A Bi-criteria Optimization Model for Adjusting the Decision Tree Parameters

Mohammad Azad[1,*], Mikhail Moshkov[2]

[1] *Jouf University*
Department of Computer Science,College of Computer and Information Sciences
Sakaka 72441, Saudi Arabia

[2] *King Abdullah University of Science and Technology Computer,*
Electrical and Mathematical Sciences & Engineering Division
Thuwal 23955-6900, Saudi Arabia

**Corresponding author: mikhail.moshkov@kaust.edu.sa*

Abstract

Decision trees play a very important role in knowledge representation because of its simplicity and self-explanatory nature. We study the optimization of the parameters of the decision trees to find a shorter as well as more accurate decision tree. Since these two criteria are in conflict, we need to find a decision tree with suitable parameters that can be a trade off between two criteria. Hence, we design two algorithms to build a decision tree with a given threshold of the number of vertices based on the bi-criteria optimization technique. Then, we calculate the local and global misclassification rates for these trees. Our goal is to study the effect of changing the threshold for the bi-criteria optimization of the decision trees. We apply our algorithms to 13 decision tables from UCI Machine Learning Repository and recommend the suitable threshold that can give us more accurate decision trees with a reasonable number of vertices.

Keywords: Decision trees; Directed acyclic graph; Knowledge representation; Pareto optimal points; Misclassification; Bi-criteria optimization

1. Introduction

Decision trees are a very common and accepted model for the classifier (Azad *et al.*, 2015, 2018; Breiman *et al.*, 1984). They can also be used to find the patterns and represent knowledge from the data (Alsolami *et al.*, 2020; Azad *et al.*, 2019; Breiman *et al.*, 1984). Furthermore, it can be used as algorithms to solve problems (Moshkov, 2005; Rokach and Maimon, 2008). The problems considered in this paper are similar to the problems in medical diagnosis (Podgorelec *et al.*, 2002), image analysis (Mehmood *et al.*, 2018; Sarwar *et al.*, 2018), pattern recognition (Yahyaoui *et al.*, 2020) etc. We choose decision tree in this paper because of its easy to use nature and easily comprehensible and explainable by humans as well as directly exchangeable to production rules (Almuallim *et al.*, 2002). Optimizing the parameters of decision tree can be extremely useful based on the context: minimizing depth or average depth can be equivalent to minimizing the time complexity of decision trees, minimizing number of vertices or number of leaf/non-leaf vertices can be equivalent to minimizing the space complexity of decision trees and minimizing the number of misclassification can be equivalent to maximizing the accuracy of decision trees etc. In this paper, we consider optimizing three parameters of decision trees: number of vertices and misclassification rates (global and local).

It is important to consider not only optimizing single criteria but also optimizing two criteria at the same time which is popularly known as the bi-criteria optimization (BCO). BCO is a part of a broader class of optimization problems named as multi-criteria optimization (MCO). In the literature, we can find it in various names e.g., multiple objective discrete optimization, vector optimization, multiple objective combinatorial optimization, multi-attribute optimization, multiple criteria decision making, Pareto optimization (Chang, 2015; Ehrgott and Gandibleux, 2000; Kouvelis and Sayın, 2006), etc. Let us see some examples in which MCO has been applied successfully in many real world scientific and engineering problems where we choose optimal solutions as a trade-offs between two or more conflicting objectives. One of the examples is the maximizing performance while minimizing fuel consumption and emission of pollutants of a vehicle (Chang, 2015). Another example is the searching for new remedial drugs in the medical field: maximizing strength of the drug while minimizing synthesis costs and unwanted side effects (Emmerich Michael T. M., 2018; Rosenthal and Borschbach, 2017; van der Horst *et al.*, 2012). There are many other examples that brought the attention of the research community: the minimization of different types of error rates in machine learning (false positives, false negatives) (Wang *et al.*, 2015; Yevseyeva *et al.*, 2013), the optimization of delivery costs and inventory costs in logistics (Geiger and Sevaux, 2011), the optimization of building designs with respect to health, energy efficiency, and cost criteria (Hopfe *et al.*, 2012). In this paper, we consider BCO relative to the number of vertices and misclassification rate (either global or local).

In BCO and MCO, the objective functions are said to be in conflict and the goal is to find a solution that is optimal in some sense. The solutions that can not be improved in one objective without deteriorating the performance in any other objective are called Pareto optimal or nondominated solutions. They are named after Italian economist Vilfredo Pareto (de Weck, 2004).

In this paper, we build the set of Pareto optimal points (POPs) for BCO relative to the number of vertices and misclassification rate (either global or local). It is important to find a decision tree with a reasonable number of vertices to make it understandable and with a reasonable misclassification rate (either global or local) to extract accurate knowledge from it. In (Azad *et al.*, 2019), we mentioned that we should focus not only on the global misclassification rate but also on the local misclassification rate because the misclassifications may not be equally distributed among the leaf vertices. Besides, it may happen that the fraction of misclassifications can be high for some of the leaf vertices.

Previously, we created a dynamic programming (Bellman, 1954; Tian *et al.*, 2018) approach to BCO of decision trees for decision tables relative to two parameters (AbouEisha *et al.*, 2019). The idea is to first build a directed acyclic graph (DAG) then, from DAG (using special algorithm mentioned in (AbouEisha *et al.*, 2019)) build the set of POPs. After that, we can derive decision trees which parameter values will be in accordance with the desired POP. Note that, the time complexity of such process depends on the size of the considered table and, in the worst case, it is exponential. Therefore, from the practical point of view, we can work from small to medium sized decision tables.

In (Azad *et al.*, 2019), the designed algorithms based on BCO are pertinent to medium-sized decision tables with categorical features only and, sometimes, the number of vertices in the trees is very large. In (Azad, 2020*b*), instead of conventional decision trees, we study trees based on binary splits similar to CART (Classification and Regression Tree) (Breiman *et al.*, 1984). We call such trees as BS-trees. BS-trees use binary splits instead of the initial features. The standard CART tree uses in each internal vertex the best split among all features. A BS-tree can use in each internal vertex the best split for an arbitrary feature. It extends essentially the set of decision trees under consideration. In (Azad, 2020*a*), two algorithms ($G-19$ and $L-19$) have been designed that build decision trees with at most 19 vertices (at most 10 leaves and at most nine internal vertices). The choice of 19 vertices was inspired by (AbouEisha *et al.*, 2019).

