Experimental Study of Algorithms for Transforming Decision Rule Systems into Decision Trees

Kerven Durdymyradov[a] and Mikhail Moshkov[b]
Computer, Electrical and Mathematical Sciences & Engineering Division,
King Abdullah University of Science and Technology (KAUST),
Thuwal 23955-6900, Saudi Arabia

Keywords: Algorithms, Decision Trees, Decision Rule Systems.

Abstract: The examination of the relationships between decision trees and systems of decision rules represents a significant area of research within computer science. While methods for converting decision trees into systems of decision rules are well-established and straightforward, the inverse transformation problem presents considerable challenges. Our previous work has demonstrated that the complexity of constructing complete decision trees can be superpolynomial in many cases. In our book, we proposed three polynomial time algorithms that do not construct the entire decision tree but instead outline the computation path within this tree for a specified input. Additionally, we introduced a dynamic programming algorithm that calculates the minimum depth of a decision tree corresponding to a given decision rule system. In the present paper, we describe these algorithms and the theoretical results obtained in the book. The primary objective of this paper is to experimentally compare the performance of the three algorithms and evaluate their outcomes against the optimal results generated by the dynamic programming algorithm.

1 INTRODUCTION

Decision trees (AbouEisha et al., 2019; Breiman et al., 1984; Moshkov, 2005; Quinlan, 1993; Rokach and Maimon, 2007) and decision rule systems (Boros et al., 1997; Chikalov et al., 2013; Fürnkranz et al., 2012; Moshkov and Zielosko, 2011; Pawlak, 1991; Pawlak and Skowron, 2007) are extensively utilized for knowledge representation, served as classifiers that predict outcomes for new objects and as algorithms for addressing various problems in fault diagnosis, combinatorial optimization, and more. Both decision trees and rules rank among the most interpretable models for classification and knowledge representation (Molnar, 2022).

The investigation of the relationships between decision trees and decision rule systems is a significant area of study within computer science. While the methods for converting decision trees into systems of decision rules are well-established and straightforward (Quinlan, 1987; Quinlan, 1993; Quinlan, 1999), this paper focuses on the complexities associated with the inverse transformation problem, which is far from trivial.

This paper advances the syntactic approach to the investigation of the problem, as proposed in the works of (Moshkov, 1998; Moshkov, 2001). This approach operates under the assumption that, rather than having access to input data, we possess only a system of decision rules that needs to be transformed into a decision tree.

Consider a system S of decision rules of the form

$$(a_{i_1} = \delta_1) \wedge \cdots \wedge (a_{i_m} = \delta_m) \rightarrow \sigma,$$

where $a_{i_1}, \ldots, a_{i_m}$ are attributes, $\delta_1, \ldots, \delta_m$ are their corresponding values, and σ represents a decision.

We examine the algorithmic problem known as Extended All Rules (denoted as $EAR(S)$), which involves determining, for a given input (a tuple of values for all attributes included in S), all rules that can be satisfied by this input (i.e., those with a true left-hand side) or establishing that no such rules exist. The term "Extended" signifies that in this context, any attribute can assume any value, not just those found within the system S.

Our objective is to minimize the number of queries required to retrieve attribute values. To achieve this, we investigate decision trees as potential algorithms for addressing the stated problem.

[a] https://orcid.org/0009-0009-7445-3807
[b] https://orcid.org/0000-0003-0085-9483

Durdymyradov, K. and Moshkov, M.
Experimental Study of Algorithms for Transforming Decision Rule Systems into Decision Trees.
DOI: 10.5220/0013533300004000

In *Proceedings of the 17th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2025) - Volume 1: KDIR*, pages 173-180

In our work (Durdymyradov and Moshkov, 2024), we demonstrated that there exist systems of decision rules for which the minimum depth of the decision trees required to solve the problem is significantly less than the total number of attributes in the rule system. For these specific systems of decision rules, utilizing decision trees is a reasonable approach.

Our other work (Durdymyradov and Moshkov, 2023) demonstrated that the complexity of constructing complete decision trees can be superpolynomial in many cases. Therefore in our book (Durdymyradov et al., 2024), we considered also a different approach: rather than constructing the entire decision tree, we focus on polynomial time algorithms that describe the functioning of the decision tree for a given tuple of attribute values.

In (Durdymyradov et al., 2024), we explored three such algorithms. Two of these algorithms are based on node covers for the hypergraph associated with the decision rule system, where the nodes represent attributes from the rule system and the edges correspond to the rules. The third algorithm is inherently greedy in its approach. Additionally, we examined a dynamic programming algorithm that, given a decision rule system, returns the minimum depth of decision trees for that system. In (Durdymyradov et al., 2024), we did not study the considered algorithms experimentally.

