	(1)	91.67	(5)	93.57	(3)	93.76	(2)	91.79	(4)
segment	80.17	(6)	96.21	(2)	87.31	(5)	**98.07**	(1)	95.93	(3)	94.78	(4)
ecoli	85.51	(2)	84.85	(3)	83.30	(6)	**86.25**	(1)	83.51	(5)	84.72	(4)
spambase	79.56	(6)	91.19	(4)	81.31	(5)	**95.65**	(1)	92.19	(3)	93.94	(2)
Average ranking	4.800		2.925		4.350		2.125		3.700		3.100	

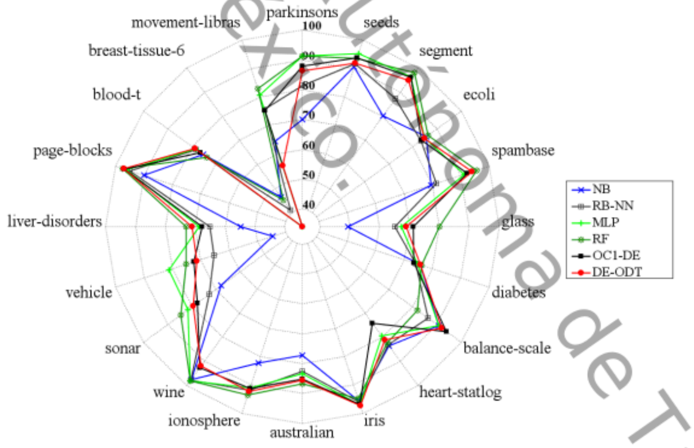


Figure 4.7: Graphical comparison of the average accuracies obtained by several classification methods.

than those generated by the other MH-based approaches for DTI in six of the eight datasets.

The Friedman statistics computed by analyzing the results got by these four methods is 14.36, and the corresponding p-value is 2.454×10^{-3} so that H_0 is rejected. Table 4.11 shows the adjusted p-values obtained by the post-hoc test, and the graph corresponding to these values is depicted in and Fig. 4.9. In this table 4.13 is shown that the DE-ADT$^{R}_{SPV}$ method has a better performance than the other MH-based approaches to induce axis-parallel DTs since it has the lowest average rank (1.25) and its results are statistically different from the others.

Table 4.9: p-values for multiple comparisons among several classification methods.

Method	AR	OC1-DE		DE-ODT	
		Unadjusted	BH	Unadjusted	BH
NB	4.800	6.2979e-02	3.7787e-01	4.0591e-03	**2.8414e-02**
MLP	2.925	1.9019e-01	7.6079e-01	7.6737e-01	8.9374e-01
RBF-NN	4.350	2.7118e-01	7.7679e-01	3.4610e-03	1.3844e-01
RF	2.125	7.7623e-03	5.4336e-03	0.9342e-02	3.9736e-01
OC1-DE	3.700	-	-	3.1049e-01	7.6079e-01
DE-ODT	3.100	3.1049e-01	7.6079e-01	-	-

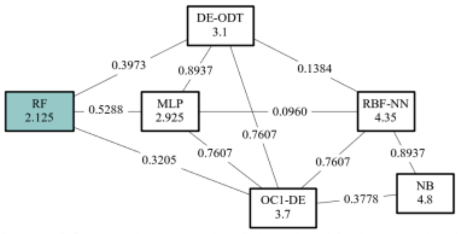


Figure 4.8: p-values graph of the classifiction methods.

Table 4.10: Average accuracies obtained by other MH-based approaches for DTI and the DE-ADT$^{R}_{SPV}$ method.

Dataset	LEGAL-Tree$_{LA}$		LEGAL-Tree$_{PA}$		GALE		DE-ADT$^{R}_{SPV}$	
glass	67.00	(3)	70.00	(2)	59.00	(4)	**71.67**	(1)
diabetes	75.00	(3)	**76.00**	(1)	74.00	(4)	75.48	(2)
balance-scale	77.00	(3.5)	77.00	(3.5)	78.00	(2)	**80.00**	(1)
heart-statlog	79.00	(2)	78.00	(3.5)	78.00	(3.5)	**82.92**	(1)
iris	95.00	(2.5)	95.00	(2.5)	94.00	(4)	**97.25**	(1)
ionosphere	91.00	(2)	90.00	(3)	89.00	(4)	**92.88**	(1)
lymph	**80.00**	(1)	77.00	(3.5)	77.00	(3.5)	78.83	(2)
credit-g	72.00	(2)	71.00	(3.5)	71.00	(3.5)	**73.65**	(1)
Average ranking	2.825		3.250		2.175		1.750	

Table 4.11: p-values for multiple comparisons among other MH-based approaches for DTI and the DE-ADT$^{R}_{SPV}$ method.

Method	AR	DE-ADT$^{R}_{SPV}$	
		Unadjusted	BH
LEGAL-Tree$_{LA}$	2.38	8.1361e-02	1.9745e-01
LEGAL-Tree$_{PA}$	2.81	1.5494e-02	**4.6482e-02**
GALE	3.56	3.4030e-04	**2.0418e-03**
DE-ADT$^{R}_{SPV}$	1.25	-	-

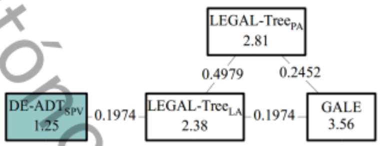


Figure 4.9: p-values graph of multiple comparisons among other MH-based approaches for DTI and the DE-ADT$^{R}_{SPV}$ method.

4.3.2 Comparison with DTI methods

In Table 4.12 and Fig. 4.10 are shown the average accuracies of the DTs induced by the DTI methods as well as those achieved by the DE-ADT$_{SPV}$ variants. It is observed that the DE-ADT$_{SPV}$ variants produce better results than those generated by the other DTI methods. In particular, the DE-ADT$^{R}_{SPV}$ variant obtains the best results from this experiment, as it yields higher average accuracies than those got by the compared DTI techniques in 16 datasets.

The Friedman statistics computed by analyzing the results got by these five methods 22.02, and the corresponding p-value is 6.461×10^{-5} so that H_0 is rejected. The BH post-hoc test is then applied to find all possible hypotheses which cannot be rejected. Table 4.13 shows this values, and the graph corresponding to these p-values is depicted in and Fig. 4.11. In Table 4.13 is shown that the DE-ADT$^{R}_{SPV}$ method has a better performance than the other DTI methods since it has the lowest average rank (1.35) and its results are statistically different from the others.

On the other hand, the average sizes of the DTs constructed by the DE-ADT$_{SPV}$ variants and also of those induced by J48 and sCART methods are shown in Table 4.14 and Fig. 4.12. These results indicate

Table 4.12: Average accuracies obtained by the DTI algorithms and the DE-ADT variants.

Dataset	**J48**		**sCART**		**DE-ADT$^B_{SPV}$**		**DE-ADT$^R_{SPV}$**	
car	92.22	(2)	**97.37**	(1)	81.19	(4)	90.53	(3)
molecular-p	79.06	(3)	47.17	(4)	80.12	(2)	**84.88**	(1)
tic-tac-toe	85.28	(2)	**93.57**	(1)	75.49	(4)	84.46	(3)
glass	67.62	(4)	71.26	(2)	67.82	(3)	**71.67**	(1)
diabetes	74.49	(4)	74.56	(3)	74.73	(2)	**75.48**	(1)
balance-scale	77.82	(3)	78.74	(2)	75.70	(4)	**80.00**	(1)
heart-statlog	78.15	(3)	78.07	(4)	81.81	(2)	**82.92**	(1)
iris	94.73	(3)	94.20	(4)	95.92	(2)	**97.25**	(1)
australian	84.35	(4)	85.19	(3)	85.57	(2)	**86.42**	(1)
ionosphere	89.74	(3)	88.86	(4)	89.97	(2)	**92.88**	(1)
wine	93.20	(3)	89.49	(4)	93.47	(2)	**94.37**	(1)
^{B} sonar	73.61	(3)	70.67	(4)	74.19	(2)	**78.74**	(1)
vehicle	**72.28**	(1)	69.91	(3)	65.49	(4)	72.21	(2)
liver-disorders	65.83	(4)	66.64	(3)	66.96	(2)	**70.33**	(1)
page-blocks	**96.99**	(1)	96.76	(2)	95.10	(4)	96.74	(3)
lymph	75.81	(4)	77.16	(2)	76.50	(3)	**78.83**	(1)
credit-g	71.25	(4)	73.43	(2)	71.63	(3)	**73.65**	(1)
cmc	51.44	(4)	55.21	(2)	54.18	(3)	**55.57**	(1)
haberman	72.16	(4)	73.24	(3)	74.29	(2)	**75.76**	(1)
dermatology	94.10	(3)	94.43	(2)	92.81	(4)	**95.70**	(1)
Average ranking		3.1		2.75		2.8		**1.35**

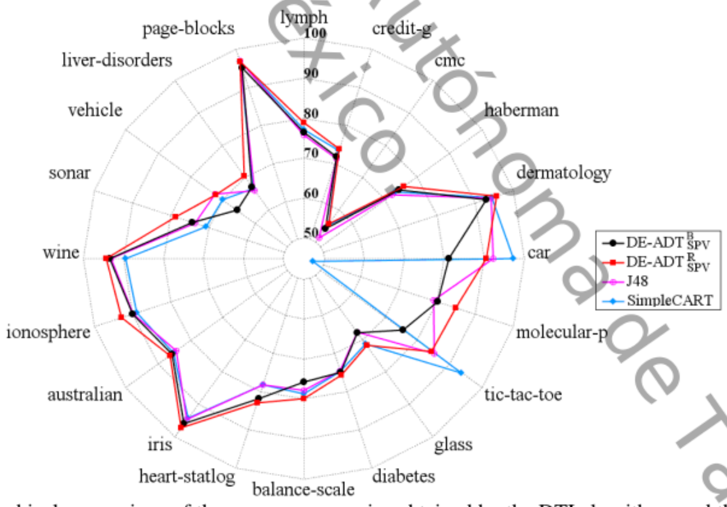


Figure 4.10: Graphical comparison of the average accuracies obtained by the DTI algorithms and the DE-ADT variants.

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

Table 4.13: p-values for multiple comparisons among the DTI algorithms and the DE-ADT variants.

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

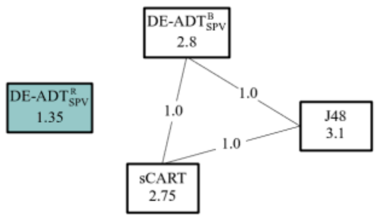


Figure 4.11: p-values graph of the DTI algorithms and the DE-ADT variants.

Table 4.14: Average DT sizes obtained by the DTI algorithms and the DE-ADT variants.

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

4.3.3 Comparison with other classification methods

In Table 4.15 and Fig. 4.13 are shown the average accuracies got by several classification methods as well as those obtained by the DE-ADT^{R}_{SPV} method. In this Table can be observed that the RF algorithm and the MLP method construct more accurate classifiers than the others, and also that the DE-ADT^{R}_{SPV} induces DTs with better accuracy than the models built by both the RBF-NN algorithm and the NB method.

The Friedman statistics computed by analyzing the results got by these five methods with 20 datasets is 10.4, and the corresponding p-value is 0.034222 so that H_0 is rejected. Table 4.16 shows the results of the post-hoc test, and Fig. 4.14 shows the graph corresponding to these p-values. The p-values obtained by the BH post-hoc test point out that the RF method is statistically different only with the RBF-NN algorithm and the NB method, and the comparison between the remaining pairs of algorithms indicates that they have a similar performance. The RF method is the best ranked in this comparison, and the AR of the DE-ADT^{R}_{SPV} variant places it as the third best classification method.

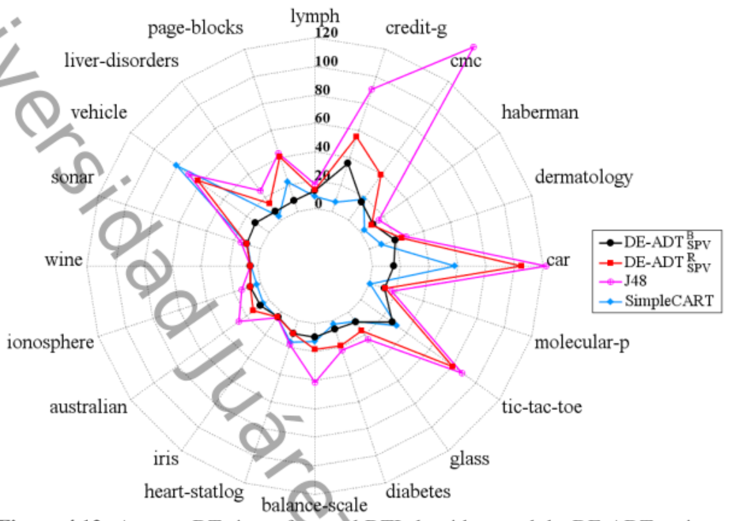


Figure 4.12: Average DT sizes of several DTI algorithms and the DE-ADT variants.