In this paper, we try to understand the behavior of the parameters of the extracted decision trees if we change the value of the threshold from 19 to some other values like 21, 23, etc. Furthermore, we extend the algorithms $G-19$ and $L-19$ to algorithms G_α and L_α where α is a threshold that can take values $13,\ldots,35$ with step 2. We apply the considered algorithms to 13 data sets from the UCI Machine Learning Repository (Dua and Graff, 2017), and study three parameters of the created trees (number of

Table 1. Example of a decision table S_0

$$S_0 = \begin{array}{|cc|c|} \hline f_1 & f_2 & \\ \hline 0 & 0 & 2 \\ 0 & 1 & 2 \\ 1 & 0 & 3 \\ 1 & 1 & 1 \\ \hline \end{array}$$

nodes, global misclassification rates, local misclassification rates). For each threshold value, we took the average value of three parameters based on the 13 tables. Based on the achieved results, we can easily adjust the number of vertices and misclassification error rates of decision trees based on the considered bi-criteria optimization model. Moreover, we recommend the range of threshold values between 15 to 21 with step 2 which can be valuable for significant analysis of decision trees for knowledge representation.

The main contribution of the paper are (i) creation of two algorithms (G_α and L_α), (ii) performing the experiments on the state-of-art data sets from UCI ML Repository, and (iii) understanding the trade-off between two conflicting objective of number of nodes vs. misclassification rates when we fix the number of nodes to a threshold.

We arrange the remaining of the manuscript as follows. First, we explain the detailed methodology in Section 2. Then, we discuss the output of the experiments with data sets from the UCI Machine Learning Repository (Dua and Graff, 2017) in Sect. 3. Finally, Sect. 4 contains concise conclusion.

2. Detailed Methodology

Many researchers have spent a great amount of research of greedy algorithms (Azad *et al.*, 2013; Boryczka and Kozak, 2009; Breiman *et al.*, 1984; Chai *et al.*, 1996; Heath *et al.*, 1993; Quinlan, 2014) for building non-optimal decision trees. These algorithms are faster but can't guarantee the optimal parameters of decision trees and did not consider BCO. In addition, some other researchers have studied the exhaustive algorithms for building optimal decision trees (Garey, 1972; Schumacher and Sevcik, 1976). But they did not consider BCO problem of the optimal decision tree parameters. However, only few have concentrated the study on dynamic programming for bi-criteria (and multi-criteria) optimization of decision trees (AbouEisha *et al.*, 2019; Alsolami *et al.*, 2020; Azad, 2018).

In this section, we describe an expansion of dynamic programming, which gives all possible decision trees under consideration. The sequence we follow is first, we describe the preliminary ideas of decision tables, decision trees and building of directed acyclic graph (DAG) then we demonstrate the properties of BS-trees, later we explain the BCO, and finally, we explain two algorithms G_α and L_α.

2.1 Preliminaries of Decision Tables and Decision Trees

There are various types of decision tables: usual decision tables (AbouEisha *et al.*, 2019), decision tables with multiple decisions (Alsolami *et al.*, 2020; Azad, 2018) etc. In this paper, we consider usual decision table i.e., conventional single decision table where each row is attached with a single decision. We represent such decisions as nonnegative integers. Columns of this table are labeled with conditional attributes. Rows of the table are distinct. An example of a decision table S_0 is presented in Table 1. The table S_0 consists of 4 rows and 2 attributes (f_1 and f_2). We attach decisions $2, 2, 3, 1$ corresponding to the row numbers $1, 2, 3$ and 4 respectively.

A decision tree for a decision table is a finite rooted directed tree. In such trees, there are two types of vertices: non-leaf vertices and leaf vertices. We label the non-leaf vertices with the conditional attributes from the decision tables and the leaf vertices with nonnegative integers (e.g., decisions attached with rows). We label outgoing edges in each non-leaf vertex with pairwise nonnegative integers (attribute values). We depict an example of an exact decision tree for the decision table S_0 in Fig. 1.

Now, we will describe how to create decision trees for the given decision table using an expansion of dynamic programming. Previously, we described such a process in many publications (AbouEisha *et al.*, 2019; Aldilaijan *et al.*, 2019; Alsolami *et al.*, 2020; Azad, 2018; Azad and Moshkov, 2017). The

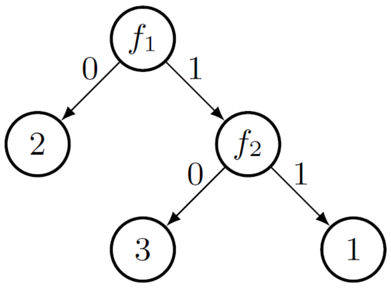


Fig. 1. A decision tree for the table S_0

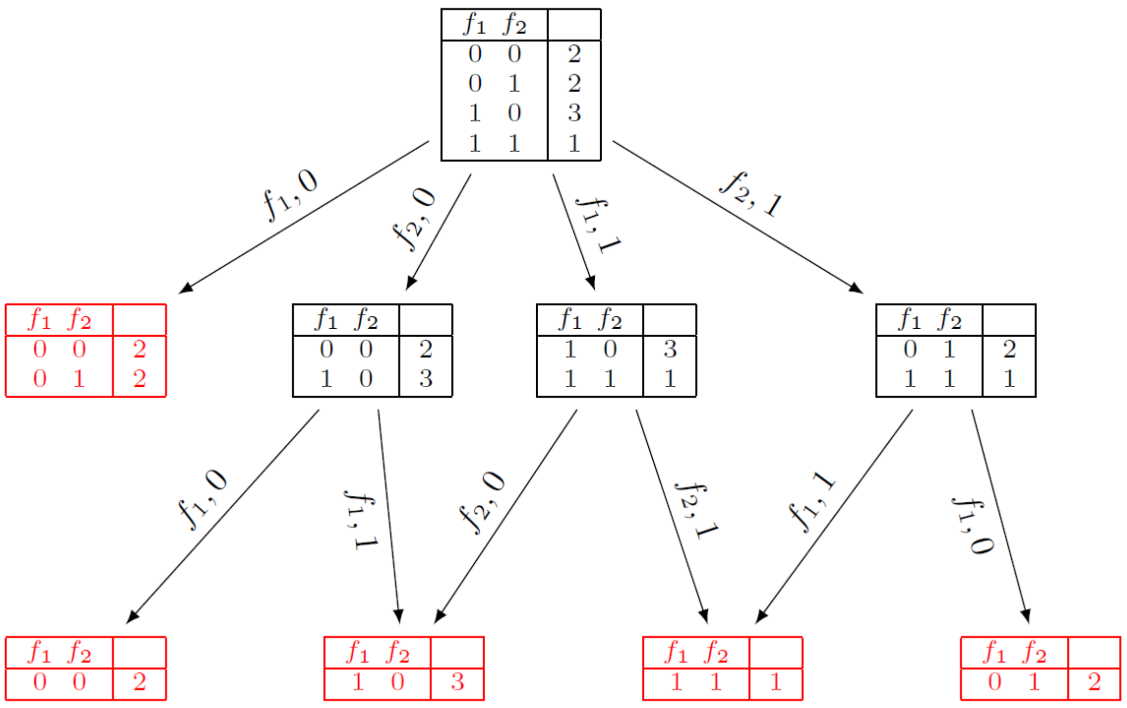


Fig. 2. The directed acyclic graph for the decision table S_0

idea is to build a directed acyclic graph (DAG) based on the expansion of dynamic programming. For example, a decision table S_0 is shown in Table 1 and corresponding DAG in Fig. 2.