The primary objective of the present paper is to experimentally compare the performance of the three algorithms and to evaluate their outcomes against the optimal results produced by the dynamic programming algorithm.

In our experiments, we measured execution time, the average path length, and the depth of decision trees generated by each algorithm. The results showed that the third algorithm which uses the greedy approach, consistently produced shorter paths and required less time than the other two algorithms. Moreover, the trees constructed by that greedy algorithm had depths closely approximating the optimal values achieved by the dynamic programming algorithm. Although others achieved nontrivial theoretical bounds on the accuracy, no theoretical bound was obtained for the greedy one. These experimental results reveal an interesting contrast between theoretical expectations and practical outcomes.

This paper consists of seven sections. Section 2 presents the main definitions and notation for this paper extracted from the book (Durdymyradov et al., 2024). Sections 3 and 4 discuss the results obtained in (Durdymyradov et al., 2024) for three algorithms that construct not the entire decision tree, but the computation path within the tree for a given input. Section 5 considers a dynamic programming algorithm from (Durdymyradov et al., 2024) that determines the minimum depth of a decision tree for a given decision rule system. Section 6 presents the experimental results, comparing the performance of the three algorithms and evaluating their efficiency against the dynamic programming approach. Finally, Sect. 7 summarizes the findings.

2 MAIN DEFINITIONS AND NOTATION

In this section, we consider main definitions and notation for this paper extracted from the book (Durdymyradov et al., 2024).

2.1 Decision Rule Systems

Let $\omega = \{0, 1, 2, \ldots\}$ and $A = \{a_i : i \in \omega\}$. Elements of the set A will be called *attributes*.

A *decision rule* is an expression of the form

$$(a_{i_1} = \delta_1) \wedge \cdots \wedge (a_{i_m} = \delta_m) \to \sigma,$$

where $m \in \omega$, $a_{i_1}, \ldots, a_{i_m}$ are pairwise different attributes from A and $\delta_1, \ldots, \delta_m, \sigma \in \omega$.

We denote this decision rule by r. The expression $(a_{i_1} = \delta_1) \wedge \cdots \wedge (a_{i_m} = \delta_m)$ will be called the *left-hand side*, and the number σ will be called the *right-hand side* of the rule r. The number m will be called the *length* of the decision rule r. Denote $A(r) = \{a_{i_1}, \ldots, a_{i_m}\}$ and $K(r) = \{a_{i_1} = \delta_1, \ldots, a_{i_m} = \delta_m\}$. If $m = 0$, then $A(r) = K(r) = \emptyset$. We assume that each decision rule has an *identifier* and identifiers are elements of a linearly ordered set.

Two decision rules r_1 and r_2 are *equal* if $K(r_1) = K(r_2)$ and the right-hand sides of the rules r_1 and r_2 are equal. Two decision rules are *identical* if they are equal and have equal identifiers.

A *system of decision rules* S is a finite nonempty set of decision rules with pairwise differen identifiers. The system S can have equal rules with different identifiers. Two decision rule systems S_1 and S_2 are *equal* if there is a one-to-one correspondence $\mu : S_1 \to S_2$ such that, for any rule $r \in S_1$, the rules r and $\mu(r)$ are identical.

Denote $A(S) = \bigcup_{r \in S} A(r)$, $n(S) = |A(S)|$, and $d(S)$ the maximum length of a decision rule from S. Let $n(S) > 0$. For $a_i \in A(S)$, let $V_S(a_i) = \{\delta : a_i = \delta \in \bigcup_{r \in S} K(r)\}$ and $EV_S(a_i) = V_S(a_i) \cup \{*\}$, where the symbol $*$ is interpreted as a number that does not belong to the set $V_S(a_i)$. Denote $k(S) = \max\{|V_S(a_i)| : a_i \in A(S)\}$. If $n(S) = 0$, then $k(S) = 0$. We denote by Σ the set of systems of decision rules.

Let $S \in \Sigma$, $n(S) > 0$, and $A(S) = \{a_{j_1}, \ldots, a_{j_n}\}$, where $j_1 < \cdots < j_n$. Denote $EV(S) = EV_S(a_{j_1}) \times \cdots \times EV_S(a_{j_n})$. For $\bar{\delta} = (\delta_1, \ldots, \delta_n) \in EV(S)$, denote $K(S, \bar{\delta}) = \{a_{j_1} = \delta_1, \ldots, a_{j_n} = \delta_n\}$.

We will say that a decision rule r from S is *realizable* for a tuple $\bar{\delta} \in EV(S)$ if $K(r) \subseteq K(S, \bar{\delta})$. It is clear that any rule with an empty left-hand side is realizable for the tuple $\bar{\delta}$.

