Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 22 (2013) 359 – 365

Procedia Computer Science

17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems - KES2013

Totally Optimal Decision Trees for Monotone Boolean Functions with at most Five Variables

Igor Chikalov, Shahid Hussain, Mikhail Moshkov

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

Abstract

In this paper, we present the empirical results for relationships between time (depth) and space (number of nodes) complexity of decision trees computing monotone Boolean functions, with at most five variables. We use Dagger (a tool for optimization of decision trees and decision rules) to conduct experiments. We show that, for each monotone Boolean function with at most five variables, there exists a totally optimal decision tree which is optimal with respect to both depth and number of nodes.

Keywords: Totally optimal decision trees; monotone Boolean functions; number of nodes and depth of decision trees.

1. Introduction

Decision trees can be used as classifiers, a way for representing knowledge, and also as algorithms for solving different problems (see for example [1]). These different uses require optimizing decision trees for different criteria. For this purpose, we have created a software system for decision trees (as well as decision rules) called dagger—a tool based on dynamic programming which allows us to optimize decision trees (and decision rules) relative to various cost functions such as depth (length), average depth (average length), total number of nodes, and number of misclassifications sequentially [2, 3, 4, 5].

Often, during experiments with Dagger, on data from UCI ML Repository [6], we get totally optimal decision trees – simultaneously optimal relative to the depth and number of number of nodes. For example, in [7] we show that breast-cancer and house-vote datasets have totally optimal trees while there does not exist such totally optimal decision trees for the dataset lymphography. These totally optimal decision trees can be useful from the points of view of knowledge representation and efficiency of algorithms.

Studying relationship between time and space complexity of algorithms is an important topic of computational complexity theory. These relationships are considered often for non-universal computational models such as branching programs and decision trees [8, 9], where time and space complexity is independent of the length of input. The considered relationships become trivial if there exist totally optimal algorithms i.e., optimal with

E-mail address: {igor.chikalov, shahid.hussain, mikhail.moshkov}@kaust.edu.sa.

doi:10.1016/j.procs.2013.09.113

f_1	$\cdots$	f_m	d
b_{11}	$\cdots$	b_{1m}	c_1
$\vdots$	$\ddots$	$\vdots$	$\vdots$
b_{N1}	$\cdots$	b_{Nm}	c_N

Fig. 1. Decision table T with m attributes and N rows

respect both time and space complexity. To understand the phenomenon of existence of totally optimal decision trees (whether it is usual or rare), we studied monotone Boolean functions.

In this paper, we study decision trees for computation of monotone Boolean functions with n variables, $0 \leq n \leq 5$. We consider the depth and the number of nodes of decision tree as time and space complexity, respectively. For each monotone Boolean function with at most five variables, we study relationship between depth and number of nodes in decision trees computing this function. As a result, we obtain that, for each monotone Boolean function with at most five variables, there exists a totally optimal decision tree i.e., a decision tree with both minimum depth and minimum number of nodes.

This paper is organized into five sections including the Introduction. Section 2 presents some important basic notions related to decision tables/trees, cost functions, and representation of sets of decision trees for a given decision table. Section 3 describes in detail the procedure of optimization for decision trees. Main result of this paper goes into Section 4, including the plots for totally optimal decision trees for monotone Boolean functions. Section 5, concludes the paper followed by references.

2. Basic Notions

In the following, we consider notions of decision tables and decision trees in general case. Later, we will discuss the corresponding notions for monotone Boolean functions.

2.1. Decision Tables and Trees

A *decision table* is a rectangular array of values, arranged in rows and columns. The columns of a decision table are labeled with conditional attributes and rows of the table contain values of corresponding attributes. In this chapter, we consider only decision tables with discrete attributes. These tables contain neither missing values nor equal rows. Consider a decision table T depicted in Fig 1. Here $f_1, \ldots, f_m$, are names of columns (conditional attributes); $c_1, \ldots, c_N$, nonnegative integers, which are interpreted as decisions (values of the decision attribute d); b_{ij} are nonnegative integers which are interpreted as values of conditional attributes. We assume that all rows are pairwise different. We denote by $E(T)$ the set of attributes (columns of T). For $f_i \in E(T)$, we say $E(T, f_i)$ is the set of values for the column f_i.