After building the DAG, we can extract all decision trees for the decision table under consideration. The DAG in Fig. 2 describes two decision trees (illustrated in Fig. 3) for the decision table S_0. Even though the time complexity of the building of the DAG is exponential in the worst case subject to the size of the table, it is still possible to use this method for small to medium sized decision tables (AbouEisha *et al.*, 2019; Alsolami *et al.*, 2020; Azad, 2018).

There are many ways to split or partition the tables into subtables. The splits based on each value are considered in paper (Azad *et al.*, 2019) which are applicable for categorical attributes only. In this paper, we consider the binary split similar to (Azad *et al.*, 2015, 2018; Breiman *et al.*, 1984). The advantage of binary split in comparison to each value split is that it can be applied for both categorical and numerical attributes.

2.2 Binary Split Trees (BS-Trees)

Being motivated by CART algorithm, we consider BS-trees similar to trees in CART. The difference is that a BS-tree uses in each non-leaf vertex the optimal partition for an arbitrary attribute, whereas, the

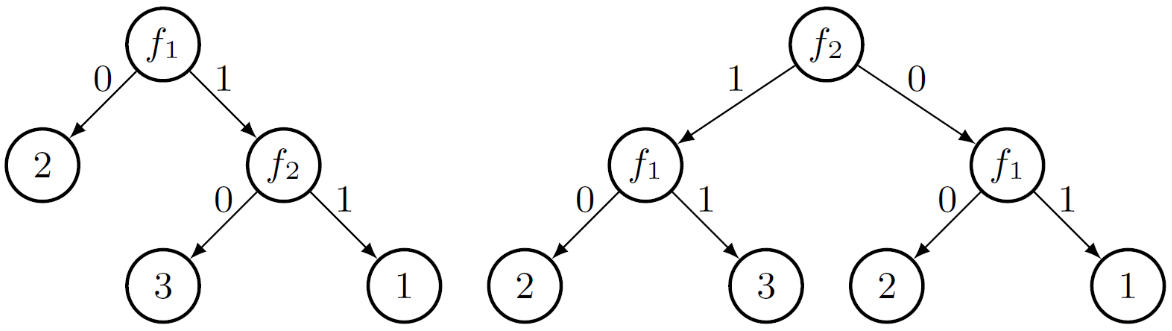


Fig. 3. Two possible decision trees for the table S_0

standard CART tree uses in each non-leaf vertex the optimal partition among all attributes.

Given a decision table S containing $f_1, \dots, f_n$ attributes (either numerical or categorical), we convert each attribute f_i $(1 \le i \le n)$ into binary attribute (as in CART) using the following procedure (we have two scenarios depending on the type of attributes):

1. Categorical attribute: Let A be the set of values of the categorical attribute f_i. Then A is partitioned into two nonempty subsets A_1 and A_0. If the value of attribute f_i belongs to A_0 then we make the binary split bs equal to 0. Similarly, if the value of attribute f_i belongs to A_1 then we make the binary split bs equal to 1. We need to take splits for all possible partitions of A.

2. Numerical attribute: Let us compare the values of the numerical attribute f_i with a real-value threshold ψ. The split bs is considered as 0 if f_i's value is smaller than ψ, and 1 otherwise. We consider splits for all possible thresholds ψ.

As a result of the partition bs, two subtables $S_{(bs=0)}$ and $S_{(bs=1)}$ are obtained. This is the main idea behind the binary partitioning.

In literature, "entropy" (Quinlan, 1986, 2014), "gini index" (AbouEisha *et al.*, 2019) and "abs" (Alsolami *et al.*, 2020; Azad, 2020*b*; Azad *et al.*, 2015, 2018) were used as uncertainty measures. In this paper, we apply "abs" as the uncertainty measure. We get two subtables $S_{(bs=0)}$ and $S_{(bs=1)}$ by the partition bs for the table S. We define the notion of impurity function $I(S, bs)$ as the weighted sum of uncertainties of these subtables i.e., $S_{(bs=0)}$ and $S_{(bs=1)}$ (the weights are based on the proportion of the total objects (rows) in the corresponding subtables). Now, we need to find the partition bs for which the value of $I(S, bs)$ is minimum after applying the partition bs for the attribute f_i in S. Then we call such partition as the best partition for the attribute f_i.

We consider BS-trees where every leaf vertex is labeled by a non-negative integer. We label every non-leaf vertex by a partition based on one of the attributes. Also, two arcs are going out from this non-leaf vertex (one is labeled with 0 and another with 1). Let us consider a decision tree δ and any vertex v of it. Then, we can map a subtable $S(\delta, v)$ for the given table S for each vertex v. This subtable $S(\delta, v)$ consists of all objects (rows) of the given table S for which the action of the decision tree proceeds through the vertex v. We build our tree based on the following assumptions:

1. For each non-leaf vertex v, $S(\delta, v)$ contains the objects (rows) that have different decisions. Furthermore, the vertex v is labeled by an optimal split for a non-constant attribute f_i on $S(\delta, v)$.

2. For each leaf vertex v, the vertex is labeled with a most frequent decision e.g., a decision which is bound with the largest amount of objects (rows) from $S(\delta, v)$.

We examine the following three parameters of the tree δ (Azad *et al.*, 2019):

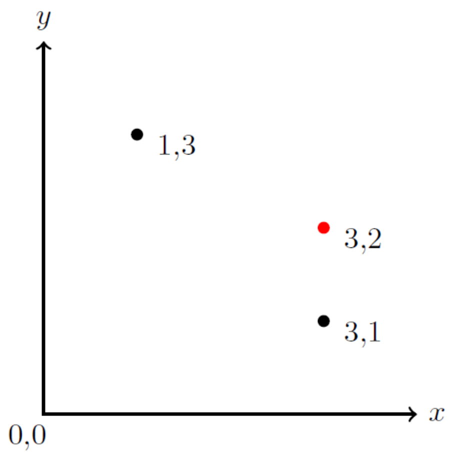


Fig. 4. Example of POPs. The black points are POPs and the red point is not POP

1. The number of vertices: we denote it by N. It is the total number of vertices including leaf and non-leaf vertices.
2. The global misclassification rate: we denote it by me^G. We calculate the number of misclassifications of δ on S and divide it by the number of rows in S.
3. The local misclassification rate: we denote it by me^L. For each leaf vertex of δ, we calculate the ratio of the number of misclassifications among rows accepted by this vertex to the number of all rows accepted by the vertex. Then, we take the maximum of the obtained values among all leaf vertices of δ.