We now define the problem *Extended All Rules* related to the rule system S: for a given tuple $\bar{\delta} \in EV(S)$, it is required to find the set of rules from S that are realizable for the tuple $\bar{\delta}$. We denote this problem $EAR(S)$.

In the special case, when $n(S) = 0$, all rules from S have an empty left-hand side. In this case, it is natural to consider the set S as the solution to the problem $EAR(S)$.

2.2 Decision Trees

A *finite directed tree with root* is a finite directed tree in which only one node has no entering edges. This node is called the *root*. The nodes without leaving edges are called *terminal* nodes. The nodes that are not terminal will be called *working* nodes. A *complete path* in a finite directed tree with root is a sequence $\xi = v_1, d_1, \ldots, v_m, d_m, v_{m+1}$ of nodes and edges of this tree in which v_1 is the root, v_{m+1} is a terminal node and, for $i = 1, \ldots, m$, the edge d_i leaves the node v_i and enters the node v_{i+1}.

A *decision tree over a decision rule system* S is a labeled finite directed tree with root Γ satisfying the following conditions:

- Each working node of the tree Γ is labeled with an attribute from the set $A(S)$.
- Let a working node v of the tree Γ be labeled with an attribute a_i. Then exactly $|EV_S(a_i)|$ edges leave the node v and these edges are labeled with pairwise different elements from the set $EV_S(a_i)$.
- Each terminal node of the tree Γ is labeled with a subset of the set S.

Let Γ be a decision tree over the decision rule system S. We denote by $CP(\Gamma)$ the set of complete paths in the tree Γ. Let $\xi = v_1, d_1, \ldots, v_m, d_m, v_{m+1}$ be a complete path in Γ. We correspond to this path a set of attributes $A(\xi)$ and an equation system $K(\xi)$. If $m = 0$ and $\xi = v_1$, then $A(\xi) = \emptyset$ and $K(\xi) = \emptyset$. Let $m > 0$ and, for $j = 1, \ldots, m$, the node v_j be labeled with the attribute a_{i_j} and the edge d_j be labeled with the element $\delta_j \in \omega \cup \{*\}$. Then $A(\xi) = \{a_{i_1}, \ldots, a_{i_m}\}$ and $K(\xi) = \{a_{i_1} = \delta_1, \ldots, a_{i_m} = \delta_m\}$. We denote by $\tau(\xi)$ the set of decision rules attached to the node v_{m+1}.

A system of equations $\{a_{i_1} = \delta_1, \ldots, a_{i_m} = \delta_m\}$, where $a_{i_1}, \ldots, a_{i_m} \in A$ and $\delta_1, \ldots, \delta_m \in \omega \cup \{*\}$, will be called *inconsistent* if there exist $l, k \in \{1, \ldots, m\}$ such that $l \neq k$, $i_l = i_k$, and $\delta_l \neq \delta_k$. If the system of equations is not inconsistent, then it will be called *consistent*.

Let S be a decision rule system and Γ be a decision tree over S. We will say that Γ *solves* the problem $EAR(S)$ if any path $\xi \in CP(\Gamma)$ with consistent system of equations $K(\xi)$ satisfies the following conditions:

- For any decision rule $r \in \tau(\xi)$, the relation $K(r) \subseteq K(\xi)$ holds.
- For any decision rule $r \in S \setminus \tau(\xi)$, the system of equations $K(r) \cup K(\xi)$ is inconsistent.

For any complete path $\xi \in CP(\Gamma)$, we denote by $h(\xi)$ the number of working nodes in ξ. The value $h(\Gamma) = \max\{h(\xi) : \xi \in CP(\Gamma)\}$ is called the *depth* of the decision tree Γ.

Let S be a decision rule system. We denote by $h(S)$ the minimum depth of a decision tree over S, which solves the problem $EAR(S)$.

Let $n(S) = 0$. Then there is only one decision tree solving the problem $EAR(S)$. This tree consists of one node labeled with the set of rules S. Therefore if $n(S) = 0$, then $h(S) = 0$.

2.3 Additional Definitions

In this section, we consider definitions that will be used in the description of algorithms.

Let S be a decision rule system and $\alpha = \{a_{i_1} = \delta_1, \ldots, a_{i_m} = \delta_m\}$ be a consistent equation system such that $a_{i_1}, \ldots, a_{i_m} \in A$ and $\delta_1, \ldots, \delta_m \in \omega \cup \{*\}$. We now define a decision rule system S_α. Let r be a decision rule for which the equation system $K(r) \cup \alpha$ is consistent. We denote by r_α the decision rule obtained from r by the removal from the left-hand side of r all equations that belong to α. The rules r and r_α have the same identifiers. We will say that the rule r_α *corresponds* to the rule r. Then S_α is the set of decision rules r_α such that $r \in S$ and the equation system $K(r) \cup \alpha$ is consistent.