Let $f_{i_1}, \ldots, f_{i_t} \in E(T)$ form a subset of t attributes from T and let $a_1, \ldots, a_t$ be their corresponding values, then we denote by $T(f_i, a_1) \ldots (f_i, a_t)$, the subtable of the table T, which consists of only the rows (of T) that are at the intersection of columns $f_{i_1}, \ldots, f_{i_t}$, have values $a_1, \ldots, a_t$, respectively. Such nonempty tables (including the table T) are called *separable subtables* of the table T. For a subtable Θ of the table T, we denote $R(\Theta)$, the number of unordered pairs of rows that are labeled with different decisions.

A *decision tree* Γ *over the table* T is a finite directed rooted tree in which each terminal node is labeled with a decision. Each nonterminal node is labeled with a conditional attribute, and for each nonterminal node the outgoing edges are labeled with pairwise different nonnegative integers. For each node v of Γ, we associate a subtable $T(v)$ of the table T. If v is the root node then $T(v) = T$. For every other node v of Γ, $T(v) = T(f_{i_1}, a_1) \ldots (f_{i_t}, a_t)$, where $f_{i_1}, \ldots, f_{i_t}$ are the attributes attached to the nodes in path from the root to v and $a_1, \ldots, a_t$ are values of these attributes that are attached to the edges in this path.

We say that a tree Γ is *a decision tree for* T if for any node v of Γ following conditions are satisfied:

- If $R(T(v)) = 0$ then, v is a terminal nodes, labeled with the common decision for $T(v)$,

- Otherwise, v is labeled with an attribute $f_i \in E(T(v))$, and if $E(T(v), f_i) = \{a_1, \ldots, a_t\}$ then, t outgoing edges labeled with $a_1, \ldots, a_t$, respectively, from the node v.

Let Γ be a decision tree from T. For any row r of T, there exists exactly one terminal node v of Γ such that r belongs to the table $T(v)$. Let b be labeled with the decision b. We say that b is the *result of the work of decision tree* Γ *on* r.

2.2. Cost Functions

We will consider cost functions which are given in the following way: values of the considered cost function ψ, which are nonnegative numbers, are defined by induction on pairs (T, Γ), where T is a decision table and Γ is a decision tree for T or an α-decision tree for T. Let Γ be a decision tree represented in Fig 2. Then $\psi(T, \Gamma) = \psi^0(T, b)$ where ψ^0 is an operator which transforms a decision table T and a nonnegative integer b into a nonnegative number. Let Γ be a decision tree depicted in Fig 3. Then

$$\psi(T, \Gamma) = F(N(T), \psi(T(f_i, a_1), \Gamma_1), \ldots, \psi(T(f_i, a_t), \Gamma_t)).$$

Here $N(T)$ is the number of rows in the table T, and $F(n, \psi_1, \psi_2, \ldots)$ is an operator which transforms the considered tuple of nonnegative numbers into a nonnegative number. Note that the number of variables $\psi_1, \psi_2, \ldots$ is not bounded from above.

The considered cost function will be called *monotone* if for any natural t and any nonnegative numbers $a, c_1, \ldots, c_t, d_1, \ldots, d_t$, from inequalities $c_1 \leq d_1, \ldots, c_t \leq d_t$ the inequality $F(a, c_1, \ldots, c_t) \leq F(a, d_1, \ldots, d_t)$ follows.

The considered cost function will be called *strongly monotone* if it is monotone and for any natural t and any nonnegative numbers $a, c_1, \ldots, c_t, d_1, \ldots, d_t$ from inequalities $a > 0$, $c_1 \leq d_1, \ldots, c_t \leq d_t$ and inequality $c_i < d_i$, which is true for some $i \in \{1, \ldots, t\}$, the inequality $F(a, c_1, \ldots, c_t) < F(a, d_1, \ldots, d_t)$ follows.