One can show that the global misclassification rate is at most the local misclassification rate.

2.3 Bi-criteria Optimization (BCO) and Pareto Fronts

When we consider BCO in the presence of trade-offs between two objectives that may be in conflict, we need to find optimal solutions (pair of values of two o bjectives) (Chang, 2 015). For example, if we increase the number of vertices in the trees then the misclassification rate (either global or local) decreases and vice versa. A solution is called nondominated if one objective functions can not be improved in value without degrading another objective function value. We call such solutions as Pareto optimal points (POPs) (Chang, 2015).

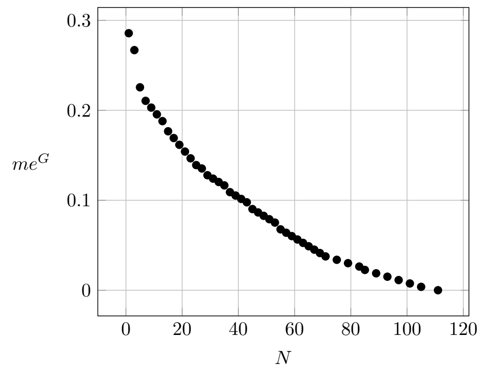
Let us consider a point (i_1, j_1) in D (where D is a finite set of points in a two-dimensional space). This point is called a Pareto optimal point (POP) for D if we do not find any other point (i_2, j_2) in D where $i_2 \leq i_1$ and $j_2 \leq j_1$ such that $(i_1, j_1) \neq (i_2, j_2)$ (AbouEisha *et al.*, 2019; Alsolami *et al.*, 2020; Azad, 2018). We can see a simple example in Fig. 4 where the black points are POPs and the red point is not POP. The collection of POPs for a particular problem is called the Pareto front of the problem.

In this paper, our goal is to minimize both number of vertices and misclassification rate (either global or local) of the trees at the same time and they are in conflict. The algorithm $\mathcal{A}_{\mathrm{POPs}}$ (AbouEisha *et al.*, 2019; Alsolami *et al.*, 2020; Azad, 2018) will be used to find all POPs for this BCO relative to the number of vertices and misclassification rate (either global or local).

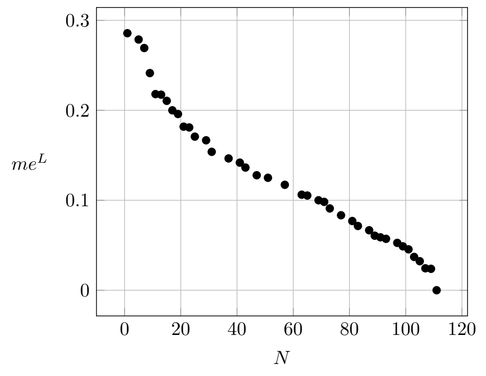
2.4 Two Algorithms for the building of BS-Trees

In this section, we describe how to build the BS-Trees.

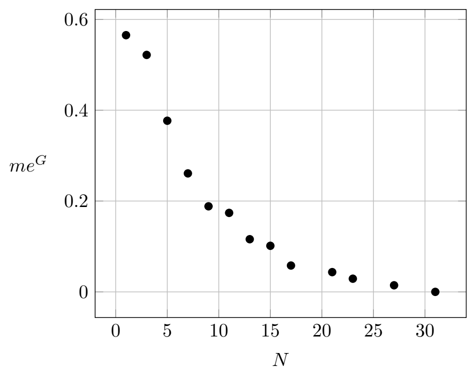
The first step is to create a directed acyclic graph (DAG) for the given decision table, vertices of which are some subtables of the considered decision table. For this purpose, we use the algorithm $\mathcal{A}_1$ from (AbouEisha *et al.*, 2019; Alsolami *et al.*, 2020; Azad, 2018; Azad *et al.*, 2015). The algorithm $\mathcal{A}_1$ works as follows: it first partitions the main table into subtables based on the all possible combination of non-constant attributes and their values. This process of partitioning will continue to the newly generated