We correspond to a decision rule system S a hypergraph $G(S)$ with the set of nodes $A(S)$ and the set of edges $\{A(r) : r \in S\}$. A *node cover* of the hypergraph $G(S)$ is a subset B of the set of nodes $A(S)$ such that $A(r) \cap B \neq \emptyset$ for any rule $r \in S$ with $A(r) \neq \emptyset$. If $A(S) = \emptyset$, then the empty set is the only node cover of the hypergraph $G(S)$. Denote by $\beta(S)$ the minimum cardinality of a node cover of the hypergraph $G(S)$.

Let S be a decision rule system with $n(S) > 0$. We denote by S^+ the subsystem of S containing only rules of the length $d(S)$.

Two decision rules r_1 and r_2 from S^+ are called *equivalent* if $K(r_1) = K(r_2)$. This equivalence relation provides a partition of the set S^+ into equivalence classes. We denote by $S^{\max}$ the set of rules that contains exactly one representative from each equivalence class and does not contain any other rules. It is clear that a set of attributes is a node cover for the hypergraph $G(S^+)$ if and only if this set is a node cover for the hypergraph $G(S^{\max})$.

3 ALGORITHMS BASED ON NODE COVERS

In this section, we consider two polynomial time algorithms from the book (Durdymyradov et al., 2024) that, for a given tuple of attribute values, describe the work on this tuple of a decision tree, which solves the problem $EAR(S)$. These algorithms are based on node covers for hypergraphs corresponding to decision rule systems.

3.1 Algorithms for Construction of Node Covers

In this section, we consider two algorithms for the construction of node covers.

3.1.1 Algorithm $\mathcal{N}_c$

Let S be a decision rule system with $n(S) > 0$. We now describe a polynomial time algorithm $\mathcal{N}_c$ for the construction of a node cover B for the hypergraph $G(S^+)$ such that $|B| \leq \beta(S^+)d(S)$.

Algorithm $\mathcal{N}_c$.

Set $B = \emptyset$. We find in S^+ the rule r_1 with the minimum identifier and add all attributes from $A(r_1)$ to B. We remove from S^+ all rules r such that $A(r_1) \cap A(r) \neq \emptyset$. Denote the obtained system by S_1^+. If $S_1^+ = \emptyset$, then B is a node cover of $G(S^+)$. If $S_1^+ \neq \emptyset$, then we find in S_1^+ the rule r_2 with the minimum identifier and add all attributes from $A(r_2)$ to B. We remove from S_1^+ all rules r such that $A(r_2) \cap A(r) \neq \emptyset$. Denote the obtained system by S_2^+. If $S_2^+ = \emptyset$, then B is a node cover of $G(S^+)$. If $S_2^+ \neq \emptyset$, then we find in S_2^+ the rule r_3 with the minimum identifier, and so on.

3.1.2 Algorithm $\mathcal{N}_g$

Let S be a decision rule system with $n(S) > 0$. We now describe a polynomial time algorithm $\mathcal{N}_g$ for the construction of a node cover B for the hypergraph $G(S^+)$ such that $|B| \leq \beta(S^+) \ln |S^{\max}| + 1$. We will say that an attribute a_i covers a rule $r \in S^{\max}$ if $a_i \in A(r)$.

Algorithm $\mathcal{N}_g$.

Set $B = \emptyset$. During each step, the algorithm chooses an attribute $a_i \in A(S^{\max})$ with the minimum index i, which covers the maximum number of rules from $S^{\max}$ uncovered during previous steps, and add it to the set B. The algorithm will finish the work when all rules from $S^{\max}$ are covered.

3.2 Algorithm $\mathcal{A}_{\mathcal{N}}$, $\mathcal{N} \in \{\mathcal{N}_c, \mathcal{N}_g\}$

Let S be a decision rule system with $n(S) > 0$ and $\mathcal{N} \in \{\mathcal{N}_c, \mathcal{N}_g\}$. We now describe a polynomial time algorithm $\mathcal{A}_{\mathcal{N}}$ that, for a given tuple of attribute values $\bar{\delta}$ from the set $EV(S)$, describes the work on this tuple of a decision tree Γ, which solves the problem $EAR(S)$.

Algorithm $\mathcal{A}_{\mathcal{N}}$.

Set $Q := S$.