For an arbitrary row r of the decision table T, we denote by $l(r)$, *the length of path* from the root to the terminal node v of T such that r is in $T(v)$. We say that the *depth*, represented as h, is the maximum of all path lengths for all rows in T. That is,

$$h(T, \Gamma) = \max_r \{l(r)\},$$

where max is taken on all rows r of the table T. We will drop T from $h(T, \Gamma)$ when it is obvious from the context. That is, we will write $h(\Gamma)$ instead if, T is known.

For a decision tree Γ of a decision table T, we represent the total number of nodes Γ by $L(\Gamma)$. It is interesting to note that depth and number of nodes of a decision tree are bounded above by values depending upon the size of the table. That is m and $2N - 1$ are the upper bounds for h and L for a decision table with m conditional attributes and N rows. Furthermore, depth of a decision tree is a monotone cost function and number of nodes for a decision tree is strongly monotone cost function.

2.3. Representation of Sets of Decision Trees

Consider an algorithm for construction of a graph $\Delta(T)$, which represents the set of all decision trees for the table T. Nodes of this graph are some separable subtables of the table T. During each step we process one node and mark it with the symbol *. We start with the graph that consists of one node T and finish when all nodes of the graph are processed.

Let the algorithm has already performed p steps. We now describe the step number $(p + 1)$. If all nodes are processed then the work of the algorithm is finished, and the resulting graph is $\Delta(T)$. Otherwise, choose a node (table) Θ that has not been processed yet. If $R(\Theta) = 0$, label the considered node with the *common decision* b for Θ, mark it with symbol * and proceed to the step number $(p + 2)$. If $R(\Theta) > 0$, then for each $f_i \in E(\Theta)$ draw a bundle of edges from the node Θ (this bundle of edges will be called f_i-*bundle*). Let $E(\Theta, f_i) = \{a_1, \ldots, a_t\}$. Then draw t edges from Θ and label these edges with pairs $(f_i, a_1), \ldots, (f_i, a_t)$ respectively. These edges enter into nodes $\Theta(f_i, a_1), \ldots, \Theta(f_i, a_t)$. If some of the nodes $\Theta(f_i, a_1), \ldots, \Theta(f_i, a_t)$ are not present in the graph then add these nodes to the graph. Mark the node Θ with the symbol * and proceed to the step number $(p + 2)$. Now for

Fig. 2. Trivial DT

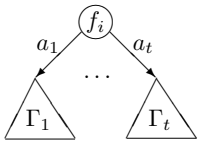

Fig. 3. Aggregated DT

each node Θ of the graph $\Delta(T)$, we describe the set of decision trees corresponding to the node Θ. We will move from terminal nodes, which are labeled with numbers, to the node T. Let Θ be a node, which is labeled with a number b. Then the only trivial decision tree depicted in Fig 2 corresponds to the node Θ.

Let Θ be a nonterminal node (table) then there is a number of bundles of edges starting in Θ. We consider an arbitrary bundle and describe the set of decision trees corresponding to this bundle. Let the considered bundle be an f_i-bundle where $f_i \in (\Theta)$ and $E(\Theta, f_i) = \{a_1, \ldots, a_t\}$. Let $\Gamma_1, \ldots, \Gamma_t$ be decision trees from sets corresponding to the nodes $\Theta(f_i, a_1), \ldots, \Theta(f_i, a_t)$. Then the decision tree depicted in Fig 3 belongs to the set of decision trees, which correspond to this bundle. All such decision trees belong to the considered set, and this set does not contain any other decision trees. Then the set of decision trees corresponding to the node Θ coincides with the union of sets of decision trees corresponding to the bundles starting in Θ. We denote by $D(\Theta)$ the set of decision trees corresponding to the node Θ.

The following proposition shows that the graph $\Delta(T)$ can represent all decision trees for the table T.