Table 4.15: Average accuracies obtained by several classification algorithms and the DE-ADT variants.

Dataset	**NB**		**MLP**		**RBF-NN**		**RF**		**DE-ADT$^{R}_{SPV}$**	
car	85.46	(5)	**99.41**	(1)	88.80	(4)	94.63	(2)	90.53	(3)
molecular-p	90.19	(3)	**91.70**	(1)	89.53	(4)	91.13	(2)	84.88	(5)
tic-tac-toe	69.65	(5)	**97.39**	(1)	70.88	(4)	96.93	(2)	84.46	(3)
glass	49.44	(5)	67.29	(3)	65.09	(4)	**79.95**	(1)	71.67	(2)
diabetes	75.76	(2)	74.75	(4)	74.04	(5)	**76.18**	(1)	75.48	(3)
balance-scale	90.53	(2)	**90.69**	(1)	86.34	(3)	81.71	(4)	80.00	(5)
heart-statlog	**83.59**	(1)	79.41	(5)	83.11	(2)	82.41	(4)	82.92	(3)
iris	95.53	(4)	96.93	(2)	96.00	(3)	94.73	(5)	**97.25**	(1)
australian	77.19	(5)	83.42	(3)	82.55	(4)	**86.77**	(1)	86.42	(2)
ionosphere	82.17	(5)	91.05	(4)	91.71	(3)	**93.39**	(1)	92.88	(2)
wine	97.47	(4)	**98.03**	(1.5)	97.70	(3)	**98.03**	(1.5)	9[illegible].37	(5)
sonar	67.69	(5)	81.59	(2)	72.60	(4)	**84.47**	(1)	78.74	(3)
vehicle	44.68	(5)	**81.11**	(1)	65.35	(4)	75.14	(2)	72.21	(3)
liver-disorders	54.87	(5)	68.72	(3)	65.04	(4)	**72.99**	(1)	70.33	(2)
page-blocks	90.01	(5)	96.28	(3)	94.91	(4)	**97.54**	(1)	96.74	(2)
lymph	83.11	(3)	83.24	(2)	79.66	(4)	**83.92**	(1)	78.83	(5)
credit-g	75.16	(2)	71.58	(5)	73.58	(4)	**76.27**	(1)	73.65	(3)
cmc	50.48	(4)	51.53	(2)	50.19	(5)	50.63	(3)	**55.57**	(1)
haberman	75.36	(2)	70.29	(4)	73.14	(3)	68.17	(5)	**75.76**	(1)
dermatology	**97.40**	(1)	96.45	(4)	96.58	(3)	96.86	(2)	95.70	(5)
Average ranking		3.65		2.625		3.7		**2.075**		2.95

4.3.4 Predefined height and refinement rate

For evaluating the relevance of using the dataset information to define the size of individuals evolving in the DE-ADT$_{SPV}$ method, both the average heights of the constructed DTs and the number of refinements applied to the best DT in the last population of the evolutionary process are analyzed. Table 4.17 and Fig. 4.15 show the average heights produced by the DE-ADT$_{SPV}$ variants, and also the predefined height

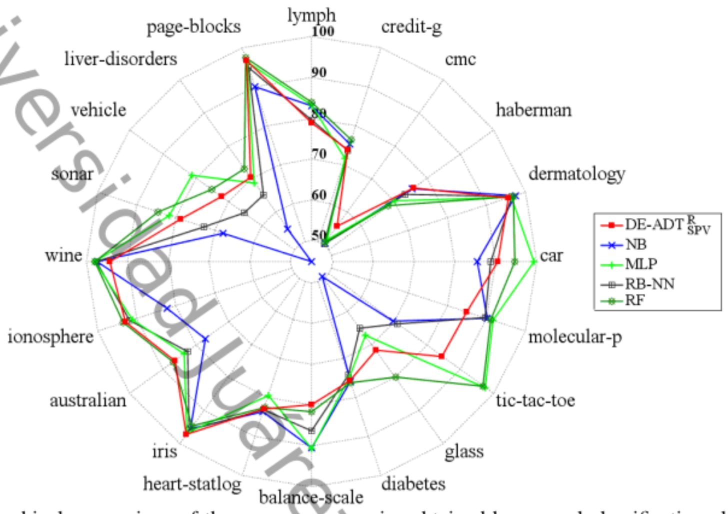


Figure 4.13: Graphical comparison of the average accuracies obtained by several classification algorithms and the DE-ADT variants.

Table 4.16: p-values for multiple comparisons among several classification algorithms and the DE-ADT variants.

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

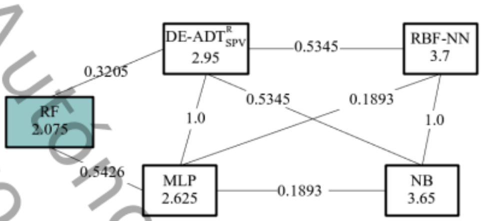


Figure 4.14: p-values graph of the classification algorithms and the DE-ADT variants.

computed before to apply the evolutionary process. In this Table is also shown the refinements percentages of the DTs constructed by the DE-ADT$^R_{SPV}$ variant, where the number of refinements represents the non-optimal leaf nodes of the best DT in the final population of the DE algorithm. These nodes are replaced with several sub-trees. In Table 4.17 is observed that the average heights of the DTs constructed are less than the predefined height in six datasets, and they surpass it in two or more levels in nine datasets. Two characteristics persist in the datasets with deep DTs: they have more than 600 training instances and more than two class labels. When the refinement percentage is analyzed, it is observed that this value is higher than 25% for these datasets.

4.4 Final remarks

In light of the experimental results, it can be affirmed that the DE-based approaches for DTI are efficient induction procedures since they construct DTs with high accuracy and a smaller number of leaf nodes. The refinement applied in the best DT in the final population permits to improve the training accuracy of the model. Notwithstanding the results yielded by these DE-based approaches are not better than those produced by the RF algorithm and the MLP-based classifier, they are statistically equivalent. An advantage

Table 4.17: Average DT heights of the DE-ADT variants.

Dataset	DT height			Leaf nodes		Refinement	
	Predefined	**Basic**	**Refined**	**Best DT**	**Refined DT**	**Number**	**%**
car	4	4.00	**6.00**	15.92	105.33	12.92	**81.16**
molecular-p	7	3.50	3.57	11.26	11.62	0.35	3.11
tic-tac-toe	5	5.00	**6.84**	25.30	79.80	15.48	**61.19**
glass	5	5.45	**7.73**	9.10	15.93	3.14	**34.51**
diabetes	5	5.25	**9.61**	6.44	18.55	2.11	**32.76**
balance-scale	4	5.00	**6.74**	10.11	18.43	7.24	**71.61**
heart-statlog	5	5.68	5.70	9.03	9.77	0.86	9.52
iris	4	3.99	4.01	4.35	4.35	0.00	0.00
australian	5	5.75	**7.49**	7.03	13.54	2.27	**32.29**
ionosphere	7	6.16	6.76	6.00	7.79	1.55	25.83
wine	5	4.02	4.09	5.18	5.25	0.11	2.12
sonar	7	5.83	6.15	7.87	10.40	2.41	30.62
vehicle	6	6.02	**20.04**	10.40	61.84	7.42	**71.35**
liver-disorders	4	4.79	**7.25**	7.68	14.29	2.28	**29.69**
page-blocks	5	5.25	**13.08**	7.83	40.45	6.85	**87.48**
lymph	6	4.89	4.89	12.91	13.14	0.96	7.44
credit-g	6	4.98	**7.24**	34.41	55.57	7.85	**22.81**
cmc	5	5.35	**6.50**	14.96	38.87	2.80	**18.72**
haberman	3	3.36	3.50	8.85	9.09	0.46	5.20
dermatology	7	6.00	7.15	18.46	24.07	3.69	19.99

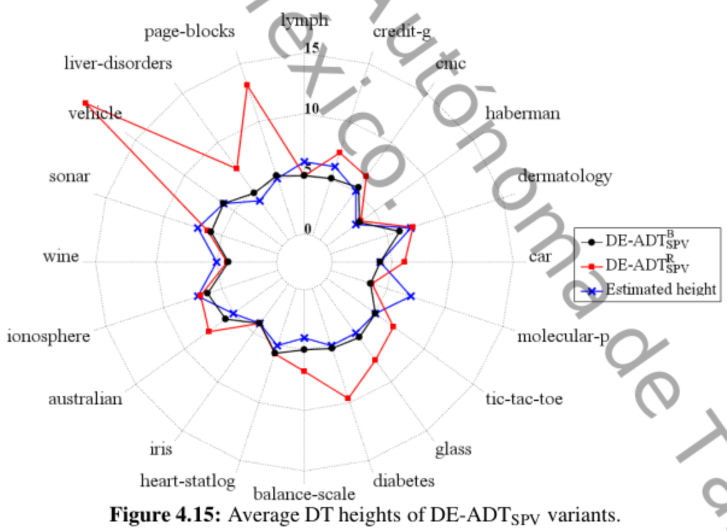


Figure 4.15: Average DT heights of DE-ADT$_{SPV}$ variants.

of the DE-ADT$^R_{SPV}$ variant is that it constructs models whose decisions and operations are easily understood, and although the RF method also builds DTs, its voting scheme makes it very difficult to trace the way in which the model takes its decisions. DE algorithm is an effective approach for constructing axis-parallel DTs when a rule to map a DT from a real-valued chromosome is implemented. An advantage of this approach is that the DE operators can be applied without any modification, and the chromosomes in population represent only feasible DTs.

In this thesis, an analysis of the run-time of the algorithms is not performed, since it is known that

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

Conclusions and future works

IN this thesis, three differential-evolution-based approaches to induce oblique and axis-parallel decision trees are described. This work was motivated by the fact that metaheuristics have shown to be a practical approach to induce more compact and precise decision trees than traditional techniques. Since metaheuristics perform an intelligent search in the tree space, they can better discover the relationships between the attributes describing a dataset, and they can also avoid the inherent problems of a constructive approach such as the traditional tree-induction procedure. The differential evolution algorithm was chosen in this work since it has proved to be an efficient algorithm to solve continuous optimization problems, and also because its use to induce decision trees is scarce.

In this thesis, the decision trees are represented as sequences of values, unlike other approaches such as genetic programming and coevolutionary algorithms, in which the individuals are encoded with tree-like structures. This linear encoding scheme allows applying the differential evolution algorithm without any modification, with the aim to exploit its exploration and exploitation skills. On the other hand, if a tree-like structure is used, several operators ensuring the generation of feasible trees must be implemented. Furthermore, it is known that some operators have a destructive effect on the induction process when they are used with this hierarchical representation.

Unlike other metaheuristic-based approaches in which the candidate solution are encoded with a fixed-size representation, the differential-evolution-based methods proposed in this thesis define the size of the individual according to the characteristics of the dataset whose model is built. This procedure is a contribution of this work since it can be used in any metaheuristic applied to induce decision trees, even for those such as genetic programming and coevolutionary algorithms.

This thesis defines a mapping scheme to obtain feasible trees of a sequence of values is introduced, associating each element of the sequence to the internal nodes of the tree and also uses the training set to add leaf nodes to it. In the case of building axis-parallel DT, this scheme applies the smallest-position-value rule to determine the order of evaluation of the attributes in the dataset, as well as implements three rules to avoid the inclusion of redundant nodes in the tree. This procedure is another contribution of the thesis which can be applied to any metaheuristic using individuals as sequences of values.

Furthermore, once the evolutionary process is ended, the best tree of the population is refined by substituting non-optimal leaf nodes by sub-trees, trying to increase the accuracy of the model. This refinement can also be considered a contribution to the present work.

Finally, the use of the algorithm of differential evolution to induce axis-parallel DTs introduced in this thesis is the first algorithm of this type, since there is no evidence in the existing literature of a similar approach.