Step 1. If $Q = \emptyset$ or all rules from Q have an empty left-hand side, then the tree Γ finishes its work. The result of this work is the set of decision rules from S that correspond to the rules with an empty left-hand side from the set Q. Otherwise, we move on to Step 2.

Step 2. Using the algorithm $\mathcal{N}$, we construct a node cover B of the hypergraph $G(Q^+)$. The decision tree Γ sequentially computes values of the attributes from the set B ordered by ascending attribute indices. As a result, we obtain a system α consisting of $|B|$ equations of the form $a_{i_j} = \delta_j$, where $a_{i_j} \in B$ and δ_j is the computed value of the attribute a_{i_j} (value of this attribute from the tuple $\bar{\delta}$). Set $Q := Q_\alpha$ and move on to Step 1.

We now consider bounds on accuracy obtained in (Durdymyradov et al., 2024) for the algorithm $\mathcal{A}_{\mathcal{N}}$, $\mathcal{N} \in \{\mathcal{N}_c, \mathcal{N}_g\}$.

Theorem 1. *Let S be a decision rule system with $n(S) > 0$. Then the algorithm $\mathcal{A}_{\mathcal{N}_c}$ describes the work of a decision tree Γ, which solves the problem $EAR(S)$ and for which $h(\Gamma) \leq h(S)^3$.*

Theorem 2. *Let S be a decision rule system with $n(S) > 0$. Then the algorithm $\mathcal{A}_{\mathcal{N}_g}$ describes the work of a decision tree Γ, which solves the problem $EAR(S)$ and for which $h(\Gamma) \leq h(S)^3 \ln(k(S)+1) + h(S)$.*

4 COMPLETELY GREEDY ALGORITHM

Let S be a decision rule system with $n(S) > 0$. We now consider a polynomial time algorithm $\mathcal{A}_G$ from the book (Durdymyradov et al., 2024) that, for a given tuple of attribute values $\bar{\delta}$ from the set $EV(S)$, describes the work on this tuple of a decision tree Γ, which solves the problem $EAR(S)$. This algorithm is a completely greedy algorithm by nature.

Algorithm $\mathcal{A}_G$.

Set $Q := S$.

Step 1. Set $P := Q$. If $P = \emptyset$ or all rules from P have an empty left-hand side, then the tree Γ finishes its work. The result of this work is the set of decision rules from S that correspond to the rules with an empty left-hand side from the set P. Otherwise, we move on to Step 2.

Step 2. We choose an attribute $a_i \in A(P)$ with the minimum index i, which covers the maximum number of rules from P. The decision tree Γ computes the value of this attribute. As a result, we obtain a system of equations $\{a_i = \delta\}$, where δ is the computed value of the attribute a_i (value of this attribute from the tuple $\bar{\delta}$). Set $Q := P_{\{a_i=\delta\}}$ and move on to Step 1.

One can show that the decision tree Γ described by the algorithm $\mathcal{A}_G$ solves the problem $EAR(S)$. We have no bounds on the accuracy of the algorithm $\mathcal{A}_G$.

5 DYNAMIC PROGRAMMING ALGORITHM

In this section, we consider a dynamic programming algorithm from the book (Durdymyradov et al., 2024) that, for a decision rule system S with $n(S) > 0$, computes the minimum depth of a decision tree solving the problem $EAR(S)$. This algorithm uses a directed acyclic graph (DAG), which nodes are systems of decision rules.

5.1 Construction of DAG $\Delta(S)$

Let S be a decision rule system with $n(S) > 0$. We now describe an algorithm $\mathcal{A}_{DAG}$ that constructs a DAG $\Delta(S)$.

Algorithm $\mathcal{A}_{DAG}$.

Step 1. Construct a DAG Δ containing only one node S that is not marked as processed and move on to Step 2.

Step 2. If all nodes in Δ are marked as processed, then return this DAG as $\Delta(S)$. The work of the algorithm is finished.

Otherwise, choose a node Q that is not marked as processed. If the set of rules Q is empty or contains only rules with an empty left-hand side, then mark the node Q as processed and move on to Step 2.

Otherwise, for each attribute $a_i \in A(Q)$ and each number $\delta \in EV_S(a_i)$, add to Δ an edge that leaves the node Q, enters the node $Q_{\{a_i=\delta\}}$, and is labeled with the equation $a_i = \delta$. If the node $Q_{\{a_i=\delta\}}$ does not belong to Δ, then add it to Δ and do not mark it as processed. Mark the node Q as processed and move on to Step 2.

A node Q of the DAG $\Delta(S)$ will be called *terminal* if this node has no leaving edges.