Proposition 1. *Let T be a decision table and Θ a node in the graph $\Delta(T)$. Then the set $D(\Theta)$ coincides with the set of all decision trees for the table Θ.*

Proof. We prove this proposition by induction on nodes in the graph $\Delta(T)$. For each terminal node Θ, only one decision tree exists as depicted in Fig 2, and the set $D(T)$ contains only this tree. Let Θ be a nonterminal node and the statement of proposition hold for all its descendants.

Consider an arbitrary decision tree $\Gamma \in D(\Theta)$. Obviously, Γ contains more than one node. Let the root of Γ be labeled with an attribute f_i and the edges leaving root be labeled with the numbers $a_1, \ldots, a_t$. For $j = 1, \ldots, t$, denote by Γ_j the decision tree connected to the root with the edge labeled with the number a_j. From the definition of the set $D(\Theta)$ it follows that f_i is contained in the set $E(\Theta)$, $E(\Theta, f_i) = \{a_1, \ldots, a_t\}$ and for $j = 1, \ldots, t$, the decision tree Γ_j belongs to the set $D(\Theta(f_i, a_j))$. According to the inductive hypothesis, the tree Γ_j is a decision tree for the table $\Theta(f_i, a_j)$. Then the tree Γ is a decision tree for the table Θ.

Now we consider an arbitrary decision tree Γ for the table Θ. According to the definition, the root of Γ is labeled with an attribute f_i from the set $E(\Theta)$, edges leaving the root are labeled with numbers from the set $E(\Theta, f_i)$ and the subtrees whose roots are nodes, to which these edges enter, are decision trees for corresponding descendants of the node Θ. Then, according to the definition of the set $D(\Theta)$ and to inductive hypothesis, the tree Γ belongs to the set $D(\Theta)$. □

2.4. Monotone Boolean Functions

A monotone Boolean function is a Boolean function $f(x_1, \ldots, x_n)$ satisfying the following condition: for any $(a_1, \ldots, a_n), (b_1, \ldots, b_n) \in \{0, 1\}^n$, if $a_1 \leq b_1, \ldots, a_n \leq b_n$ then $f(a_1, \ldots, a_n) \leq f(b_1, \ldots, b_n)$. The number of monotone Boolean functions, $M(n)$, for n variables, $0 \leq n \leq 5$, is shown in Table 1 (see [10]).

A decision table T_f for a Boolean function $f(x_1, \ldots, x_n)$ consists of $n + 1$ columns, where first n columns are labeled with variables $x_1, \ldots, x_n$ and $(n + 1)$-st column is the decision attribute. There are 2^n rows in such tables, which are all n-tuples with values from $\{0, 1\}$. Each row $(a_1, \ldots, a_n)$ is labeled with the value $f(a_1, \ldots, a_n)$ as the decision.

We use a binary decision tree Γ to compute a Boolean function of n variable $f : \{x_1, \ldots, x_n\} \to \{0, 1\}$, such that the terminal nodes of Γ are labeled with either 0 or 1 and every nonterminal node is labeled with a variable x_i from the domain of f. Each nonterminal node has two outgoing edges labeled with 0 and 1, respectively. We consider the depth of a tree – the maximum length of a path from the root to a terminal node – as time complexity and the number of nodes as space complexity.

Table 1. Number $M(n)$ of monotone Boolean functions with n, $0 \le n \le 5$, variables

n	0	1	2	3	4	5
$M(n)$	2	3	6	20	168	7581

3. Optimization of Decision Trees

In the following we describe procedure of optimization of decision trees. We also discuss possibility of sequential optimization.

3.1. Bundle-Preserved Subgraph of the Graph $\Delta(T)$

We say that a proper subgraph of the graph $\Delta(T)$ is a *bundle-preserved subgraph of* $\Delta(T)$ if we can remove all but one bundles of outgoing edges for each node in $\Delta(T)$ at the same time preserving the number of nodes in the graph. Let such a subgraph be called G, then it is clear that all terminal nodes of G are terminal nodes of $\Delta(T)$. Furthermore, we can associate a set of decision trees to each node Θ of G and these decision trees belong to the set $D(\Theta)$. We denote this set of decision trees by $D_G(\Theta)$.