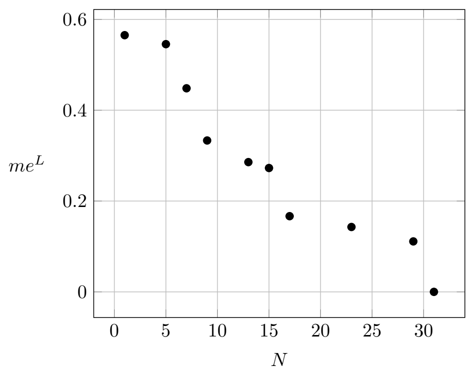
(a) S2, N and me^G



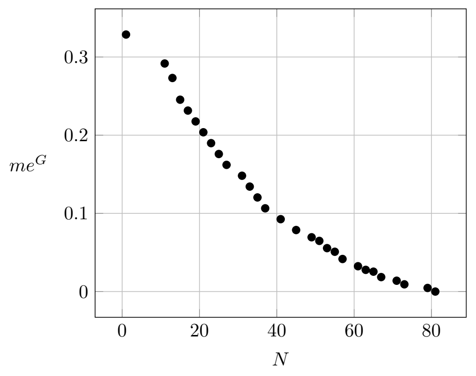
(b) S2, N and me^L



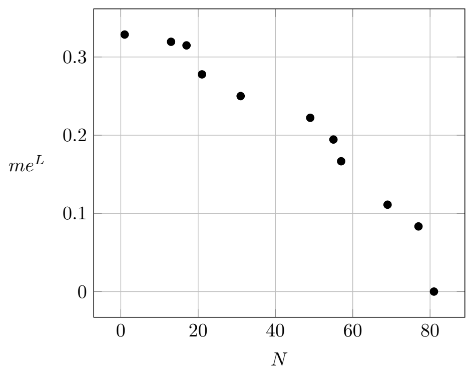
(c) S4, N and me^G



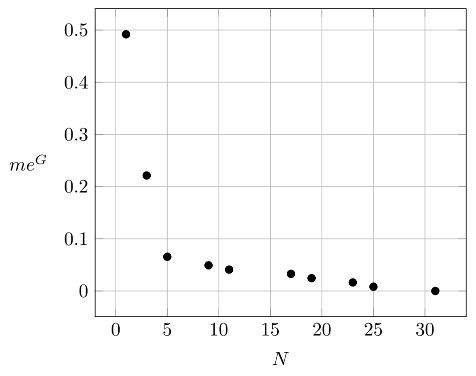
(d) S4, N and me^L



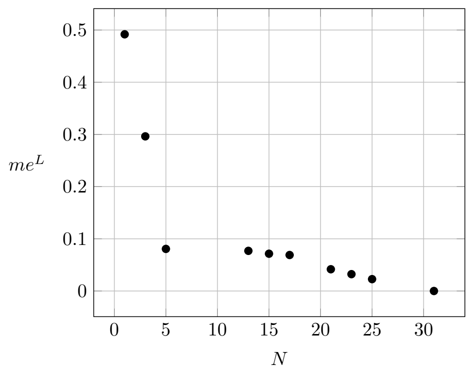
(e) S6, N and me^G



(f) S6, N and me^L



(g) S8, N and me^G



(h) S8, N and me^L

Fig. 5. Sets of Pareto optimal points for decision tables S2, S4, S6, and S8 (see Table 2) for pairs of parameters N, me^G and N, me^L

subtables until the newly generated subtables contain the same decision for each row. Also note that we generate a single instance of the subtables in the graph to maintain the property of the DAG. As an example, we can find the DAG for the decision table S_0 in Fig. 2.

As the next step, we use the algorithm $\mathcal{A}_{\mathrm{POPs}}$ from (AbouEisha *et al.*, 2019; Alsolami *et al.*, 2020; Azad, 2018; Azad *et al.*, 2015). The output of this algorithm is the Pareto front – the set of all POPs (Pareto optimal points) for BCO of BS-trees relative to two criteria for each vertex (subtable) of the created DAG. The algorithm $\mathcal{A}_{\mathrm{POPs}}$ works as follows: it traverses through smaller subtables to bigger subtables using the created DAG in a bottom up fashion. It begins with nodes that are labeled with subtables containing the same decision for each row until it reaches the root node which is labeled with the main table. At each node, it creates the set of POPs for that node based on the specific formula to combine POPs from the child nodes and remove those points which are not POPs due to this combination. This process continues and at the end, we will get the set of POPs at the root node for the main table. Next, for each POP, we can extract a tree with parameter values equal to the coordinates of this point.

In this paper, we use the algorithm $\mathcal{A}_{\mathrm{POPs}}$ to solve two BCO problems: relative to the parameters N and me^G (see examples in Fig. 5 (a), (c), (e), (g)) and relative to the parameters N and me^L (see examples in Fig. 5 (b), (d), (f), (h)).

We now describe two algorithms to design decision trees using the algorithm $\mathcal{A}_{\mathrm{POPs}}$. We assume that the DAG for the considered decision table is already created by the algorithm $\mathcal{A}_1$. Note that the time complexity of the algorithms $\mathcal{A}_1$ and $\mathcal{A}_{\mathrm{POPs}}$ is exponential in the worst case subject to the size of the decision tables. Therefore, in the worst case, the following two algorithms have exponential time complexity as well.

Algorithm 1 G_α-algorithm

```
Input: A DAG for the given decision table $S$, threshold $\alpha$
Output: Decision tree $\delta$
Build the set of POPs using $\mathcal{A}_{\mathrm{POPs}}$ algorithm to the DAG for the given table $S$ based on the parameters
  $N$ and $me^G$
$Points$ = the set of POPs build in the previous step, (by default, the points are sorted based on the first
  parameter $N$)
$P = (0, 0)$ /* Save the desired POP in variable $P$ */
for each point $(a, b)$ from $Points$ do
    if $a \le \alpha$ then
        $P = (a, b)$ // save the point which first coordinate is at most the
          threshold
    else
        break the loop // leave the loop since we found the maximum $N$ which
          is at most $\alpha$ (this is true since the points are sorted based on
          $N$)
Derive a decision tree $\delta$ from the DAG for the point $P$
Return $\delta$
```

2.4.1 G_α-algorithm

We apply the algorithm $\mathcal{A}_{\mathrm{POPs}}$ to the DAG for the decision table S. The algorithm's output is the set of POPs for the BCO of BS-trees for parameters N and me^G. We choose a POP with the maximum value of the parameter N which is at most α. After that, we extract a decision tree δ, for which the parameters N and me^G are equal to the coordinates of this POP. The tree δ is the output of G_α-algorithm. We describe the pseudo code of G_α-algorithm in Algorithm 1.

Table 2. Description of the decision tables used in the experiments

Table	Name	Rows	Attributes
$S1$	BALANCE-SCALE	625	5
$S2$	BREAST-CANCER	266	10
$S3$	CARS	1728	7
$S4$	HAYES-ROTH-DATA	69	5
$S5$	HOUSE-VOTES-84	279	17
$S6$	MONKS-2-TEST	432	6
$S7$	MONKS-2-TRAIN	169	5
$S8$	MONKS-3-TRAIN	122	6
$S9$	LYMPHOGRAPHY	148	19
$S10$	NURSERY	12960	9
$S11$	SPECT-TEST	169	23
$S12$	TIC-TAC-TOE	958	10
$S13$	ZOO-DATA	59	17

2.4.2 L_α-algorithm

This algorithm is same as G_α except that we create the set of POPs for the BCO of BS-trees for parameters N and me^L. We choose a POP with the maximum value of the parameter N which is at most α. After that, we extract a decision tree δ, for which the parameters N and me^L are equal to the coordinates of this POP. The tree δ is the output of L_α-algorithm.

3. Results of Experiments

We have used 13 decision tables from the UCI Machine Learning Repository (Dua and Graff, 2017) for the experiments (described in Table 2).

We applied G_α and L_α algorithms to each of these tables for the threshold $\alpha = 13, \ldots, 35$ with step 2 and found values of the parameters N, me^G, and me^L for the created decision trees. Table 3 describes the experimental results for the four thresholds 21, 23, 31 and 33 ('Avg' means average values of the parameters).

We can see from the results that the L_α-algorithm in comparison to G_α-algorithm gives us smaller values of the parameter me^L. Similarly, the G_α-algorithm in comparison to L_α-algorithm gives us the smaller values of the parameter me^G. Furthermore, the general pattern is that the N parameter is decreasing, whereas, me^G and me^L parameters are increasing with the decrease of the value of the threshold. Hence, we can easily adjust the number of vertices to get the desired misclassification rates using this optimization model.

We took the averages of the parameters over the 12 thresholds which are shown in Table 4. Here, N_A denotes the average of the parameter N, me^G_A denotes the average of the parameter me^G and me^L_A denotes the average of the parameter me^L. Below, we analyze the changes of the average values of the parameters for the decrease of the threshold.