As a result of the statistical tests carried out on the experimental results achieved in this thesis can be observed that the OC1-DE method does not be better induction algorithm to the other methods and that the EFTI method and the LEGAL-Tree$_{LA}$ algorithm are statistically equivalent to the DE-ODT method, the DE-ADT algorithm, respectively. Both DE-based methods implementing a global search do not be statistically equivalent to the J48 and the CART method, but they are statistically equivalent to the other classification methods. The following table shows the values of statistical significance are obtained:

Method	**OC1-DE**	**DE-ODT**	**DE-ADT$_{SPV}$**
Metaheuristics-based approaches for decision tree induction:			
OC1	1.6316e-01	**2.1828e-03**	-
OC1-SA	6.2576e-01	**1.5575e-02**	-
OC1-GA	1.0645e-01	**1.0333e-03**	-
OC1-ES	1.0000e+00	**3.5562e-02**	-
HBDT	6.8736e-01	**1.0179e-02**	-
GATree	8.6616e-02	**6.6756e-04**	-
EFTI	1.0000e+00	2.8452e-01	-
DE-OC1	-	1.0000e+00	-
DE-ODT	1.0000e+00	-	
LEGAL-Tree$_{LA}$	-	-	1.9745e-01
LEGAL-Tree$_{PA}$	-	-	**4.6482e-02**
GALE	-	-	**2.0418e-03**
Decision tree induction methods:			
J48	1.1134e-01	**2.5375e-02**	**1.0885e-04**
sCART	**2.5375e-02**	**1.4131e-03**	**1.2103e-03**
OC1-DE	-	5.9572e-01	
DE-ODT	5.9572e-01	-	
Other classification methods:			
NB	3.7787e-01	**2.8414e-02**	5.3445e-01
MLP	7.6079e-01	8.9374e-01	1.0000e+00
RBF-NN	7.7679e-01	1.3844e-01	5.3445e-01
RF	5.4336e-03	3.9736e-01	3.2047e-01
OC1-DE	-	7.6079e-01	-
DE-ODT	7.6079e-01	-	-

The differential-evolution-based approaches to build decision trees generate more precise and compact classifiers as compare to other decision tree induction methods with a statistical significance level not greater than 0.05.

After results of all experiments and statistical analysis, we conclude that our hypothesis: “The differential-evolution-based approaches to build decision trees generate more precise and compact classifiers as compare to other decision tree induction methods with a statistical significance level not greater than 0.05”, is accepted for the Differential-Evolution-based methods conducting a global search of a near-optimal decision trees. The OC1-DE algorithm does not reach the required statistical significance values since it is a recursive partitioning approach that finds good partitions in each internal node of the tree but does not have the ability of the global search methods.

Several lines of future work are derived from this thesis:

1. The effect of using other discretization strategies of the vector parameters determining the order of use of the attributes to build axis-parallel DTs must be analyzed. The effect of applying some technique for repairing the parameters of the vector must also be studied since in the current proposal those parameters generating redundant internal nodes are ignored. Finally, other metaheuristics can be implemented using the representation proposed in this thesis, and also they can compare the experimental results with the current developed versions.

2. The impact of using different versions of the differential evolution algorithms described in the state of the art literature to induce decision trees should be evaluated since it is expected to improve the

performance of the induced trees by applying schemes maintaining the diversity of the population, and applying an adjustment of the control parameters of the algorithm.

3. The effect of applying multi-objective metaheuristics considering the combination of accuracy and the size of the tree within the induction process should be studied.

4. The use of other performance measures such as sensitivity, specificity, and others to guide the search of a near-optimal decision tree should be analyzed.

5. To improve the classification accuracy, a procedure must be implemented that uses a classifiers ensemble constructed by the differential evolution algorithm.

6. Finally, the scalability of the algorithm should be studied when trying to generate models for a dataset with many attributes and/or instances.

References

[1] S. Abe. *Support Vector Machines for Pattern Classification*. Springer, 2005.

[2] J. Abonyi, J. A. Roubos, and 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):1–21, 2003.

[3] A. Agapitos, M. O'Neill, A. Brabazon, and T. Theodoridis. Maximum Margin Decision Surfaces for Increased Generalisation in Evolutionary Decision Tree Learning. In S. Silva et al., editors, *EuroGP 2011*, volume 6621 of *LNCS*, pages 61–72. Springer, 2011.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J. B. Bocca et al., editors, *VLDB 1994*, volume 1215, pages 487–499. Morgan Kaufmann, 1994.

[5] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.

[6] R. Ahmed and C. M. Rahman. Induction of Better Decision Trees Using Population Oriented Multi-Objective Simulated Annealing. In *ICCIT 2004*, pages 1–6. BRAC University, 2004.

[7] M. J. Aitkenhead. A co-evolving decision tree classification method. *Expert Systems with Applications*, 34(1):18–25, 2008.

[8] J. E. Alvarez-Benitez, R. M. Everson, and J. E. Fieldsend. A MOPSO Algorithm Based Exclusively on Pareto Dominance Concepts. In C. A. Coello et al., editors, *EMO 2005*, pages 459–473. Springer, 2005.

[9] J. András and D. Dumitrescu. Evolving orthogonal decision trees. *Studia Universitatis Babes-Bolyai. Series Informatica*, 48(2):33–44, 2003.

[10] S. Arlot and A. Celisse. A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79, 2010.

[11] Š. H. Babič, P. Kokol, V. Podgorelec, M. Zorman, M. Šprogar, and M. M. Štiglic. The Art of Building Decision Trees. *Journal of Medical Systems*, 24(1):43–52, 2000.

[12] Z. Bandar, H. Al-Attar, and D. McLean. Genetic Algorithm Based Multiple Decision Tree Induction. In *Proceedings of the 6th International Conference on Neural Information Processing (ICONIP'99)*, volume 2, pages 429–434, Perth, Australia, 1999. IEEE.

[13] R. C. Barros, M. P. Basgalupp, A. C. P. L. F. Carvalho, and A. A. Freitas. Towards the Automatic Design of Decision Tree Induction Algorithms. In N. Krasnogor, editor, *GECCO'11*, pages 567–574. ACM, 2011.

[14] R. C. Barros, M. P. Basgalupp, A. C. P. L. F. Carvalho, and A. A. Freitas. A Survey of Evolutionary Algorithms for Decision-Tree Induction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(3):291–312, 2012.

[15] R. C. Barros, M. P. Basgalupp, A. C. P. L. F. Carvalho, and A. A. Freitas. A hyper-heuristic evolutionary algorithm for automatically designing decision-tree algorithms. In T. Soule, editor, *GECCO'12*, pages 1237–1244. ACM, 2012.

[16] R. C. Barros, M. P. Basgalupp, A. C. P. L. F. Carvalho, and A. A. Freitas. Automatic design of decision-tree algorithms with evolutionary algorithms. *Evolutionary Computation*, 21(4):659–684, 2013.

[17] R. C. Barros, A. C. P. L. F. Carvalho, and A. A. Freitas. *Automatic Design of Decision-Tree Induction Algorithms*, chapter Decision-Tree Induction, pages 7–45. Springer, 2015.

[18] M. P. Basgalupp, R. C. Barros, and T. Barabasz. A Grammatical Evolution Based Hyper-heuristic for the Automatic Design of Split Criteria. In Ch. Igel, editor, *GECCO'14*, pages 1311–1318. ACM, 2014.

[19] M. P. Basgalupp, R. C. Barros, A. C. P. L. F. de Carvalho, and A. A. Freitas. Evolving decision trees with beam search-based initialization and lexicographic multi-objective evaluation. *Information Sciences*, 258:160–181, February 2014.

[20] M. P. Basgalupp, R. C. Barros, and V. Podgorelec. Evolving Decision-tree Induction Algorithms with a Multi-objective Hyper-heuristic. In *SAC'15*, pages 110–117. ACM, 2015.

[21] M. P. Basgalupp, A. C. P. L. F. Carvalho, R. C. Barros, D. D. Ruiz, and A. A. Freitas. Lexicographic multi-objective evolutionary induction of decision trees. *Int. Journal of Bio-Inspired Computation*, 1(1–2):105–117, 2009.

[22] I. A. Basheer and M. Hajmeer. Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods*, 43(1):3 – 31, 2000.

[23] W. A. Belson. Matching and prediction on the principle of biological classification. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, pages 65–75, 1959.

[24] K. P. Bennett. Decision tree construction via linear programming. Technical report, Center for Parallel Optimization, Computer Sciences Department, University of Wisconsin, 1992.

[25] K. P. Bennett. Global tree optimization: A non-greedy decision tree algorithm. *Computing Science and Statistics*, pages 156–156, 1994.

[26] K. P. Bennett and J. A. Blue. An extreme point tabu search method for data mining. Technical report, Rensselaer Polytechnic Institute, Troy, NY, 1996.

[27] N. Beume, B. Naujoks, and M. Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653 – 1669, 2007.

[28] J. C. Bezdek, R. Ehrlich, and W. Full. FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2):191–203, 1984.

[29] R. Biedrzycki and J. Arabas. Evolutionary and greedy exploration of the space of decision trees. *Evolutionary Computation and Global Optimization*, pages 479–489, 2006.

[30] M. Birattari. *Tuning metaheuristics: a machine learning perspective*, volume 197 of *Studies in Computational Intelligence*. Springer, 2009.

[31] J. R. Bitner, G. Ehrlich, and E. M. Reingold. Efficient Generation of the Binary Reflected Gray Code and Its Applications. *Communications of the ACM*, 19(9):517–521, September 1976.

[32] P. E. Blower and K. P. Cross. Decision tree methods in pharmaceutical research. *Current Topics in Medicinal Chemistry*, 6(1):31–39, 2006.

[33] C. Blum. Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, 2(4):353–373, 2005.

[34] L. Bobrowski and M. Krętowski. Induction of multivariate decision trees by using dipolar criteria. In D. A. Zighed et al., editors, *PKDD 2000*, volume 1910 of *LNAI*, pages 331–336. Springer, 2000.

[35] P. P. Bonissone. Soft computing: the convergence of emerging reasoning technologies. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 1(1):6–18, 1997.

[36] U. Boryczka and J. Kozak. Ant Colony Decision Trees - A New Method for Constructing Decision Trees Based on Ant Colony Optimization. In J. S. Pan et al., editors, *ICCCI 2010, Part I*, volume 6421 of *LNAI*, pages 373–382. Springer, 2010.

[37] U. Boryczka and J. Kozak. An adaptive discretization in the ACDT algorithm for continuous attributes. In *Computational Collective Intelligence. Technologies and Applications*, pages 475–484. Springer, 2011.

[38] U. Boryczka and J. Kozak. New insights of cooperation among ants in Ant Colony Decision Trees. In A. Abraham et al., editors, *NaBIC 2011*, pages 255–260. IEEE, 2011.

[39] U. Boryczka and J. Kozak. Enhancing the effectiveness of ant colony decision tree algorithms by co-learning. *Applied Soft Computing*, 30:166..178, 2015.

[40] L. Bosnjak, S. Karakatic, and V. Podgorelec. Using similarity-based selection in evolutionary design of decision trees. In *MIPRO 2015*, pages 1206–1211. IEEE, 2015.

[41] M. C. J. Bot and W. B. Langdon. Application of Genetic Programming to Induction of Linear Classification Trees. In R. Poli et al., editors, *EuroGP 2000*, volume 1802 of *LNCS*, pages 247–258. Springer, 2000.

[42] M. C. J. Bot and W. B. Langdon. Improving Induction of Linear Classification Trees with Genetic Programming. In L. D. Whitley et al., editors, *GECCO-2000*, pages 403–410. Morgan Kaufmann, 2000.

[43] B. Bouchon-Meunier, R. R. Yager, and L. A. Zadeh. *Fuzzy Logic and Soft Computing*. World Scientific, 1995.

[44] C. V. Bratu, C. Savin, and R. Potolea. A hybrid algorithm for medical diagnosis. In *EUROCON-2007*, pages 668–673. IEEE, 2007.

[45] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[46] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Chapman and Hall, 1984.

[47] L.A. Breslow and D.W. Aha. Simplifying decision trees: A survey. *The Knowledge Engineering Review*, 12(01):1–40, 1997.

[48] K. Bringmann, T. Friedrich, F. Neumann, and M. Wagner. Approximation-guided Evolutionary Multi-objective Optimization. In T. Walsh, editor, *IJCAI'11*, pages 1198–1203. AAAI, 2011.

[49] C.E. Brodley and P.E. Utgoff. Multivariate decision trees. *Machine Learning*, 19(1):45–77, 1995.

[50] D. S. Broomhead and D. Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, HMSO, 1988.

[51] R. S. Bucy and R. S. Diesposti. Classification Tree Optimization by Simulated Annealing. Summary report, The Aerospace Corporation, 1991.