5.2 Dynamic Programming Algorithm $\mathcal{A}_{DP}$

Let S be a decision rule system with $n(S) > 0$. We now describe a dynamic programming algorithm A_{DP}, which returns the minimum depth $h(S)$ of a decision tree that solves the problem $EAR(S)$. This algorithm is used the DAG $\Delta(S)$. During each step, the algorithm labels a node Q of the DAG $\Delta(S)$ with a number $H(Q)$.

Algorithm $\mathcal{A}_{DP}$.

Step 1. If the node S of the DAG $\Delta(S)$ is labeled with a number $H(S)$, then return this number as $h(S)$. The algorithm finishes its work. Otherwise, move on to Step 2.

Step 2. Choose a node Q in $\Delta(S)$ that is not labeled with a number and such that either Q is a terminal node or all children of Q are labeled with numbers.

Let node Q be a terminal node. Then label this node with the number $H(Q) = 0$ and move on to Step 1.

Let all children of the node Q be labeled with numbers. For each $a_i \in A(Q)$, compute the number $H(Q,a_i) = 1 + \max\{H(Q_{\{a_i=\delta\}}) : \delta \in EV_S(a_i)\}$, label the node Q with the number $H(Q) = \min\{H(Q,a_i) : a_i \in A(Q)\}$ and move on to Step 1.

The correctness of this algorithm was proved in (Durdymyradov et al., 2024).

Theorem 3. *Let S be a decision rule system with $n(S) > 0$. Then the number $H(S)$ returned by the algorithm $\mathcal{A}_{DP}$ is equal to the minimum depth $h(S)$ of a decision tree solving the problem $EAR(S)$.*

6 EXPERIMENTAL RESULTS

This section presents the experimental outcomes of the algorithms utilized in this study. The decision rule systems used in the experiments were generated by the algorithm similar to described in (Moshkov and Zielosko, 2011), which constructs decision rule systems from decision tables (datasets). The datasets employed for these experiments were obtained from the UCI Machine Learning Repository (Kelly et al., 2024). In some datasets, there are missing values. Each such value was replaced with the most common value of the corresponding attribute. In some datasets, there are equal rows with, possibly, different decisions. In this case, each group of identical rows is replaced with a single row from the group with the most common decision for this group. A summary of the decision rule systems, along with their relevant characteristics, is provided in Table 1. Names of decision rule systems coincide with the names of the initial datasets.

6.1 First Group of Experiments

In the first group of experiments, we considered five decision rule systems: Balance Scale (BS), Breast Cancer (BC), Car Evaluation (CE), Congressional Voting Records (CVR), and Tic-Tac-Toe Endgame (TTT). Three algorithms were applied to these systems: $\mathcal{A}_{\mathcal{N}_c}$, $\mathcal{A}_{\mathcal{N}_g}$, and $\mathcal{A}_G$. The performance of each algorithm was measured in terms of the maximum length of complete paths across all possible input tuples (the depth of decision trees generated), the average length of complete paths across all possible input tuples, and the average computation time per input tuple in milliseconds. These results are summarized in Table 2.

6.2 Second Group of Experiments

In the second group of experiments, we considered four larger decision rule systems: Chess (Ch), Molecular Biology (MB), Mushroom (M), and Soybean (S). Instead of considering all possible input tuples, we randomly selected 1000 input tuples. The same three algorithms were applied, and the results are shown in Table 3, using the same performance metrics as in the first group of experiments.

6.3 Third Group of Experiments

In the third group of experiments, subsystems were created by randomly selecting 10 sample rules from the decision rule systems: Balance Scale (BS), Breast Cancer (BC), Car Evaluation (CE), Congressional Voting Records (CVR), and Tic-Tac-Toe Endgame (TTT). The depth, average path length, and runtime were calculated for each subsystem across four algorithms: $\mathcal{A}_{DP}$, $\mathcal{A}_{\mathcal{N}_c}$, $\mathcal{A}_{\mathcal{N}_g}$, and $\mathcal{A}_G$. This process was repeated 100 times to obtain the average values of these metrics, as shown in Table 4. The runtime is reported as the average time for processing one subsystem, in seconds.

In our experiments, we evaluated the performance of three algorithms, $\mathcal{A}_{\mathcal{N}_c}$, $\mathcal{A}_{\mathcal{N}_g}$, and $\mathcal{A}_G$, by measuring their efficiency in constructing complete paths in the decision trees for the given tuples of attribute values. Our results indicate that, in most cases, the maximum and average lengths of paths produced by $\mathcal{A}_G$ are smaller than those generated by the other two algorithms. A similar trend was observed in terms of computational time.