We can see the changes of the average values of N, me^G and me^L with the decrease of the threshold from 35 to 13 (by step 2) for both G_α and L_α algorithms in Table 5. If we investigate the changes, we do not find any threshold which is much better than others. In general, if we decrease the threshold, the average value of the parameter N decreases and the average values of the parameters me^L and me^G increase step by step. Rather than finding a single threshold, we may consider a range of thresholds which can be enough interesting.

– Let us first consider the me^G_A parameter of the G_α-algorithm. If we decrease the threshold, initially the average value is increasing slowly and at the threshold 15, it greatly increases compared to the previous threshold 17.

– Next let us consider the me^L_A parameter of the G_α-algorithm. If we decrease the threshold, the behavior of the average value is not stable. Sometimes it is increasing then it decreases, again it is

Table 3. Different thresholds and corresponding results

Table	G_{21} algorithm			L_{21} algorithm		
	N	me^G	me^L	N	me^G	me^L
$S1$	21	0.18	0.46	21	0.20	0.31
$S2$	21	0.15	0.28	21	0.17	0.18
$S3$	21	0.09	0.39	21	0.12	0.22
$S4$	21	0.04	0.17	17	0.06	0.17
$S5$	19	0.02	0.02	19	0.02	0.02
$S6$	21	0.05	0.20	21	0.06	0.10
$S7$	21	0.20	0.33	21	0.25	0.28
$S8$	21	0.18	0.29	19	0.21	0.25
$S9$	19	0.02	0.25	21	0.02	0.04
$S10$	21	0.10	0.32	21	0.11	0.20
$S11$	21	0.01	0.06	21	0.02	0.02
$S12$	21	0.16	0.39	21	0.19	0.24
$S13$	17	0.00	0.00	17	0.00	0.0
Avg	20.38	0.09	0.24	20.08	0.11	0.16

Threshold 21

Table	G_{23} algorithm			L_{23} algorithm		
	N	me^G	me^L	N	me^G	me^L
$S1$	23.00	0.17	0.33	23.00	0.22	0.25
$S2$	23.00	0.15	0.28	23.00	0.17	0.18
$S3$	23.00	0.08	0.40	21.00	0.12	0.22
$S4$	23.00	0.03	0.17	23.00	0.04	0.14
$S5$	19.00	0.02	0.03	19.00	0.02	0.02
$S6$	23.00	0.04	0.20	23.00	0.07	0.09
$S7$	23.00	0.19	0.33	21.00	0.25	0.28
$S8$	23.00	0.16	0.32	23.00	0.21	0.24
$S9$	23.00	0.02	0.08	23.00	0.02	0.03
$S10$	23.00	0.10	0.34	21.00	0.11	0.20
$S11$	21.00	0.01	0.13	23.00	0.01	0.02
$S12$	23.00	0.15	0.39	23.00	0.19	0.23
$S13$	17.00	0.00	0.00	17.00	0.00	0.00
Avg	22.08	0.09	0.23	21.77	0.11	0.15

Threshold 23

Table	G_{31} algorithm			L_{31} algorithm		
	N	me^G	me^L	N	me^G	me^L
$S1$	31.00	0.15	0.50	23.00	0.22	0.25
$S2$	31.00	0.12	0.25	31.00	0.14	0.15
$S3$	31.00	0.06	0.34	29.00	0.14	0.21
$S4$	31.00	0.00	0.00	31.00	0.00	0.00
$S5$	31.00	0.01	0.04	27.00	0.01	0.02
$S6$	31.00	0.01	0.20	31.00	0.02	0.04
$S7$	31.00	0.15	0.33	31.00	0.22	0.25
$S8$	31.00	0.11	0.32	25.00	0.18	0.23
$S9$	31.00	0.00	0.00	31.00	0.00	0.00
$S10$	31.00	0.09	0.34	31.00	0.11	0.18
$S11$	29.00	0.00	0.00	29.00	0.00	0.00
$S12$	31.00	0.10	0.39	31.00	0.15	0.22
$S13$	17.00	0.00	0.00	17.00	0.00	0.00
Avg	29.77	0.06	0.21	28.23	0.09	0.12

Threshold 31

Table	G_{33} algorithm			L_{33} algorithm		
	N	me^G	me^L	N	me^G	me^L
$S1$	33.00	0.15	0.50	23.00	0.22	0.25
$S2$	33.00	0.12	0.33	31.00	0.14	0.15
$S3$	33.00	0.06	0.34	29.00	0.14	0.21
$S4$	31.00	0.00	0.00	31.00	0.00	0.00
$S5$	31.00	0.01	0.02	33.00	0.01	0.01
$S6$	33.00	0.01	0.08	33.00	0.01	0.03
$S7$	33.00	0.13	0.33	31.00	0.22	0.25
$S8$	33.00	0.10	0.32	33.00	0.16	0.22
$S9$	31.00	0.00	0.00	31.00	0.00	0.00
$S10$	33.00	0.08	0.29	33.00	0.10	0.18
$S11$	29.00	0.00	0.00	29.00	0.00	0.00
$S12$	33.00	0.09	0.39	33.00	0.16	0.21
$S13$	17.00	0.00	0.00	17.00	0.00	0.00
Avg	31.00	0.06	0.20	29.77	0.09	0.12

Threshold 33

Table 4. Average results for 13 decision tables

Threshold	G_α-algorithm			L_α-algorithm		
α	N_A	me_A^G	me_A^L	N_A	me_A^G	me_A^L
13	12.7500	0.1325	0.2326	12.7500	0.1457	0.1883
15	14.5000	0.1195	0.2467	14.2500	0.1357	0.1811
17	17.0000	0.1091	0.2730	16.2308	0.1267	0.1734
19	18.5385	0.1017	0.2402	17.6154	0.1222	0.1669
21	20.3846	0.0937	0.2438	20.0769	0.1106	0.1555
23	22.0769	0.0859	0.2309	21.7692	0.1093	0.1452
25	24.2308	0.0791	0.2219	23.0000	0.1037	0.1414
27	25.9231	0.0730	0.2192	23.9231	0.1005	0.1393
29	27.3077	0.0685	0.2019	25.7692	0.0970	0.1338
31	29.7692	0.0622	0.2089	28.2308	0.0915	0.1184
33	31.0000	0.0576	0.1998	29.7692	0.0887	0.1168
35	32.2308	0.0541	0.1851	30.8462	0.0762	0.1087

Table 5. Changes of the average values of the parameters for the decrease of the threshold ('-' means decrease, otherwise increase)