[52] R. S. Bucy and R. S. Diesposti. Decision tree design by simulated annealing. *ESAIM: Mathematical Modelling and Numerical Analysis*, 27(5):515–534, 1993.

[53] F. V. Buontempo, X. Z. Wang, M. Mwense, N. Horan, A. Young, and D. Osborn. Genetic programming for the induction of decision trees to model ecotoxicity data. *Journal of Chemical Information and Modeling*, 45(4):904–912, 2005.

[54] M. Bursa and L. Lhotska. Automated Classification Tree Evolution Through Hybrid Metaheuristics. In E. Corchado et al., editors, *Innovations in Hybrid Intelligent Systems*, volume 44 of *ASC*, pages 191–198. Springer, 2007.

[55] M. Bursa and L. Lhotska. Ant-inspired algorithms for decision tree induction. In E. M. Renda et al., editors, *ITBAM 2015*, volume 9267 of *LNCS*, pages 95–106. Springer, 2015.

[56] H. Bustince, M. Pagola, E. Barrenechea, J. Fernandez, P. Melo-Pinto, P. Couto, H. R. Tizhoosh, and J. Montero. Ignorance functions. An application to the calculation of the threshold in prostate ultrasound images. *Fuzzy Sets and Systems*, 161(1):20–36, 2010.

[57] B. Calvo and R. Guzmán-Santafé. scmamp: Statistical Comparison of Multiple Algorithms in Multiple Problems. *The R Journal*, 8(1):248–256, 2016.

[58] E. Cantú-Paz and Ch. Kamath. Using Evolutionary Algorithms to Induce Oblique Decision Trees. In D. Whitley et al., editors, *GECCO 2000*, pages 1053–1060. Morgan Kaufmann, 2000.

[59] E. Cantú-Paz and Ch. Kamath. Inducing oblique decision trees with evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 7(1):54–68, 2003.

[60] S. H. Cha and C. Tappert. Constructing Binary Decision Trees using Genetic Algorithms. In H. R. Arabnia and Y. Mun, editors, *Proceedings of the 2008 International Conference on Genetic and Evolutionary Methods (GEM 2008)*, pages 49–54, Las Vegas, Nevada, USA, 2008. CSREA.

[61] S. H. Cha and C. Tappert. A genetic algorithm for constructing compact binary decision trees. *Journal of Pattern Recognition Research*, 4(1):1–13, 2009.

[62] B. B. Chai, T. Huang, X. Zhuang, Y. Zhao, and J. Sklansky. Piecewise linear classifiers using binary tree structure and genetic algorithm. *Pattern Recognition*, 29(11):1905–1917, 1996.

[63] B. B. Chai, X. Zhuang, Y. Zhao, and J. Sklansky. Binary linear decision tree with genetic algorithm. In *Proceedings of the 13th International Conference on Pattern Recognition (ICPR'96)*, volume IV, Track D: Parallel and Connectionist Systems, pages 530–534, Vienna, Austria, 1996. IEEE.

[64] C. L. Chan, C. Y. Lee, N. P. Yang, and S. Y. Shen. Classification Method Incorporating Decision Tree with Particle Swarm Optimization. In J. Watada et al., editors, *ICGEC 2011*, pages 216–219. IEEE, 2011.

[65] J. Y. Chang, Ch. W. Cho, S. H. Hsieh, and Sh. T. Chen. Genetic algorithm based Fuzzy ID3 algorithm. In N. R. Pal et al., editors, *ICONIP 2004*, volume 3316 of *LNCS*, pages 989–995. Springer, 2004.

[66] J. Chen, X. Wang, and J. Zhai. Pruning decision tree using genetic algorithms. In *AICI'09*, volume 3, pages 244–248. IEEE, 2009.

[67] Y. J. Cho, H. S. Lee, and C. H. Jun. Optimization of Decision Tree for Classification Using a Particle Swarm. *Industrial Engineering and Management Systems*, 10(4):272–278, 2011.

[68] P. Cichosz. Assessing the quality of classification models: Performance measures and evaluation procedures. *Open Engineering*, 1(2):132–158, 2011.

[69] K. J. Cios and L. M. Sztandera. Continuous ID3 algorithm with fuzzy entropy measures. In *FUZZ IEEE*, pages 469–476. IEEE, 1992.

[70] P. Civicioglu and E. Besdok. A conceptual comparison of the cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms. *Artificial Intelligence Review*, 39(4):315–346, 2013.

[71] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

[72] C. A. Coello-Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Springer, 2006.

[73] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. on Information Theory*, 13(1):21–27, 1967.

[74] D. R. Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 215–242, 1958.

[75] M. Craven and J. Shavlik. Rule extraction: Where do we go from here? *University of Wisconsin, Machine Learning Research Group, working Paper*, 99, 1999.

[76] K. A. Crockett, Z. Bandar, and A. Al-Attar. Optimising Decision Classifications Using Genetic Algorithms. In A. Dobnikar et al., editors, *Artificial Neural Nets and Genetic Algorithms*, pages 191–195. Springer, 1999.

[77] G.B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1998.

[78] S. Das, A. Abraham, and A. Konar. Automatic clustering using an improved differential evolution algorithm. *IEEE Trans. on Systems, Man, and Cybernetics–Part A: Systems and Humans*, 38(1):218–237, jan 2008.

[79] S. Das, A. Konar, and U.K. Chakraborty. Two Improved Differential Evolution Schemes for Faster Global Search. In H.G. Beyer, editor, *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation (GECCO'05)*, pages 991–998, Washington, DC, USA, 2005. ACM.

[80] S. Das and P. N. Suganthan. Differential Evolution: A Survey of the State-of-the-Art. *IEEE Transactions on Evolutionary Computation*, 15(1):4–31, 2011.

[81] K. Deb and D. Kalyanmoy. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[82] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Trans. on Evolutionary Computation*, 6(2):182–197, 2002.

[83] I. De Falco. Differential evolution for automatic rule extraction from medical databases. *Applied Soft Computing*, 13(2):1265–1283, 2013.

[84] R. K. DeLisle and S. L. Dixon. Induction of decision trees via evolutionary programming. *Journal of Chemical Information and Computer Sciences*, 44(3):862–870, 2004.

[85] J. Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *The Journal of Machine Learning Research*, 7(Dec):1–30, 2006.

[86] J. Derrac, S. García, D. Molina, and F. Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3 – 18, 2011.

[87] T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923, 1998.

[88] P. Domingos. Metacost: A general method for making classifiers cost-sensitive. In *KDD'99*, pages 155–164. ACM, 1999.

[89] M. Dorigo. *Optimization, learning and natural algorithms*. PhD thesis, Politecnico di Milano, 1992.

[90] K.L. Du and M.N.S. Swamy. *Search and Optimization by Metaheuristics*. Springer, 2016.

[91] E. Dufourq and N. Pillay. Incorporating adaptive discretization into genetic programming for data classification. In *WICT-2013*, pages 127–133. IEEE, 2013.

[92] E. Dufourq and N. Pillay. A preliminary study on the reuse of subtrees within decision trees in a genetic programming context for data classification. In *WICT-2013*, pages 285–290. IEEE, 2013.

[93] D. Dumitrescu and J. András. Generalized decision trees built with evolutionary techniques. *Studies in Informatics and Control*, 14(1):15–22, 2005.

[94] O.J. Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64, 1961.

[95] J.J. Durillo and A.J. Nebro. jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760–771, 2011.

[96] J. Dvořák and P. Savický. Softening Splits in Decision Trees using Simulated Annealing. In B. Beliczynski et al., editors, *ICANNGA 2007, Part I*, volume 4431 of *LNCS*, pages 721–729. Springer, 2007.

[97] R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *MHS'95*, volume 1, pages 39–43. IEEE, 1995.

[98] J. Eggermont. Evolving Fuzzy Decision Trees with Genetic Programming and Clustering. In J. A. Foster et al., editors, *EuroGP 2002*, volume 2278 of *LNCS*, pages 71–82. Springer, 2002.

[99] J. Eggermont, J. N. Kok, and W. A. Kosters. Genetic programming for data classification: Refining the search space. In T. Heskes et al., editors, *BNAIC'03*, pages 123–130. University of Nijmegen, 2003.

[100] J. Eggermont, J. N. Kok, and W. A. Kosters. Genetic programming for data classification: Partitioning the search space. In *SAC'04*, pages 1001–1005. ACM, 2004.

[101] A. P. Engelbrecht, S. Rouwhorst, and L. Schoeman. A building block approach to genetic programming for rule discovery. *Data Mining: A Heuristic Approach*, pages 174–189, 2001.

[102] H. A. Eschenauer, J. Koski, and A. Osyczka. Multicriteria optimization–fundamentals and motivation. In *Multicriteria Design Optimization*. Springer, 1990.

[103] P. G. Espejo, S. Ventura, and F. Herrera. A survey on the application of genetic programming to classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(2):121–144, 2010.

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

[105] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[106] T. Fawcett. PRIE: a system for generating rulelists to maximize ROC performance. *Data Mining and Knowledge Discovery*, 17(2):207–224, 2008.

[107] U. M. Fayyad and K. B. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8(1):87–102, 1992.

[108] T. A. Feo and M. G. C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2):67–71, 1989.

[109] V. Feoktistov. *Differential Evolution: In Search of Solutions*. Springer Optimization and Its Applications. Springer, 2007.

[110] C. Ferreira. Gene Expression Programming: a New Adaptive Algorithm for Solving Problems. *Complex Systems*, 13(2):87–129, 2001.

[111] C. Ferreira. *Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence*, chapter Decision Tree Induction, pages 337–380. Springer, 2006.

[112] C. Ferri, P. A. Flach, and J. Hernández-Orallo. Learning Decision Trees Using the Area Under the ROC Curve. In C. Sammut et al., editors, *ICML 2002*, pages 139–146. Morgan Kaufmann, 2002.

[113] C. Ferri, J. Hernández-Orallo, and R. Modroiu. An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30(1):27–38, 2009.

[114] J. E. Fieldsend. Optimizing decision trees using multi-objective particle swarm optimization. In C. A. Coello-Coello, S. Dehuri, and S. Ghosh, editors, *Swarm Intelligence for Multi-objective Problems in Data Mining*, volume 242 of *SCI*, pages 93–114. Springer, 2009.

[115] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[116] R.A. Fisher. *Statistical methods and scientific inference*. Hafner Publishing Co., 1956.

[117] G. Folino, C. Pizzuti, and G. Spezzano. A cellular genetic programming approach to classification. In W. Banzhaf et al., editors, *GECCO 1999*, pages 1015–1020. Morgan Kaufmann, 1999.

[118] G. Folino, C. Pizzuti, and G. Spezzano. Genetic Programming and Simulated Annealing: A Hybrid Method to Evolve Decision Trees. In R. Poli et al., editors, *EuroGP 2000*, volume 1802 of *LNCS*, pages 294–303. Springer, 2000.

[119] C. M. Fonseca and P. J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In S. Forrester, editor, *ICGA-93*, pages 416–423. Morgan Kaufmann, 1993.

[120] E. Frank. Fully supervised training of Gaussian radial basis function networks in WEKA. Technical Report 04, Department of Computer Science, The University of Waikato, 2014.

[121] E. Frank and I. H. Witten. Generating accurate rule sets without global optimization. In J. W. Shavlik, editor, *ICML'98*, pages 144–151. Morgan Kaufmann, 1998.

[122] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1–2):95–110, 1956.

[123] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus. Knowledge discovery in databases: An overview. *AI magazine*, 13(3):57, 1992.

[124] A. R. R. Freitas, R. C. P. Silva, and F. G. Guimarães. Differential Evolution and Perceptron Decision Trees for Fault Detection in Power Transformers. In V. Snášel et al., editors, *SOCO Models in Industrial & Environmental Appl.*, volume 188 of *AISC*, pages 143–152. Springer, 2013.

[125] A.A. Freitas. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer, 2002.

[126] Y. Freund and L. Mason. The alternating decision tree learning algorithm. In L. Saitta, editor, *ICML'96*, pages 124–133. Morgan Kaufmann, 1999.

[127] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In P. Vitányi, editor, *EuroCOLT'95*, volume 904 of *LNCS*, pages 23–37. Springer, 1995.

[128] M. Friedman. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[129] Z. Fu. An innovative GA-based decision tree classifier in large scale data mining. In J. M. Zytkow et al., editors, *PKDD'99*, volume 1704 of *1704*, pages 348–353. Springer, 1999.

[130] Z. Fu, B. L. Golden, S. Lele, S. Raghavan, and E. A. Wasil. Building a High-quality Decision Tree with a Genetic Algorithm. In M. Laguna et al., editors, *Computing Tools for Modeling, Optimization and Simulation*, volume 12 of *OR/CS Interfaces*, pages 25–38. Springer, 2000.