Additionally, we compared the depth of the decision trees constructed by the three algorithms with the minimum depth obtained by the algorithm $\mathcal{A}_{DP}$. Among the tested methods, $\mathcal{A}_G$ produced results closest to the optimal ones. Following this, $\mathcal{A}_{\mathcal{N}_g}$ achieved the next best performance, with $\mathcal{A}_{\mathcal{N}_c}$ ranking third.

Interestingly, the theoretical analysis revealed that $\mathcal{A}_{\mathcal{N}_c}$ offers the strongest theoretical bounds on accuracy, followed by $\mathcal{A}_{\mathcal{N}_g}$. However, we were unable to derive any theoretical bounds for the performance of $\mathcal{A}_G$.

7 CONCLUSIONS

In this paper, we examined three polynomial time algorithms presented in the book (Durdymyradov et al., 2024) that focus on outlining the computation path within decision trees for a specified input, rather than constructing the entire tree. We performed an experimental comparison of the performance of these algorithms and evaluated their results against the optimal outcomes produced by the dynamic programming algorithm.

Our experiments revealed that $\mathcal{A}_G$ consistently outperformed the other algorithms in terms of both path length and execution time, indicating its efficiency. Moreover, $\mathcal{A}_G$ constructed decision trees with depths closest to the optimal values achieved by the dynamic programming benchmark. While $\mathcal{A}_{\mathcal{N}_c}$ provided the best theoretical bound on the accuracy, followed by $\mathcal{A}_{\mathcal{N}_g}$, no theoretical bounds were derived for $\mathcal{A}_G$. These experimental results reveal an interesting contrast between theoretical expectations and practical outcomes.

ACKNOWLEDGEMENTS

Research reported in this publication was supported by King Abdullah University of Science and Technology (KAUST).

REFERENCES

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

Boros, E., Hammer, P. L., Ibaraki, T., and Kogan, A. (1997). Logical analysis of numerical data. *Math. Program.*, 79:163–190.

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

Chikalov, I., Lozin, V. V., Lozina, I., Moshkov, M., Nguyen, H. S., Skowron, A., and Zielosko, B. (2013). *Three Approaches to Data Analysis - Test Theory, Rough Sets and Logical Analysis of Data*, volume 41 of *Intelligent Systems Reference Library*. Springer.

Durdymyradov, K. and Moshkov, M. (2023). Construction of decision trees and acyclic decision graphs from decision rule systems. *CoRR*, arXiv:2305.01721.

Durdymyradov, K. and Moshkov, M. (2024). Bounds on depth of decision trees derived from decision rule systems with discrete attributes. *Ann. Math. Artif. Intell.*, 92(3):703–732.

Durdymyradov, K., Moshkov, M., and Ostonov, A. (2024). *Decision Trees Versus Systems of Decision Rules. A Rough Set Approach*, volume 160 of *Studies in Big Data*. Springer. (to appear).

Fürnkranz, J., Gamberger, D., and Lavrac, N. (2012). *Foundations of Rule Learning*. Cognitive Technologies. Springer.

Kelly, M., Longjohn, R., and Nottingham, K. (2024). The UCI Machine Learning Repository.

Molnar, C. (2022). *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 2 edition.

Moshkov, M. (1998). Some relationships between decision trees and decision rule systems. In Polkowski, L. and Skowron, A., editors, *Rough Sets and Current Trends in Computing, First International Conference, RSCTC'98, Warsaw, Poland, June 22-26, 1998, Proceedings*, volume 1424 of *Lecture Notes in Computer Science*, pages 499–505. Springer.

Moshkov, M. (2001). On transformation of decision rule systems into decision trees. In *Proceedings of the Seventh International Workshop Discrete Mathematics and its Applications, Moscow, Russia, January 29 – February 2, 2001, Part 1*, pages 21–26. Center for Applied Investigations of Faculty of Mathematics and Mechanics, Moscow State University. (in Russian).

Moshkov, M. (2005). Time complexity of decision trees. In Peters, J. F. and Skowron, A., editors, *Trans. Rough Sets III*, volume 3400 of *Lecture Notes in Computer Science*, pages 244–459. Springer.

Moshkov, M. and Zielosko, B. (2011). *Combinatorial Machine Learning - A Rough Set Approach*, volume 360 of *Studies in Computational Intelligence*. Springer.

Pawlak, Z. (1991). *Rough Sets - Theoretical Aspects of Reasoning about Data*, volume 9 of *Theory and Decision Library: Series D*. Kluwer.

Pawlak, Z. and Skowron, A. (2007). Rudiments of rough sets. *Inf. Sci.*, 177(1):3–27.