3.2. Procedure of Optimization

Let G be a bundle-preserved subgraph of the graph $\Delta(T)$, and ψ be a cost function given by operators ψ^0 and F. Below we describe a procedure, which transforms the graph G into a bundle-preserved subgraph G_ψ of G. We begin from terminal nodes and move to the node T. We attach a number to each node, and possible remove some bundles of edges, which start in the considered node. Let Θ be a terminal node labeled with a number b. We attach the number $\psi^0(T, b)$ to the node Θ. Consider a node Θ, which is not terminal, and a bundle of edges, which starts in this node. Let edges be labeled with pairs $(f_i, a_1), \ldots, (f_i, a_t)$, and edges enter to nodes $\Theta(f_i, a_1), \ldots, \Theta(f_i, a_t)$, to which numbers $\psi_1, \ldots, \psi_t$ are attached already. Then we attach to the considered bundle the number $F(N(\Theta), \psi_1, \ldots, \psi_t)$.

Among numbers attached to bundles starting in Θ we choose the minimum number p and attach it to the node Θ. We remove all bundles starting in Θ to which numbers are attached that are greater than p. When all nodes will be treated we obtain a graph. Denote this graph by G_ψ. As it was done previously, for any node Θ of G_ψ we denote by $D_{G_\psi}(\Theta)$ the set of decision trees associated with Θ.

Note that using the graph G_ψ it is easy to find the number of decision trees in the set $D_{G_\psi}(\Theta)$: $|D_{G_\psi}(\Theta)| = 1$ if Θ is a terminal node. Consider a node Θ, which is not terminal, and a bundle of edges, which start in this node and enter the nodes $\Theta_1 = \Theta(f_i, a_1), \ldots, \Theta_t = \Theta(f_i, a_t)$. We correspond to this bundle the number $|D_{G_\psi}(\Theta_1)| \times \cdots \times |D_{G_\psi}(\Theta_t)|$. Then $|D_{G_\psi}(\Theta)|$ is equal to the sum of numbers corresponding to bundles starting in Θ.

Let T be a decision table and ψ a monotone cost function. Let G be a bundle-preserved subgraph of $\Delta(T)$ and Θ an arbitrary node in G. We will denote by $D^{opt}_{\psi,G}(\Theta)$ the subset of $D_G(\Theta)$ containing all decision trees having minimum complexity relative to ψ, i.e., $D^{opt}_{\psi,G} = \{\hat{\Gamma} \in D_G(\Theta) : \psi(\Theta, \hat{\Gamma}) = \min_{\Gamma \in D_G(\Theta)} \psi(\Theta, \Gamma)\}$.

Following are two theorems (prefaced by a lemma) describe important properties of the set $D_{G_\psi}(\Theta)$ for the cases of monotone and strongly monotone cost function. (Detailed discussion and proofs for these results appear in [4].)

Lemma 1 ([4]). *Let T be a decision table and ψ be a monotone cost function defined by the pair of operators (ψ^0, F). Let G be a bundle-preserved subgraph of $\Delta(T)$, Θ be an arbitrary node in the graph G and p be a number assigned to the node Θ by optimization procedure. Then for each decision tree Γ from the set $D_{G_\psi}(\Theta)$, the equality $\psi(\Theta, \Gamma) = p$ holds.*

Theorem 1 ([4]). *Let T be a decision table and ψ a monotone cost function. Let G be a bundle-preserved subgraph of $\Delta_\alpha(T)$ and Θ an arbitrary node in the graph G. Then $D_{\alpha,G_\psi}(\Theta) \subseteq D^{opt}_{\alpha,\psi,G}(\Theta)$.*

Theorem 2 ([4]). *Let T be a decision table and ψ a strongly monotone cost function. Let G be a bundle-preserved subgraph of $\Delta(T)$ and Θ be an arbitrary node in the graph G. Then $D_{G_\psi}(\Theta) = D^{opt}_{\psi,G}(\Theta)$.*

3.3. Procedure of Sequential Optimization

Let the graph $\Delta(T)$ be constructed for a decision table T. Let ψ_1, ψ_2 be strongly monotone cost functions. Apply the procedure of optimization to the graph $\Delta(T)$ and to the cost function ψ_1. As a result we obtain a bundle-preserved subgraph $(\Delta(T))_{\psi_1}$ of the graph $\Delta(T)$. Denote this subgraph by G_1. According to Proposition 1 and Theorem 2, the set of decision trees corresponding to the node T of this graph coincides with the set of all decision trees for the table T, which have minimum cost relative to ψ_1. Denote this set by D_1.