[131] Z. Fu, B. L. Golden, S. Lele, S. Raghavan, and E. A. Wasil. A genetic algorithm-based approach for building accurate decision trees. *INFORMS Journal on Computing*, 15(1):3–22, 2003.

[132] Z. Fu, B. L. Golden, S. Lele, S. Raghavan, and E. A. Wasil. Genetically engineered decision trees: population diversity produces smarter trees. *Operations Research*, 51(6):894–907, 2003.

[133] Z. Fu, B. L. Golden, S. Lele, S. Raghavan, and E. A. Wasil. Diversification for better classification trees. *Computers & Operations Research*, 33(11):3185–3202, 2006.

[134] Z. Fu and F. Mae. A computational study of using genetic algorithms to develop intelligent decision trees. In *CEC 2001*, volume 2, pages 1382–1387. IEEE, 2001.

[135] M. Galea, Q. Shen, and J. Levine. Evolutionary Approaches to Fuzzy Modelling for Classification. *The Knowledge Engineering Review*, 19(1):27–59, March 2004.

[136] J. Gama. Oblique Linear Tree. In X. Liu et al., editors, *IDA-97*, volume 1280 of *LNCS*, pages 187–198. Springer, 1997.

[137] J. Gama and P. Brazdil. Linear tree. *Intelligent Data Analysis*, 3(1):1–22, 1999.

[138] S. García, J. Derrac, I. Triguero, C. J. Carmona, and F. Herrera. Evolutionary-based selection of generalized instances for imbalanced classification. *Knowledge-Based Systems*, 25(1):3–12, 2012.

[139] M. R. Garey. *Optimal Binary Decision Trees for Diagnostic Identification Problems*. PhD thesis, The University of Wisconsin, 1970.

[140] C. Gathercole. *An investigation of supervised learning in genetic programming*. PhD thesis, University of Edinburgh, 1998.

[141] K. Geetha and S. S. Baboo. An Empirical Model for Thyroid Disease Classification using Evolutionary Multivariate Bayesian Prediction Method. *Global Journal of Computer Science and Technology*, 16(1):1–9, 2016.

[142] F. Glover. Tabu Search - Part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[143] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1st edition, 1989.

[144] J. B. Gray and G. Fan. TARGET: tree analysis with randomly generated and evolved trees. Technical report, University of Alabama, 2003.

[145] J. B. Gray and G. Fan. Classification tree analysis using TARGET. *Computational Statistics & Data Analysis*, 52(3):1362–1372, 2008.

[146] J. J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1):122–128, 1986.

[147] T. Grubinger, A. Zeileis, and K. P. Pfeiffer. evtree: Evolutionary Learning of Globally Optimal Classification and Regression Trees in R. *Journal of Statistical Software*, 61(1):1–29, 2014.

[148] G. K. Gupta. *Introduction to Data Mining with case studies*. PHI Learning, 2014.

[149] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[150] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Elsevier, 2006.

[151] J.A. Hanley and B.J. McNeil. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology*, 148(3):839–843, 1983.

[152] S. Haruyama and Q. Zhao. Designing smaller decision trees using multiple objective optimization based GPs. In A. El-Kamel et al., editors, *Proc. of the Int. Conf. on Systems, Man and Cybernetics*, volume 6, pages 5–pp. IEEE, 2002.

[153] D. G. Heath. *A Geometric Framework for Machine Learning*. PhD thesis, Johns Hopkins University, 1993.

[154] D. G. Heath, S. Kasif, and S. Salzberg. Induction of Oblique Decision Trees. In R. Bajcsy, editor, *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 1002–1007, Chambéry, France, 1993.

[155] A. B. Hens and M. K. Tiwari. Computational time reduction for credit scoring: An integrated approach based on support vector machine and stratified sampling method. *Expert Systems with Applications*, 39(8):6774–6781, 2012.

[156] W. D. Hillis. Co-evolving Parasites Improve Simulated Evolution As an Optimization Procedure. *Physica D: Nonlinear Phenomena*, 42(1–3):228–234, 1990.

[157] T.K. Ho. The random subspace method for constructing decision forests. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[158] J. H. Holland. *Adaptation in natural and artificial systems*. U. Michigan Press, 1975.

[159] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, pages 65–70, 1979.

[160] R. C. Holte. Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. *Machine Learning*, 11(1):63–90, 1993.

[161] G. Hommel. A stagewise rejective multiple test procedure based on a modified Bonferroni test. *Biometrika*, 75(2):383–386, 1988.

[162] H. H. Hoos. *Stochastic Local Search - Methods, Models, Applications*. PhD thesis, Darmstadt University of Technology, 1998.

[163] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[164] J. Horn, N. Nafpliotis, and D. E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Proc. of the 1th IEEE Conf. in Evolutionary Computation*, pages 82–87. IEEE, 1994.

[165] T. Hothorn, K. Hornik, and A. Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674, 2006.

[166] J. Hu, G. S. Xia, F. Hu, and L. Zhang. A Comparative Study of Sampling Analysis in the Scene Classification of Optical High-Spatial Resolution Remote Sensing Imagery. *Remote Sensing*, 7(11):14988–15013, 2015.

[167] M. Hu, Y. Liao, W. Wang, G. Li, B. Cheng, and F. Chen. Decision Tree-Based Maneuver Prediction for Driver Rear-End Risk-Avoidance Behaviors in Cut-In Scenarios. *Journal of Advanced Transportation*, 2017(ID 7170358):1–12, 2017.

[168] E. B. Hunt, J. Marin, and P. J. Stone. *Experiments in induction*. Academic Press, 1966.

[169] J. Huysmans, K. Dejaeger, C. Mues, J. Vanthienen, and B. Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51(1):141–154, 2011.

[170] L. Hyafil and R. L. Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1):15–17, 1976.

[171] H. Iba, H. de Garis, and T. Sato. Genetic programming using a minimum description length principle. In K. E. Kinnear, editor, *Advances in Genetic Programming*, pages 265–284. MIT Press, 1994.

[172] A. Ittner and M. Schlosser. Non-linear decision trees-NDT. In L. Saitta, editor, *ICML'96*, pages 252–257. Morgan Kaufmann, 1996.

[173] C. Z. Janikow. A genetic algorithm method for optimizing fuzzy decision trees. *Information Sciences*, 89(3):275–296, 1996.

[174] D. Jankowski and K. Jackowski. Evolutionary Algorithm for Decision Tree Induction. In K. Saeed and V. Snášel, editors, *Proceedings of the 13th IFIP TC8 International Conference on Computer Information Systems and Industrial Management (CISIM 2014)*, volume 8838 of *LNCS*, pages 23–32. Springer, Ho Chi Minh City, Vietnam, 2014.

[175] R. Jiang. Gene-gene interaction. In M. D. Gellman, editor, *Encyclopedia of Behavioral Medicine*, pages 841–842. Springer, 2013.

[176] U. Johansson, R. König, T. Löfström, and L. Niklasson. Using Imaginary Ensembles to Select GP Classifiers. In A. I. Esparcia-Alcázar et al., editors, *EuroGP 2010*, pages 278–288. Springer, 2010.

[177] U. Johansson and L. Niklasson. Evolving decision trees using oracle guides. In *CIDM'09*, pages 238–244. IEEE, 2009.

[178] U. Johansson, C. Sönströd, and T. Löfström. One tree to explain them all. In *CEC-2011*, pages 1444–1451. IEEE, 2011.

[179] G.H. John and P. Langley. Estimating Continuous Distributions in Bayesian Classifiers. In P. Besnard and S. Hanks, editors, *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI'95)*, pages 338–345, San Francisco, CA, USA, 1995. Morgan Kaufmann.

[180] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning. *Journal Operations Research*, 39(3):378–406, May 1991.

[181] M. Jovanović, B. Delibašić, M. Vukićević, M. Suknović, and M. Martić. Evolutionary Approach for Automated Component-based Decision Tree Algorithm Design. *Intelligent Data Analysis*, 18(1):63–77, January 2014.

[182] D. Kalles and A. Papagelis. Lossless fitness inheritance in genetic algorithms for decision trees. *Soft Computing*, 14(9):973–993, 2010.

[183] 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):119–127, 1980.

[184] V. Kecman. *Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models*. MIT Press, Cambridge, MA, USA, 2001.

[185] H. Kennedy, C. Chinniah, P. V. G. Bradbeer, and L. Morss. The Construction and Evaluation of Decision Trees: A Comparison of Evolutionary and Concept Learning Methods. In D. Corne and J. L. Shapiro, editors, *AISB International Workshop on Evolutionary Computation (Selected papers)*, volume 1305 of *LNCS*, pages 147–162, Manchester, UK, 1997. Springer.

[186] T. M. Khoshgoftaar and Y. Liu. A multi-objective software quality classification model using genetic programming. *IEEE Trans. on Reliability*, 56(2):237–245, 2007.

[187] T. M. Khoshgoftaar, N. Seliya, and Y. Liu. Genetic programming-based decision trees for software quality classification. In *ICTAI 2003*, pages 374–383. IEEE, 2003.

[188] D. E. Kim. Structural risk minimization on decision trees using an evolutionary multiobjective optimization. In M. Keijzer et al., editors, *EuroGP 2004*, volume 3003 of *LNCS*, pages 338–348. Springer, 2004.

[189] D. E. Kim. Minimizing structural risk on decision tree classification. In J. Yaouchu, editor, *Multi-Objective Machine Learning*, volume 16 of *SCI*, pages 241–260. Springer, 2006.

[190] H. Kim and W. Y. Loh. Classification trees with unbiased multiway splits. *Journal of the American Statistical Association*, 96(454), 2001.

[191] M. W. Kim and J. W. Ryu. Optimized Fuzzy Classification using Genetic Algorithm. In L. Wang et al., editors, *FSKD 2005*, volume 3613 of *LNAI*, pages 392–401. Springer, 2005.

[192] M. W. Kim and J. W. Ryu. Optimized fuzzy decision tree using genetic algorithm. In I. King et al., editors, *ICONIP 2006, Part III*, volume 4234 of *LNCS*, pages 797–806. Springer, 2006.

[193] S. Kirkpatrick, D. C. Gelatt, and M. P. Vecchi. Optimization by simmulated annealing. *Science*, 220(4598):671–680, 1983.

[194] R. Kohavi and F. Provost. Glossary of terms. *Machine Learning*, 30(2–3):271–274, 1998.

[195] P. Kokol, S. Pohorec, G. Štiglic, and V. Podgorelec. Evolutionary design of decision trees for medical application. *Data Mining and Knowledge Discovery*, 2(3):237–254, 2012.

[196] E. Kolçe and N. Frasheri. The use of heuristics in decision tree learning optimization. *International Journal of Computer Engineering in Research Trends*, 1(3):127–130, 2014.

[197] R. König, U. Johansson, T. Löfström, and L. Niklasson. Improving GP classification performance by injection of decision trees. In *CEC 2010*, pages 1–8. IEEE, 2010.

[198] S. B. Kotsiantis. Decision trees: a recent overview. *Artificial Intelligence Review*, 39(4):261–283, 2013.

[199] J. R. Koza. Hierarchical Genetic Algorithms Operating on Populations of Computer Programs. In N. S. Sridharan, editor, *IJCAI'89*, pages 768–774. Morgan Kauffman, 1989.

[200] J. R. Koza. Concept formation and decision tree induction using the genetic programming paradigm. In Schwefel H. P. et al., editors, *PPSN I*, volume 496 of *LNCS*, pages 124–128. Springer, 1991.

[201] J. R. Koza. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer, 2003.

[202] J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[203] M. Krętowski. An evolutionary algorithm for oblique decision tree induction. In L. Rutkowski, J. Siekmann, R. Tadeusiewicz, and L. A. Zadeh, editors, *Proceedings of the 7th International Conference on Artificial Intelligence and Soft Computing (ICAISC 2004)*, volume 3070 of *LNAI*, pages 432–437. Springer, Zakopane, Poland, 2004.

[204] M. Krętowski. A memetic algorithm for global induction of decision trees. In V. Geffert et al., editors, *SOFSEM 2008*, volume 4910 of *LNCS*, pages 531–540. Springer, 2008.

[205] M. Krętowski and M. Grześ. Global Induction of Oblique Decision Trees: An Evolutionary Approach. In M. A. Kłopotek et al., editors, *IIPWM'05*, volume 31 of *ASC*, pages 309–318. Springer, 2005.