Threshold	G_α-algorithm			L_α-algorithm		
α	N_A	me^G_A	me^L_A	N_A	me^G_A	me^L_A
13	-1.7500	0.0130	-0.0140	-1.5000	0.0100	0.0072
15	-2.5000	0.0104	-0.0264	-1.9808	0.0091	0.0077
17	-1.5385	0.0074	0.0329	-1.3846	0.0044	0.0065
19	-1.8462	0.0080	-0.0037	-2.4615	0.0116	0.0114
21	-1.6923	0.0078	0.0129	-1.6923	0.0013	0.0103
23	-2.1538	0.0068	0.0091	-1.2308	0.0056	0.0038
25	-1.6923	0.0061	0.0027	-0.9231	0.0032	0.0021
27	-1.3846	0.0045	0.0172	-1.8462	0.0035	0.0055
29	-2.4615	0.0063	-0.0069	-2.4615	0.0056	0.0154
31	-1.2308	0.0046	0.0091	-1.5385	0.0027	0.0016
33	-1.2308	0.0035	0.0146	-1.0769	0.0125	0.0081
35	–	–	–	–	–	–

increasing then again decreases.

– Let us consider the me^G_A parameter of the L_α-algorithm. If we decrease the threshold, the average value in general is increasing and at the threshold 19, it seriously increases compared to the threshold 21.

– Let us consider the me^L_A parameter of the L_α-algorithm. If we decrease the threshold, the average value in general increasing and at the thresholds 21 and 19 it seriously increases from the threshold 23.

The average values really depend on the nature of the considered decision tables as well as the size of the tables. Therefore it may be difficult to find a single threshold which can be considered as optimal one. But based on the above analysis, we may recommend the range 15 to 21 of thresholds that can be useful for building of decision trees for knowledge representation.

4. Conclusions

The goal of this paper is to understand which threshold of number of nodes should be chosen if we would like to minimize the number of nodes and misclassification of the decision trees at the same time. In this way, it is possible to obtain an understandable decision tree. For this, we designed two algorithms G_α and L_α to build binary split decision trees (BS-trees) having at most α vertices (where α varies between 13 to 35 with step 2). These algorithms are based on the bi-criteria optimization of N and misclassification rate (either me^G or me^L). We found an optimization model to adjust the parameters of decision trees and recommend a range of threshold 15 to 21 which can give us acceptable values of the parameters me^G and me^L. In the future, it is possible to extend these algorithms to multi-label decision tables using BCO as explained in (Alsolami *et al.*, 2020; Azad, 2018). Furthermore, we are also trying to invent new algorithms to restrict the number of branches in the DAG to reduce the time complexity of the algorithms. Moreover, in future, we are planning to extend bi-criteria optimization to the case of decision trees with hypothesis (Azad *et al.*, 2021*a*,*b*,*c*,*d*).

ACKNOWLEDGMENTS

We are greatly indebted to Jouf University and King Abdullah University of Science & Technology for the support of this research.

References

AbouEisha, H., Amin, T., Chikalov, I., Hussain, S. and Moshkov, M. (2019), *Extensions of Dynamic Programming for Combinatorial Optimization and Data Mining*, Vol. 146 of *Intelligent Systems Reference Library*, Springer.

Aldilaijan, A., Azad, M. and Moshkov, M. (2019), 'Experimental study of totally optimal decision trees', *Fundam. Inform.* **165**(3-4), 245–261. **URL:** *https://doi.org/10.3233/FI-2019-1784*

Almuallim, H., Kaneda, S. and Akiba, Y. (2002), 3 - development and applications of decision trees, *in* C. T. Leondes, ed., 'Expert Systems', Academic Press, Burlington, pp. 53–77.

Alsolami, F., Azad, M., Chikalov, I. and Moshkov, M. (2020), *Decision and Inhibitory Trees and Rules for Decision Tables with Many-valued Decisions*, Vol. 156 of *Intelligent Systems Reference Library*, Springer.

Azad, M. (2018), Decision and Inhibitory Trees for Decision Tables with Many-Valued Decisions, PhD thesis, King Abdullah University of Science & Technology, Thuwal, Saudi Arabia.
URL: *http://hdl.handle.net/10754/628023*

Azad, M. (2020*a*), 'Decision trees with at most 19 vertices for knowledge representation', *Trans. Rough Sets* **22**, 1–7.
URL: *https://doi.org/10.1007/978-3-662-62798-3 1 _*

Azad, M. (2020*b*), 'Knowledge representation using decision trees constructed based on binary splits', *KSII Transactions on Internet and Information Systems* **14**(10), 4007–4024.

Azad, M., Chikalov, I., Hussain, S. and Moshkov, M. (2015), Multi-pruning of decision trees for knowledge representation and classification, *in* '3rd IAPR Asian Conference on Pattern Recognition, ACPR 2015, Kuala Lumpur, Malaysia, November 3-6, 2015', IEEE, pp. 604–608.

Azad, M., Chikalov, I., Hussain, S. and Moshkov, M. (2018), Restricted multi-pruning of decision trees, *in* '13th International FLINS Conference on Data Science and Knowledge Engineering for Sensing Decision Support, FLINS', pp. 371–378.

Azad, M., Chikalov, I., Hussain, S. and Moshkov, M. (2021*a*), 'Entropy-based greedy algorithm for decision trees using hypotheses', *Entropy* **23**(7).
URL: *https://www.mdpi.com/1099-4300/23/7/808*

Azad, M., Chikalov, I., Hussain, S. and Moshkov, M. (2021*b*), Minimizing depth of decision trees with hypotheses (to appear), *in* 'International Joint Conference on Rough Sets (IJCRS 2021), 19–24 September 2021, Bratislava, Slovakia'.

Azad, M., Chikalov, I., Hussain, S. and Moshkov, M. (2021*c*), Minimizing number of nodes in decision trees with hypotheses (to appear), *in* '25th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2021), 8–10 September 2021, Szczecin, Poland'.

Azad, M., Chikalov, I., Hussain, S. and Moshkov, M. (2021*d*), 'Optimization of decision trees with hypotheses for knowledge representation', *Electronics* **10**(13).
URL: *https://www.mdpi.com/2079-9292/10/13/1580*

Azad, M., Chikalov, I. and Moshkov, M. (2019), Decision trees for knowledge representation, *in* K. Ropiak, L. Polkowski and P. Artiemjew, eds, '28th International Workshop on Concurrency, Specification and Programming, CS&P 2019, Olsztyn, Poland, September 24–26, 2019', Vol. 2571 of *CEUR Workshop Proceedings*, CEUR-WS.org.
URL: *http://ceur-ws.org/Vol-2571/CSP2019 paper 1.pdf*

Azad, M., Chikalov, I., Moshkov, M. and Zielosko, B. (2013), 'A greedy algorithm for construction of decision trees for tables with many-valued decisions - A comparative study', *Fundam. Inform.* **128**(1-2), 1–15.
URL: *https://doi.org/10.3233/FI-2013-929*

Azad, M. and Moshkov, M. (2017), ‘Multi-stage optimization of decision and inhibitory trees for decision tables with many-valued decisions’, *Eur. J. Oper. Res.* **263**(3), 910–921.
URL: *https://doi.org/10.1016/j.ejor.2017.06.026*

Bellman, R. (1954), ‘The theory of dynamic programming’, *Bull. Amer. Math. Soc.* **60**(6), 503–515.