Apply the procedure of optimization to the graph G_1 and the cost function ψ_2. As a result we obtain the bundle-preserved subgraph $(G_1)_{\psi_2}$ of the graph $\Delta(T)$. Denote this subgraph by G_2. The set of decision trees corresponding to the node T of this graph coincides with the set of all decision trees from D_1 which have minimum cost relative to ψ_2. It is possible to continue this process of consecutive optimization relative to various criteria.

If ψ_2 is a monotone cost function then according to Theorem 1 the set of decision trees, corresponding to the node T of the graph G_2, is a subset of the set of all decision trees from D_1 which have minimum complexity relative to ψ_2.

4. Results

We say that a decision tree Γ for computing a Boolean function f is totally optimal if and only if it has simultaneously the minimum depth and the minimum number of nodes among all decision trees computing the function f. For example a totally optimal decision tree computing the function $x \vee y$ is depicted in Fig 4.

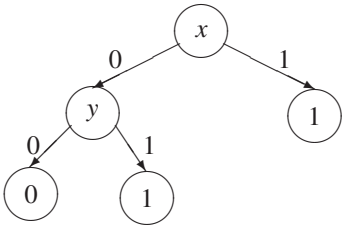

Fig. 4. Totally optimal decision tree for the function $x \vee y$ with depth 2 and 5 nodes

The main result of the paper is the following theorem.

Theorem 3. *For each monotone Boolean function with at most five variables, there exists a totally optimal decision tree computing this function.*

The proof for this theorem follows as: for any monotone Boolean function f with at most five variables we construct the decision table T_f and optimize this table in two different ways: a) optimize relative to depth, b) optimize relative to the number of nodes and depth (in sequence). From Theorems 1 and 2 and from the fact that depth is a monotone cost function and number of nodes is strongly monotone cost function, it follows that a totally optimal tree for f exists if and only if the depth of decision trees obtained by procedure of optimizations in a) and in b) is the same.

For each monotone Boolean function with n variables, $0 \leq n \leq 5$, we also measured the depth and the number of nodes of totally optimal decision tree computing this function. The sets of corresponding pairs (depth, number of nodes) are represented in Fig 5 for $n = 0, 1, 2, 3, 4$, and $n = 6$.

All computations were done using our software system Dagger [5] for optimization and analysis of decision trees and rules. This system is based on extensions of dynamic programming and allows to make sequential optimization of decision trees relative to different cost functions [4] and to study relationships between pairs of cost functions [7]. The work of Dagger consists of construction and transformations of a directed acyclic graph such that nodes of this graph are subproblems of the initial problem. In the case of monotone Boolean function f, the set of nodes is a set of subfunctions of the function f obtained by substituting some of the variables with constants 0 and 1.