[206] M. Krętowski and M. Grześ. Global learning of decision trees by an evolutionary algorithm. In K. Saeed et al., editors, *Information Processing and Security Systems*, pages 401–410. Springer, 2005.

[207] M. Krętowski and M. Grześ. Evolutionary Learning of Linear Trees with Embedded Feature Selection. In L. Rutkowski et al., editors, *ICAISC 2006*, volume 4029 of *LNAI*, pages 400–409. Springer, 2006.

[208] M. Krętowski and M. Grześ. Evolutionary Induction of Decision Trees for Misclassification Cost Minimization. In B. Beliczynski et al., editors, *ICANNGA 2007, Part I*, volume 4431 of *LNCS*, pages 1–10. Springer, 2007.

[209] W.H. Kruskal and W.A. Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.

[210] C. S. Kuo, T. P. Hong, and C. L. Chen. Learning Classification Trees by Genetic Programming. In *The 2006 Int. Conf. on Hybrid Information Technology*, 2006.

[211] C. S. Kuo, T. P. Hong, and C. L. Chen. A Knowledge-Acquisition Strategy Based on Genetic Programming. In Y. J. Na et al., editors, *ICCIT 2007*, pages 217–221. IEEE, 2007.

[212] C. S. Kuo, T. P. Hong, and C. L. Chen. Applying genetic programming technique in classification trees. *Soft Computing*, 11(12):1165–1172, 2007.

[213] C. S. Kuo, T. P. Hong, and C. L. Chen. A knowledge-evolution strategy based on genetic programming. In G. Lee et al., editors, *ICHIT'08*, pages 43–48. IEEE, 2008.

[214] T.N. Lal, O. Chapelle, J. Weston, and A. Elisseeff. Embedded methods. In I. Guyon, M. Nikravesh, S. Gunn, and L.A. Zadeh, editors, *Feature Extraction: Foundations and Applications*, volume 207 of *Studies in Fuzziness and Soft Computing*, pages 137–165. Springer, 2006.

[215] A. H. Land and A. G. Doig. An Automatic Method of Solving Discrete Programming Problems. *Econometrica*, 28(3):497–520, 1960.

[216] N. Landwehr, M. Hall, and E. Frank. Logistic model trees. *Machine Learning*, 95(1–2):161–205, 2005.

[217] V. E. Lee, L. Liu, and R. Jin. Decision Trees: Theory and Algorithms. In *Data Classification: Algorithms and Applications*, pages 87–120. CRC Press, 2014.

[218] N. Leema, H.K. Nehemiah, and A. Kannan. Neural network classifier optimization using Differential Evolution with Global Information and Back Propagation algorithm for clinical datasets. *Applied Soft Computing*, 49:834–844, December 2016.

[219] D. Levi. Hereboy: A fast evolutionary algorithm. In J. Lohn et al., editors, *EH'00*, pages 17–24. IEEE, 2000.

[220] J. Li. *FGP: A Genetic Programming Based Financial Forecasting Tool*. PhD thesis, University of Essex, 2000.

[221] J. Li, L. Ding, and B. Li. Differential evolution-based parameters optimisation and feature selection for support vector machine. *International Journal of Computational Science and Engineering*, 13(4):355–363, 2016.

[222] J. Li, X. Li, and X. Yao. Cost-sensitive classification with genetic programming. In *CEC 2005*, volume 3, pages 2114–2121. IEEE, 2005.

[223] X. Li and M. Yin. An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure. *Advances in Engineering Software*, 55:10–31, 2013.

[224] X. B. Li, J. R. Sweigart, J. T. C. Teng, J. M. Donohue, L. Thombs, and S. M. Wang. Multivariate decision trees using linear discriminants and tabu search. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 33(2):194–205, 2003.

[225] M. Lichman. UCI Machine Learning Repository, 2013. University of California, Irvine, School of Information and Computer Sciences.

[226] R. P. Lippmann. An introduction to computing with neural nets. *ASSP Magazine*, 4(2):4–22, 1987.

[227] J. J. Liu and J. T. Y. Kwok. An extended genetic rule induction algorithm. In *CEC 2000*, volume 1, pages 458–463. IEEE, 2000.

[228] K. H. Liu and C. G. Xu. A genetic programming-based approach to the classification of multiclass microarray datasets. *Bioinformatics*, 25(3):331–337, 2009.

[229] X. Llorà and J. M. Garrell. Evolution of decision trees. In *CCIA'2001*, pages 115–122. ACIA Press, 2001.

[230] X. Llorà and J. M. Garrell. Inducing partially-defined instances with evolutionary algorithms. In C. E. Brodley et al., editors, *ICML 2001*, pages 337–344. Morgan Kaufmann, 2001.

[231] X. Llorà and J. M. Garrell. Coevolving Different Knowledge Representations With Fine-grained Parallel Learning Classifier Systems. In W. B. Langdon et al., editors, *GECCO 2002*, pages 934–941. Morgan Kaufmann, 2002.

[232] X. Llorà and J. M. Garrell. Prototype induction and attribute selection via evolutionary algorithms. *Intelligent Data Analysis*, 7(3):193–208, 2003.

[233] X. Llorà and S. W. Wilson. Mixed decision trees: Minimizing knowledge representation bias in LCS. In K. Deb et al., editors, *GECCO 2004*, volume 3103 of *LNCS*, pages 797–809. Springer, 2004.

[234] W. Y. Loh. Fifty Years of Classification and Regression Trees. *Int. Statistical Review*, 82(3):329–348, 2014.

[235] W. Y. Loh and Y. S. Shih. Split selection methods for classification trees. *Statistica sinica*, 7(4):815–840, 1997.

[236] J. D. Lohn, W. F. Kraus, and G. L. Haith. Comparing a coevolutionary genetic algorithm for multiobjective optimization. In *CEC'02*, volume 2, pages 1157–1162. IEEE, 2002.

[237] S. Lomax and S. Vadera. A Survey of Cost-sensitive Decision Tree Induction Algorithms. *ACM Computing Surveys*, 45(2):16:1–16:35, 2013.

[238] R. A. Lopes, A. R. R. Freitas, R. C. P. Silva, and F. G. Guimarães. Differential evolution and perceptron decision trees for classification tasks. In H. Yin, J. A. F. Costa, and G. Barreto, editors, *Proceedings of the 13th International Conference Intelligent Data Engineering and Automated Learning (IDEAL 2012)*, volume 7435 of *LNCS*, pages 550–557. Springer, Natal, Brazil, 2012.

[239] J. Luengo, S. García, and 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 Systems with Applications*, 36(4):7798–7808, 2009.

[240] J. F. Lutsko and B. Kuijpers. Simulated annealing in the construction of near-optimal decision trees. In P. Cheeseman et al., editors, *Selecting Models from Data*, volume 89 of *LNCS*, pages 453–462. Springer, 1994.

[241] R. E. Marmelstein and G. B. Lamont. Pattern Classification using a Hybrid Genetic Program Decision Tree Approach. In J. R. Koza et al., editors, *GP-98*, pages 223–231. Morgan Kaufmann, 22–25 1998.

[242] J. McCarthy. Recursive functions of symbolic expressions and their computation by machine, Part I. *Communications of the ACM*, 3(4):184–195, 1960.

[243] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[244] A. McGovern, K. L. Elmore, D. J. Gagne-II, S. E. Haupt, C. D. Karstens, R. Lagerquist, T. Smith, and J. K. Williams. Using Artificial Intelligence to Improve Real-Time Decision Making for High-Impact Weather. *Bulletin of the American Meteorological Society*, March 2017.

[245] R. R. F. Mendes, F. B. de Voznika, A. A. Freitas, and J. C. Nievola. Discovering fuzzy classification rules with genetic programming and co-evolution. In L. De Raedt et al., editors, *PKDD 2001*, volume 2168 of *LNAI*, pages 314–325. Springer, 2001.

[246] Z. Michalewicz and C. Z. Janikow. GENOCOP: a genetic algorithm for numerical optimization problems with linear constraints. *Communications of the ACM*, 39(12es):175, 1996.

[247] J. Mingers. Expert systems - rule induction with statistical data. *Journal of the Operational Research Society*, 38(1):39–47, 1987.

[248] J. Mingers. An Empirical Comparison of Pruning Methods for Decision Tree Induction. *Machine Learning*, 4(2):227–243, 1989.

[249] L. Mingyong and C. Erbao. An improved differential evolution algorithm for vehicle routing problem with simultaneous pickups and deliveries and time windows. *Engineering Applications of Artificial Intelligence*, 23(2):188–195, 2010.

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

[251] D. J. Montana. Strongly typed genetic programming. *Evolutionary computation*, 3(2):199–230, 1995.

[252] J. N. Morgan and J. A. Sonquist. Problems in the analysis of survey data, and a proposal. *Journal of the American statistical association*, 58(302):415–434, 1963.

[253] S. Morishita. On classification and regression. In S. Arikawa and H. Motoda, editors, *Proceedings of the First International Conference on Discovery Science (DS'98)*, volume 1532 of *LNCS*, pages 40–57. Springer, Fukuoka, Japan, 1998.

[254] A. A. Motsinger-Reif, S. Deodhar, S. J. Winham, and N. E. Hardison. Grammatical evolution decision trees for detecting gene-gene interactions. *BioData mining*, 3(1):1, 2010.

[255] E. M. Mugambi and A. Hunter. Multi-objective genetic programming optimization of decision trees for classifying medical data. In V. Palade et al., editors, *KES 2003*, pages 293–299. Springer, 2003.

[256] E. M. Mugambi, A. Hunter, G. Oatley, and L. Kennedy. Polynomial-fuzzy decision tree structures for classifying medical data. *Knowledge-Based Systems*, 17(2–4):81–87, 2004.

[257] P. M. Murphy and M. J. Pazzani. Exploring the decision forest: An empirical investigation of Occam's razor in decision tree induction. *Journal of Artificial Intelligence Research*, 1:257–275, 1994.

[258] F. Murtagh. Multilayer perceptrons for classification and regression. *Neurocomputing*, 2(5):183–197, 1991.

[259] S. K. Murthy. Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey. *Data Mining and Knowledge Discovery*, 2(4):345–389, 1998.

[260] S. K. Murthy, S. Kasif, and S. Salzberg. A System for Induction of Oblique Decision Trees. *Journal of Artificial Intelligence Research*, 2(1):1–32, 1994.

[261] S. K. Murthy, S. Kasif, S. Salzberg, and R. Beigel. OC1: A randomized algorithm for building oblique decision trees. In *AAAI'93*, volume 93, pages 322–327. AAAI press, 1993.

[262] P. Nemenyi. Distribution-free multiple comparisons. *Biometrics*, 18(2):263, 1962.

[263] F. Neri and V. Tirronen. Recent advances in differential evolution: a survey and experimental analysis. *Artificial Intelligence Review*, 33(1-2):61–106, 2010.

[264] D. F. Nettleton, A. Orriols-Puig, and A. Fornells. A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial Intelligence Review*, 33(4):275–306, 2010.

[265] S. C. Ng and K. S. Leung. Induction of quadratic decision trees using genetic algorithms. In *Proc. of 2003 Intelligent Automation Conf.*, pages 979–984, 2003.

[266] S. C. Ng and K. S. Leung. Induction of quadratic decision trees using genetic algorithms and k-d trees. *WSEAS Transactions on Computers*, 3(3):839–845, 2004.

[267] A. Niimi and E. Tazaki. Genetic programming combined with association rule algorithm for decision tree construction. In *KES 2000*, volume 2, pages 746–749. IEEE, 2000.

[268] N. I. Nikolaev and V. Slavov. Inductive genetic programming with decision trees. In M. van Someren et al., editors, *ECML-97 Part II*, volume 1224 of *LNCS*, pages 183–190. Springer, 1997.

[269] M. Norouzi, M. Collins, M. A. Johnson, D. J. Fleet, and P. Kohli. Efficient Non-greedy Optimization of Decision Trees. In C. Cortes et al., editors, *Advances in Neural Information Processing Systems*, volume 28, pages 1729–1737. Curran, 2015.

[270] S. W. Norton. Generating Better Decision Trees. In *IJCAI'89*, pages 800–805. Morgan Kaufmann, 1989.

[271] M. Núñez. The use of background knowledge in decision tree induction. *Machine Learning*, 6(3):231–250, 1991.

[272] S. Oka and Q. Zhao. Design of decision trees through integration of C4.5 and GP. In A. Namatame et al., editors, *Proc. of the 4th Japan-Australia Joint Workshop on Intelligent and Evolutionary Systems*, pages 128–135, 2000.

[273] M. Oltean and D. Dumitrescu. Multi expression programming. Technical Report UBB-01-2002, Babes-Bolyai University, Cluj-Napoca, Romania, 2002.