Boryczka, U. and Kozak, J. (2009), New algorithms for generation decision trees-ant-miner and its modifications, *in* A. Abraham, A. E. Hassanien, A. C. P. de Leon Ferreira de Carvalho and V. Snásel, eds, ‘Foundations of Computational Intelligence - Volume 6: Data Mining’, Vol. 206 of *Studies in Computational Intelligence*, Springer, pp. 229–262.

Breiman, L., Friedman, J. H., Olshen, R. A. and Stone, C. J. (1984), *Classification and Regression Trees*, Wadsworth and Brooks, Monterey, CA.

Chai, B.-B., Zhuang, X., Zhao, Y. and Sklansky, J. (1996), Binary linear decision tree with genetic algorithm, *in* ‘Proceedings - International Conference on Pattern Recognition’, Vol. 4, pp. 530–534.

Chang, K.-H. (2015), Chapter 5 - multiobjective optimization and advanced topics, *in* K.-H. Chang, ed., ‘Design Theory and Methods Using CAD/CAE’, Academic Press, Boston, pp. 325–406.

de Weck, O. (2004), ‘Multiobjective optimization: History and promise’, *Invited Keynote Paper, GL2-2, the Third China-Japan-Korea Joint Symposium on Optimization of Structural and Mechanical Systems* **2**.

Dua, D. and Graff, C. (2017), ‘UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences’.
URL: *http://archive.ics.uci.edu/ml*

Ehrgott, M. and Gandibleux, X. (2000), ‘A survey and annotated bibliography of multiobjective combinatorial optimization’, *OR Spectr.* **22**(4), 425–460.

Emmerich Michael T. M., D. A. H. (2018), ‘A tutorial on multiobjective optimization: fundamentals and evolutionary methods’, *Natural Computing* **17**(3), 1585–609.

Garey, M. R. (1972), ‘Optimal binary identification procedures’, *SIAM Journal on Applied Mathematics* **23**, 173–186.

Geiger, M. J. and Sevaux, M. (2011), The biobjective inventory routing problem – problem solution and decision support, *in* J. Pahl, T. Reiners and S. Voß, eds, ‘Network Optimization’, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 365–378.

Heath, D., Kasif, S. and Salzberg, S. (1993), ‘Induction of oblique decision trees’, *Journal of Artificial Intelligence Research* **2**(2), 1–32.

Hopfe, C., Emmerich, M., Marijt, R. and Hensen, J. (2012), Robust multi-criteria design optimization in building design, *in* ‘Proceedings of Building Simulation and Optimization’, pp. 118–125.

Kouvelis, P. and Sayın, S. (2006), ‘Algorithm robust for the bicriteria discrete optimization problem’, *Ann. Oper. Res.* **147**(1), 71–85.

Mehmood, Z., Gul, N., Altaf, M., Mahmood, T., Saba, T., Rehman, A. and Mahmood, M. (2018), ‘Scene search based on the adapted triangular regions and soft clustering to improve the effectiveness of the visual-bag-of-words model’, *EURASIP Journal on Image and Video Processing* **2018**.

Moshkov, M. (2005), Time complexity of decision trees, *in* J. F. Peters and A. Skowron, eds, ‘Trans. Rough Sets III’, Vol. 3400 of *Lecture Notes in Computer Science*, Springer, pp. 244–459.

Podgorelec, V., Kokol, P., Stiglic, B. and Rozman, I. (2002), 'Decision trees: An overview and their use in medicine', *Journal of medical systems* **26**, 445–63.

Quinlan, J. (1986), 'Induction of decision trees', *Mach Learn* **1**, 81–106.

Quinlan, J. R. (2014), *C4. 5: programs for machine learning*, Elsevier.

Rokach, L. and Maimon, O. (2008), *Data Mining with Decision Trees: Theory and Applications*, World Scientific Publishing, River Edge, NJ.

Rosenthal, S. and Borschbach, M. (2017), Design perspectives of an evolutionary process for multi-objective molecular optimization, *in* '9th International Conference on Evolutionary Multi-Criterion Optimization - Volume 10173', EMO 2017, Springer-Verlag, Berlin, Heidelberg, pp. 529–544.

Sarwar, A., Mehmood, Z., Saba, T., Qazi, K., Adnan, A. and Jamal, H. (2018), 'A novel method for content-based image retrieval to improve the effectiveness of the bag-of-words model using a support vector machine', *Journal of Information Science* **45**.

Schumacher, H. and Sevcik, K. C. (1976), 'The synthetic approach to decision table conversion', *Commun. ACM* **19**(6), 343–351.

Tian, J., Cheng, J. and Gong, Y. (2018), 'Optimization of the operating cost of sewage conveyance', *Kuwait Journal of Science* **45**(2), 104–115.

van der Horst, E., Marques-Gallego, P., Mulder-Krieger, T., Veldhoven, J., Kruisselbrink, J., Aleman, A., Emmerich, M., Brussee, J., Bender, A. and Ijzerman, A. (2012), 'Multi-objective evolutionary design of adenosine receptor ligands', *Journal of Chemical Information and Modeling* **52**(7), 1713–1721.

Wang, P., Emmerich, M., Li, R., Tang, K., Back, T. and Yao, X. (2015), 'Convex hull-based multi-objective genetic programming for maximizing receiver operating characteristic performance', *IEEE Transactions on Evolutionary Computation* **19**(2), 188–200.

Yahyaoui, H., El-Qurna, J. and Almulla, M. (2020), 'Specification and recognition of service trust behaviors', *Kuwait Journal of Science* **47**(1), 33–41.

Yevseyeva, I., Basto-Fernandes, V., Ruano-Ordas, D. and Mendez, J. R. (2013), 'Optimising anti-spam filters with evolutionary algorithms', *Expert Systems with Applications* **40**(10), 4010 – 4021.

Submitted	: 07/11/2020
Revised	: 10/07/2021
Accepted	: 10/07/2021
DOI	: 10.48129/kjs.10725