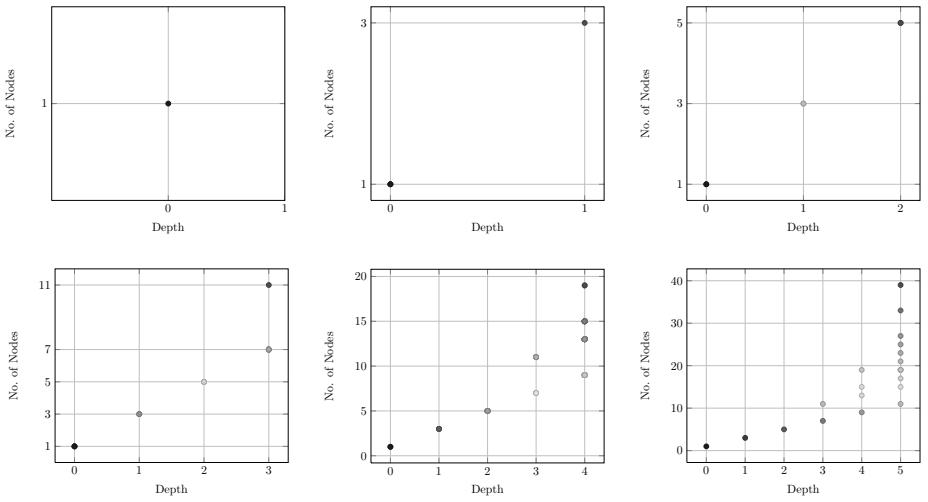

Fig. 5. Depth and number of nodes for totally optimal decision trees computing monotone Boolean functions with 0, 1, 2, 3, 4, and 5 variables

5. Conclusions

We defined and studied the notion of totally optimal decision tree for monotone Boolean functions with at most five variables. Later we are planning to study totally optimal decision trees for monotone Boolean functions with more than five variables to prove or disprove the hypothesis that, for each monotone Boolean function, there exists a totally optimal decision tree computing this function.

Acknowledgments

The authors wish to express their gratitude to anonymous reviewers for their useful comments and suggestions, which greatly improved the overall quality of the paper.

References

[1] A. Alkhalid, I. Chikalov, M. Moshkov, Constructing an optimal decision tree for fast corner point detection, in: RSKT, 2011, pp. 187–194.
[2] A. Alkhalid, I. Chikalov, M. Moshkov, On algorithm for building of optimal α-decision trees, in: M. S. Szczuka, M. Kryszkiewicz, S. Ramanna, R. Jensen, Q. Hu (Eds.), RSCTC, Springer, Heidelberg, 2010, pp. 438–445.
[3] A. Alkhalid, I. Chikalov, M. Moshkov, A tool for study of optimal decision trees, in: J. Yu, S. Greco, P. Lingras, G. Wang, A. Skowron (Eds.), RSKT, Vol. LNCS 6401, Springer, 2010, pp. 353–360.
[4] A. Alkhalid, I. Chikalov, S. Hussain, M. Moshkov, Extensions of dynamic programming as a new tool for decision tree optimization, in: S. Ramanna, L. C. Jain, R. J. Howlett (Eds.), Emerging Paradigms in Machine Learning, Vol. 13 of Smart Innovation, Systems and Technologies, Springer Berlin Heidelberg, 2013, pp. 11–29.
[5] A. Alkhalid, T. Amin, I. Chikalov, S. Hussain, M. Moshkov, B. Zielosko, Computational Informatics, Social Factors and New Information Technologies: Hypermedia Perspectives and Avant-Garde Experiencies in the Era of Communicability Expansion, Blue Herons, 2011, Ch. Dagger: a tool for analysis and optimization of decision trees and rules, pp. 29–39.
[6] A. Frank, A. Asuncion, UCI Machine Learning Repository (2010).
URL `http://archive.ics.uci.edu/ml`
[7] I. Chikalov, S. Hussain, M. Moshkov, Relationships for cost and uncertainty of decision trees, in: A. Skowron, Z. Suraj (Eds.), Rough Sets and Intelligent Systems - Professor ZdzisÅaw Pawlak in Memoriam, Vol. 43 of Intelligent Systems Reference Library, Springer Berlin Heidelberg, 2013, pp. 203–222.
[8] P. Beame, M. E. Saks, J. S. Thathachar, Time-space tradeoffs for branching programs, in: FOCS, 1998, pp. 254–263.
[9] M. J. Moshkov, Time complexity of decision trees, in: J. F. Peters, A. Skowron (Eds.), T. Rough Sets, Vol. 3400 of Lecture Notes in Computer Science, Springer, Heidelberg, 2005, pp. 244–459.
[10] R. Church, Numerical analysis of certain free distributive structures, Duke Mathematical Journal (6) (1940) 732–734.