[274] A. Omielan and S. Vadera. ECCO: A New Evolutionary Classifier with Cost Optimisation. In Z. Shi, D. Leake, and S. Vadera, editors, *Proceedings of the 7th IFIP TC 12 International Conference of Intelligent Information Processing VI (IIP 2012)*, volume 385 of *IFIP ICT*, pages 97–105. Springer, Guilin, China, 2012.

[275] G. C. Onwubolu. *Differential Evolution for the Flow Shop Scheduling Problem*, pages 585–611. Springer, 2004.

[276] G. C. Onwubolu and D. Davendra. Scheduling flow shops using differential evolution algorithm. *European Journal of Operational Research*, 171(2):674–692, 2006.

[277] C. Orsenigo and C. Vercellis. Discrete support vector decision trees via tabu search. *Computational statistics & data analysis*, 47(2):311–322, 2004.

[278] F. E. B. Otero, A. A. Freitas, and C. G. Johnson. Inducing decision trees with an ant colony optimization algorithm. *Applied Soft Computing*, 2012.

[279] J. Pacheco, E. Alfaro, S. Casado, M. Gámez, and N. García. A GRASP method for building classification trees. *Expert Systems with Applications*, 39(3):3241–3248, 2012.

[280] N. R. Pal, S. Chakraborty, and A. Bagchi. RID3: An ID3-like algorithm for real data. *Information Sciences*, 96(3–4):271–290, 1997.

[281] M. A. Panduro, C. A. Brizuela, L. I. Balderas, and D. A. Acosta. A comparison of genetic algorithms, particle swarm optimization and the differential evolution method for the design of scannable circular antenna arrays. *Progress In Electromagnetics Research B*, 13:171–186, 2009.

[282] J. M. Pangilinan and G. K. Janssens. Pareto-optimality of oblique decision trees from evolutionary algorithms. *Journal of Global Optimization*, 51(2):301–311, 2011.

[283] A. Papagelis and D. Kalles. GA Tree: Genetically Evolved Decision Trees. In *ICTAI 2000*, pages 203–2006. IEEE, 2000.

[284] A. Papagelis and D. Kalles. Breeding Decision Trees Using Evolutionary Techniques. In C. E. Brodley et al., editors, *ICML'01*, pages 393–400. Morgan Kaufmann, 2001.

[285] W. Pedrycz and Z. A. Sosnowski. C-fuzzy decision trees. *IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 35(4):498–511, 2005.

[286] W. Pedrycz and Z. A. Sosnowski. Genetically optimized fuzzy decision trees. *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, 35(3):633–641, 2005.

[287] Y. Peng and 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):34, 2001.

[288] V. P. Plagianakos, D. K. Tasoulis, and M. N. Vrahatis. *A Review of Major Application Areas of Differential Evolution*, pages 197–238. Springer, 2008.

[289] V. Podgorelec and S. Karakatic. A Multi-Population Genetic Algorithm for Inducing Balanced Decision Trees on Telecommunications Churn Data. *Elektronika ir Elektrotechnika*, 19(6):121–124, 2013.

[290] V. Podgorelec, S. Karakatic, R. C. Barros, and M. P. Basgalupp. Evolving balanced decision trees with a multi-population genetic algorithm. In *CEC 2015*, pages 54–61. IEEE, 2015.

[291] V. Podgorelec and P. Kokol. Evolutionary construction of medical decision trees. In H. K. Chang et al., editors, *IEEE-EMBS*, volume 3, pages 1202–1205. IEEE, 1998.

[292] V. Podgorelec and P. Kokol. Induction of medical decision trees with genetic algorithms. In *ISCS 1999*, pages 1–7. Academic Press, 1999.

[293] V. Podgorelec and P. Kokol. Towards more optimal medical diagnosing with evolutionary algorithms. *Journal of Medical Systems*, 25(3):195–219, 2001.

[294] V. Podgorelec and P. Kokol. Evolutionary induced decision trees for dangerous software modules prediction. *Information Processing Letters*, 82(1):31–38, 2002.

[295] R. Poli, W. B. Langdon, N. F. McPhee, and J. R. Koza. *A Field Guide to Genetic Programming*. Lulu.com, 2008.

[296] M. A. Potter and K. A. DeJong. A Cooperative Coevolutionary Approach to Function Optimization. In Y. Davidor et al., editors, *PPSN III*, volume 866 of *LNCS*, pages 249–257. Springer, 1994.

[297] R. S. Prado, R. C. P. Silva, F. G. Guimarães, and O. M. Neto. Using differential evolution for combinatorial optimization: A general approach. In *Proc. of the Int. Conf. on Systems, Man and Cybernetics*, pages 11–18. IEEE, 2010.

[298] K. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Springer, 2006.

[299] B. Qian, L. Wang, R. Hu, W.L. Wang, D.X. Huang, and X. Wang. A hybrid differential evolution method for permutation flow-shop scheduling. *The Int. Journal of Advanced Manufacturing Technology*, 38(7):757–777, 2008.

[300] L. Qu, Y. Min, W. Weihong, and C. Xiaohong. Dynamic split-point selection method for decision tree evolved by gene expression programming. In *Proceedings of the 2009 IEEE Congress on Evolutionary Computation (CEC'09)*, pages 736–740, Trondheim, Norway, 2009. IEEE.

[301] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[302] J. R. Quinlan. Simplifying decision trees. *Int. Journal of Human-Computer Studies*, 27(3):221–234, 1987.

[303] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[304] J. R. Quinlan. Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research*, 4(1):77–90, 1996.

[305] J. R. Quinlan and R. L. Rivest. Inferring decision trees using the minimum description lenght principle. *Information and computation*, 80(3):227–248, 1989.

[306] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Problemata, 15. Frommann-Holzboog, 1973.

[307] R. Reed. Pruning algorithms-a survey. *IEEE Trans. on Neural Networks*, 4(5):740–747, 1993.

[308] G. Ritschard. CHAID and earlier supervised tree methods. In J. J. McArdle et al., editors, *Contemporary Issues in Exploratory Data Mining in the Behavioral Sciences*. Taylor & Francis, 2013.

[309] K. Rodriguez-Vazquez, C. M. Fonseca, and P. J. Fleming. Identifying the structure of nonlinear dynamic systems using multiobjective genetic programming. *IEEE Trans. on Systems, Man, and Cybernetics–Part A: Systems and Humans*, 34(4):531–545, 2004.

[310] L. Rokach and O. Maimon. Top-down induction of decision trees classifiers – a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(4):476–487, 2005.

[311] S. E. Rouwhorst and A. P. Engelbrecht. Searching the forest: Using decision trees as building blocks for evolutionary search in classification databases. In *CEC-2000*, volume 1, pages 633–638. IEEE, 2000.

[312] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In J. A. Anderson and E. Rosenfeld, editors, *Neurocomputing: Foundations of Research*, pages 673–695. MIT Press, 1988.

[313] C. Ryan, J. J. Collins, and M. O. Neill. Grammatical evolution: Evolving programs for an arbitrary language. In W. Banzhaf et al., editors, *EuroGP'98*, volume 1291 of *LNCS*, pages 83–96. Springer, 1998.

[314] M. D. Ryan and V. J. Rayward-Smith. The evolution of decision trees. In J. R. Koza et al., editors, *GP-98*, pages 350–358. Morgan Kaufmann, 1998.

[315] S. R. Safavian and D. Landgrebe. A Survey of Decision Tree Classifier Methodology. *IEEE Trans. on Systems, Man and Cybernetics*, 21(3):660–674, 1991.

[316] J. Sanz, H. Bustince, A. Fernández, and F. Herrera. IIVFDT: Ignorance functions based interval-valued fuzzy decision tree with genetic tuning. *Int. Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 20(supp02):1–30, 2012.

[317] M. Saremi and F. Yaghmaee. Evolutionary decision tree induction with multi-interval discretization. In *ICIS 2014*, pages 1–6. IEEE, 2014.

[318] J. B. Sathe and M. P. Mali. A hybrid Sentiment Classification method using Neural Network and Fuzzy Logic. In *Proceedings of the 11th International Conference on Intelligent Systems and Control (ISCO 2017)*, pages 93–96, Coimbatore, India, 2017. IEEE.

[319] J. D. Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In J. J. Grefenstette, editor, *ICGA-85*, pages 93–100. L. Erlbaum, 1985.

[320] H. P. Schwefel. *Evolutionsstrategie und numerische Optimierung*. Technische Universität Berlin, 1975.

[321] E. V. Sekar, J. Anuradha, A. Arya, B. Balusamy, and V. Chang. A framework for smart traffic management using hybrid clustering techniques. *Cluster Computing*, May 2017.

[322] S. Shah and 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):494–505, 1999.

[323] A. Shali, M. R. Kangavari, and B. Bina. Using genetic programming for the induction of oblique decision trees. In M. Arif-Wani, editor, *ICMLA 2007*, pages 38–43. IEEE, 2007.

[324] H. Shi. Best-first decision tree learning. Master's thesis, University of Waikato, 2007.

[325] M. Shirasaka, Q. Zhao, O. Hammami, K. Kuroda, and K. Saito. Automatic design of binary decision trees based on genetic programming. In *SEAL'98*, 1998.

[326] S. K. Shukla and M. K. Tiwari. Soft decision trees: A genetically optimized cluster oriented approach. *Expert Systems with Applications*, 36(1):551–563, 2009.

[327] E. V. Siegel. Competitively evolving decision trees against fixed training cases for natural language processing. *Advances in Genetic Programming*, 19:409–423, 1994.

[328] S. Silva. Reassembling Operator Equalisation: A Secret Revealed. In N. Krasnogor, editor, *GECCO'11*, pages 1395–1402. ACM, 2011.

[329] S. Silva and L. Vanneschi. The importance of being flat–studying the program length distributions of operator equalisation. In *Genetic Programming Theory and Practice IX*, pages 211–233. Springer, 2011.

[330] V. Slavov and N. I. Nikolaev. Fitness Landscapes and Inductive Genetic Programming. In G. D. Smith et al., editors, *Artificial Neural Nets and Genetic Algorithms*, pages 414–418. Springer, 1998.

[331] S. F. Smith. RNA search acceleration with genetic algorithm generated decision trees. In *Proceedings of the 7th International Conference on Machine Learning and Applications (ICMLA'08)*, pages 565–570, San Diego, CA, USA, 2008. IEEE.

[332] M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

[333] K. Sörensen and G. K. Janssens. Data mining with genetic algorithms on binary trees. *European Journal of Operational Research*, 151(2):253–264, 2003.

[334] Kent A. Spackman. Signal Detection Theory: Valuable Tools for Evaluating Inductive Learning. In *Proc. of the 6th Int. Workshop on Machine Learning*, pages 160–163. Morgan Kaufmann, 1989.

[335] M. Sprogar, P. Kokol, M. Zorman, V. Podgorelec, L. Lhotska, and J. Klema. Evolving groups of basic decision trees. In *CBMS 2001*, pages 183–188. IEEE, 2001.

[336] N. Srinivas and K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248, 1994.

[337] H. Steinhaus. Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci*, 1:801–804, 1956.

[338] G. Stiglic, S. Kocbek, I. Pernek, and P. Kokol. Comprehensive decision tree models in bioinformatics. *PloS one*, 7(3):1–13, 2012.

[339] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the royal statistical society. Series B (Methodological)*, pages 111–147, 1974.

[340] R. Storn and K. Price. Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[341] W. N. Street. Oblique multicategory decision trees using nonlinear programming. *INFORMS Journal on Computing*, 17(1):25–31, 2005.

[342] C. Strobl, J. Malley, and G. Tutz. An Introduction to Recursive Partitioning: Rationale, Application, and Characteristics of Classification and Regression Trees, Bagging, and Random Forests. *Psychological Methods*, 14(4):323–348, 2009.

[343] R. Struharik, V. Vranjkovic, S. Dautovic, and L. Novak. Inducing oblique decision trees. In *Proceedings of the 12th International Symposium on Intelligent Systems and Informatics (SISY–2014)*, pages 257–262, Subotica, Serbia, 2014. IEEE.

[344] C. Suenderhauf, F. Hammann, and J. Huwyler. Computational prediction of blood-brain barrier permeability using decision tree induction. *Molecules*, 17(9):10429–10445, 2012.

[345] W. A. Tackett. Genetic Programming for Feature Discovery and Image Discrimination. In S. Forrester, editor, *ICGA-93*, pages 303–311. Morgan Kaufmann, 1993.

[346] E. G. Talbi. *Metaheuristics: From Design to Implementation*. Wiley, 2006.