Quinlan, J. R. (1987). Generating production rules from decision trees. In McDermott, J. P., editor, *Proceedings of the 10th International Joint Conference on Artificial Intelligence. Milan, Italy, August 23-28, 1987*, pages 304–307. Morgan Kaufmann.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Quinlan, J. R. (1999). Simplifying decision trees. *Int. J. Hum. Comput. Stud.*, 51(2):497–510.

Rokach, L. and Maimon, O. (2007). *Data Mining with Decision Trees - Theory and Applications*, volume 69 of *Series in Machine Perception and Artificial Intelligence*. World Scientific.

APPENDIX

Table 1: Details of the decision rule systems used in the experiments.

Name of decision rule system S	$n(S)$	$d(S)$	$k(S)$	$\|S\|$
Balance Scale (BS)	4	4	5	625
Breast Cancer (BC)	9	6	11	266
Car Evaluation (CE)	6	6	4	1728
Congressional Voting Records (CVR)	16	4	2	279
Tic-Tac-Toe Endgame (TTT)	9	5	3	958
Chess (King-Rook vs. King-Pawn) (Ch)	35	11	3	3133
Molecular Biology (Splice-junction Gene Sequences) (MB)	60	5	5	3005
Mushroom (M)	19	2	10	8124
Soybean (Large) (S)	32	5	7	303

Table 2: Performance comparison of three algorithms for the first group of experiments.

	Depth			**Avg. path length**			**Avg. time**		
	$\mathcal{A}_{\mathcal{N}_c}$	$\mathcal{A}_{\mathcal{N}_g}$	$\mathcal{A}_G$	$\mathcal{A}_{\mathcal{N}_c}$	$\mathcal{A}_{\mathcal{N}_g}$	$\mathcal{A}_G$	$\mathcal{A}_{\mathcal{N}_c}$	$\mathcal{A}_{\mathcal{N}_g}$	$\mathcal{A}_G$
BS	4	4	4	4.00	**3.84**	**3.84**	7.17	4.37	**1.96**
BC	9	9	9	7.42	6.46	**6.20**	3.27	8.43	**2.93**
CE	6	6	6	6.00	6.00	**5.86**	**1.44**	9.46	3.86
CVR	16	16	16	12.75	11.79	**11.04**	9.28	19.76	**5.38**
TTT	9	9	9	9.00	8.90	**8.79**	7.48	11.15	**4.99**

Table 3: Performance comparison for larger decision rule systems using random samples of 1000 input tuples.

	Max. path length			**Avg. path length**			**Avg. time**		
	$\mathcal{A}_{\mathcal{N}_c}$	$\mathcal{A}_{\mathcal{N}_g}$	$\mathcal{A}_G$	$\mathcal{A}_{\mathcal{N}_c}$	$\mathcal{A}_{\mathcal{N}_g}$	$\mathcal{A}_G$	$\mathcal{A}_{\mathcal{N}_c}$	$\mathcal{A}_{\mathcal{N}_g}$	$\mathcal{A}_G$
Ch	35	35	35	34.33	34.62	**33.73**	56.96	124.49	**30.75**
MB	56	**48**	**48**	55.12	47.12	**46.33**	144.05	493.92	**136.84**
M	18	18	18	17.01	**15.54**	16.01	394.69	47.80	**35.61**
S	32	28	**27**	29.79	25.47	**24.47**	47.27	62.72	**13.90**

Table 4: Results for subsystems created from 10 randomly selected sample rules, averaged over 100 iterations.

	Depth				**Avg. path length**			**Avg. time**			
	$\mathcal{A}_{DP}$	$\mathcal{A}_{\mathcal{N}_c}$	$\mathcal{A}_{\mathcal{N}_g}$	$\mathcal{A}_G$	$\mathcal{A}_{\mathcal{N}_c}$	$\mathcal{A}_{\mathcal{N}_g}$	$\mathcal{A}_G$	$\mathcal{A}_{DP}$	$\mathcal{A}_{\mathcal{N}_c}$	$\mathcal{A}_{\mathcal{N}_g}$	$\mathcal{A}_G$
BS	4.0	4.0	4.0	4.0	3.93	2.59	**2.35**	**0.75**	1.27	3.58	1.58
BC	**6.15**	6.86	6.2	**6.15**	5.35	3.72	**3.38**	62.03	32.61	68.85	**24.32**
CE	**5.62**	5.81	5.67	5.64	5.26	4.30	**3.68**	3.99	3.89	7.91	**3.18**
CVR	**6.38**	6.92	6.49	**6.38**	5.48	4.07	**3.73**	70.43	2.28	4.05	**1.44**
TTT	**8.09**	8.58	8.17	8.12	6.79	4.91	**4.48**	799.39	104.26	241.78	**81.82**