[347] M. Tan. Cost-Sensitive Learning of Classification Knowledge and Its Applications in Robotics. *Machine Learning*, 13(1):7–33, 1989.

[348] T. Tanigawa and Q. Zhao. A Study on Efficient Generation of Decision Trees Using Genetic Programming. In D. Whitley et al., editors, *GECCO-2000*, pages 1047–1052. Morgan Kaufmann, 2000.

[349] M.F. Tasgetiren, M. Sevkli, Y.C. Liang, and G. Gencyilmaz. Particle swarm optimization algorithm for single machine total weighted tardiness problem. In *Proceedings of the Congress on Evolutionary Computation (CEC2004)*, pages 1412–1419, Portland, OR, USA, 2004. IEEE.

[350] M.F. Tasgetiren, P.N. Suganthan, and Q.K. Pan. An ensemble of discrete differential evolution algorithms for solving the generalized traveling salesman problem. *Applied Mathematics and Computation*, 215(9):3356–3368, 2010.

[351] T. M. Therneau and E. J. Atkinson. An introduction to recursive partitioning using the RPART routines. Technical report, Mayo Foundation, 1997.

[352] C. To and T. D. Pham. Analysis of cardiac imaging data using decision tree based parallel genetic programming. In *ISPA 2009*, pages 317–320. IEEE, 2009.

[353] T. Toffoli and N. Margolus. *Cellular Automata Machines: A New Environment for Modeling*. MIT Press, 1987.

[354] L. Trujillo, L. Muñoz, E. Galván-López, and S. Silva. neat Genetic Programming: Controlling bloat naturally. *Information Sciences*, 333:21–43, 2016.

[355] A. Tsakonas. A comparison of classification accuracy of four genetic programming-evolved intelligent structures. *Information Sciences*, 176(6):691–724, 2006.

[356] E. P. K. Tsang and J. Li. Combining ordinal financial predictions with genetic programming. In K. S. Leung et al., editors, *IDEAL 2000*, volume 1983 of *LNCS*, pages 532–537. Springer, 2000.

[357] E. P. K. Tsang, J. Li, and J. M. Butler. EDDIE Beats the Bookies. *Software: Practice and Experience*, 28(10):1033–1043, August 1998.

[358] G. K. F. Tso and K. K. W. Yau. Predicting electricity energy consumption: A comparison of regression analysis, decision tree and neural networks. *Energy*, 32(9):1761–1768, 2007.

[359] J.W. Tukey. Comparing individual means in the analysis of variance. *Biometrics*, pages 99–114, 1949.

[360] G Tür and H. A. Güvenir. Decision tree induction using genetic programming. In E. Alpaydin, editor, *TAINN'96*, pages 187–196. Bogazici University Press, 1996.

[361] P. D. Turney. Cost-sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm. *Journal of Artificial Intelligence Research*, 2(1):369–409, 1995.

[362] T. Tušar. Optimizing accuracy and size of decision trees. In *Proceedings of the 16th International Electrotechnical and Computer Science Conference (ERK-2007)*, pages 81–84, Portorož, Slovenia, 2007.

[363] K. P. Unnikrishnan and K. P. Venugopal. Alopex: A correlation-based learning algorithm for feedforward and recurrent neural networks. *Neural Computation*, 6(3):469–490, 1994.

[364] P. E. Utgoff. Incremental induction of decision trees. *Machine learning*, 4(2):161–186, 1989.

[365] P. E. Utgoff and C. E. Brodley. Linear machine decision trees. Technical report, University of Massachusetts, Amherst, MA, USA, 1991.

[366] S. Vadera. CSNL: A Cost-sensitive Non-linear Decision Tree Algorithm. *ACM Trans. on Knowledge Discovery from Data*, 4(2):6:1–6:25, May 2010.

[367] V. Vapnik. *Estimation of dependences based on empirical data*, volume 41 of *Information Science and Statistics*. Springer, 1982.

[368] C. B. Veenhuis, M. Koppen, J. Kruger, and B. Nickolay. Tree swarm optimization: An approach to PSO-based tree discovery. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, volume 2, pages 1238–1245, Edinburgh, Scotland, 2005. IEEE.

[369] A. Vella, D. Corne, and C. Murphy. Hyper-heuristic decision tree induction. In A. Abraham et al., editors, *NaBIC 2009*, pages 409–414. IEEE, 2009.

[370] J. L. Verdegay, R. R. Yager, and P. P. Bonissone. On heuristics as a fundamental constituent of soft computing. *Fuzzy Sets and Systems*, 159(7):846 – 855, 2008.

[371] J. Vesterstrøm and R. Thomsen. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In *Congress on Evolutionary Computation, 2004. CEC2004*, volume 2, pages 1980–1987. IEEE, 2004.

[372] T. Vicsek and A. Zafeiris. Collective motion. *Physics Reports*, 517(3–4):71 – 140, 2012.

[373] B. Vukobratovic and R. Struharik. Evolving full oblique decision trees. In *CINTI 2015*, pages 95–100. IEEE, 2015.

[374] M. Wall. GAlib: A C++ library of genetic algorithm components, 1996. Mechanical Engineering Department, Massachusetts Institute of Technology.

[375] P. Wang, K. Tang, E. P. K. Tsang, and X. Yao. A memetic genetic programming with decision tree-based local search for classification problems. In *CEC 2011*, pages 917–924. IEEE, 2011.

[376] P. Wang, K. Tang, T. Weise, E. P. K. Tsang, and X. Yao. Multiobjective genetic programming for maximizing ROC performance. *Neurocomputing*, 125:102–118, 2014.

[377] P. Wang, T. Weise, and R. Chiong. Novel evolutionary algorithms for supervised classification problems: an experimental study. *Evolutionary Intelligence*, 4(1):3–16, 2011.

[378] W. Wang, Q. Li, S. Han, and H. Lin. A preliminary study on constructing decision tree with gene expression programming. In J. S. Pan, P. Shi, and Y. Zhao, editors, *Proceedings of the First International Conference on Innovative Computing, Information and Control (ICICIC'06)*, volume I, pages 222–225, Beijing, China, 2006. IEEE.

[379] X. Wang, B. Chen, G. Qian, and F. Ye. On the optimization of fuzzy decision trees. *Fuzzy Sets and Systems*, 112(1):117–125, 2000.

[380] X. Z. Wang, F. V. Buontempo, A. Young, and D. Osborn. Induction of decision trees using genetic programming for modelling ecotoxicity data: adaptive discretization of real-valued endpoints. *SAR and QSAR in Environmental Research*, 17(5):451–471, 2006.

[381] X. Z. Wang, D. S. Yeung, and 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):215–226, 2001.

[382] W. Weihong, R. Wei, and L. Qu. Fuzzy decision tree construction with gene expression programming. In *ISKE 2010*, pages 244–248. IEEE, 2010.

[383] P. Whigham. Grammatically-based genetic programming. In *Proc. of the Workshop on Genetic Programming: from theory to real-world applications*, pages 33–41, 1995.

[384] A. P. White and W. Z. Liu. Bias in Information-Based Measures in Decision Tree Induction. *Machine Learning*, 15(3):321–329, 1994.

[385] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[386] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2005.

[387] I.H. Witten, E. Frank, L.E. Trigg, M.A. Hall, G. Holmes, and S.J. Cunningham. Weka: Practical machine learning tools and techniques with Java implementations. Technical Report 11, Department of Computer Science, The University of Waikato, 1999.

[388] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, and S. Y. Philip. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.

[389] Z. Xin, L. Hua, X. H. Wang, D. Zhao, C.G. Yu, Y. H. Ma, L. Zhao, X. Cao, and J.K. Yang. Reanalysis and external validation of a decision tree model for detecting unrecognized diabetes in rural chinese individuals. *International Journal of Endocrinology*, 2017(ID 3894870):1–6, 2017.

[390] S. B. Yang. Fuzzy variable-branch decision tree. *Journal of Electronic Imaging*, 19(4):043012–043012–9, 2010.

[391] D. S. Yeung, X. Z. Wang, and E. C. C. Tsang. Learning weighted fuzzy rules from examples with mixed attributes by fuzzy decision trees. In *SMC'99*, volume 3, pages 349–354. IEEE, 1999.

[392] L. Yi and K. Wanli. A New Genetic Programming Algorithm for Building Decision Tree. *Procedia Engineering*, 15:3658–3662, 2011.

[393] Y. Yuan and M. J. Shaw. Induction of fuzzy decision trees. *Fuzzy Sets and Systems*, 69(2):125–139, 1995.

[394] L. A. Zadeh. Fuzzy logic, neural networks, and soft computing. *Communications of the ACM*, 37(3):77–84, 1994.

[395] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338 – 353, 1965.

[396] K. Zhang, Z. Xu, and B. P. Buckles. Oblique decision tree induction using multimembered evolution strategies. In B. V. Dasarathy, editor, *Proceeding of Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security, SPIE 2005*, volume 5812, pages 263–270, Orlando, Florida, 2005. SPIE.

[397] Q. Zhang and H. Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. on Evolutionary Computation*, 11(6):712–731, 2007.

[398] Y. Zhang and Y. Yang. Cross-validation for selecting a model selection procedure. *Journal of Econometrics*, 187(1):95–112, 2015.

[399] H. Zhao. A multi-objective genetic programming approach to developing Pareto optimal decision trees. *Decision Support Systems*, 43(3):809–826, 2007.

[400] Q. Zhao and M. Shirasaka. A study on evolutionary design of binary decision trees. In *CEC'99*, volume 3. IEEE, 1999.

[401] K. Zielinski, P. Weitkemper, R. Laur, and K. D. Kammeyer. Parameter Study for Differential Evolution Using a Power Allocation Problem Including Interference Cancellation. In *2006 IEEE International Conference on Evolutionary Computation*, pages 1857–1864, 2006.

[402] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm. Technical report, Eidgenössische Technische Hochschule Zürich (ETH), Institut für Technische Informatik und Kommunikationsnetze (TIK), Zürich, Switzerland, 2001.

[403] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms — A comparative case study. In A. E. Eiben et al., editors, *PPSN V*, pages 292–301. Springer, 1998.

Differential-Evolution-based methods for inducing Decision Trees

INFORME DE ORIGINALIDAD

20%
ÍNDICE DE SIMILITUD

FUENTES PRIMARIAS

1	ri.ujat.mx Internet	3672 palabras — 8%
2	Rafael Rivera-Lopez, Juana Canul-Reich, Efrén Mezura-Montes, Marco Antonio Cruz-Chávez. "Induction of decision trees as classification models through metaheuristics", Swarm and Evolutionary Computation, 2021 Crossref	2418 palabras — 5%
3	cwww.intechopen.com Internet	1339 palabras — 3%
4	Rafael Rivera-Lopez, Juana Canul-Reich. "Construction of near-optimal axis-parallel decision trees using a differential-evolution-based approach", IEEE Access, 2018 Crossref	1033 palabras — 2%
5	Richter, Urban Maximilian. "Controlled self-organisation using learning classifier systems", KIT Scientific Publishing, Karlsruhe, 2009. Publicaciones	204 palabras — < 1%
6	"Artificial Intelligence and Soft Computing", Springer Nature, 2017 Crossref	200 palabras — < 1%
7	mts.intechopen.com Internet	184 palabras — < 1%

8	"Advances in Artificial Intelligence", Springer Science and Business Media LLC, 2017 Crossref	175 palabras — < 1%
9	www.intechopen.com Internet	89 palabras — < 1%
10	"Advances in Computational Intelligence. MICAI 2023 International Workshops", Springer Science and Business Media LLC, 2024 Crossref	62 palabras — < 1%
11	www.yumpu.com Internet	26 palabras — < 1%
12	Khoshgoftaar, Taghi M., Yi Liu, and Naeem Seliya. "Tree-Based Software Quality Classification Using Genetic Programming", Series on Quality Reliability and Engineering Statistics, 2006. Crossref	24 palabras — < 1%
13	Rafael Rivera-López, Efrén Mezura-Montes, Juana Canul-Reich, Marco-Antonio Cruz-Chávez. "An Experimental Comparison of Self-Adaptive Differential Evolution Algorithms to Induce Oblique Decision Trees", Mathematical and Computational Applications, 2024 Crossref	24 palabras — < 1%
14	osuluqojad.tk Internet	22 palabras — < 1%

EXCLUIR CITAS	ACTIVADO	EXCLUIR FUENTES	DESACTIVADO
EXCLUIR BIBLIOGRAFÍA	ACTIVADO	EXCLUIR COINCIDENCIAS	< 22 PALABRAS