Annals of Operations Research manuscript No.
(will be inserted by the editor)

Bi-criteria Optimization Problems for Decision Rules

Fawaz Alsolami · Talha Amin · Igor Chikalov · Mikhail Moshkov

Received: date / Accepted: date

Abstract We consider bi-criteria optimization problems for decision rules and rule systems relative to length and coverage. We study decision tables with many-valued decisions in which each row is associated with a set of decisions as well as single-valued decisions where each row has a single decision. Short rules are more understandable; rules covering more rows are more general. Both of these problems – minimization of length and maximization of coverage of rules are NP-hard. We create dynamic programming algorithms which can find the minimum length and the maximum coverage of rules, and can construct the set of Pareto optimal points for the corresponding bi-criteria optimization problem. This approach is applicable for medium-sized decision tables. However, the considered approach allows us to evaluate the quality of various heuristics for decision rule construction which are applicable for relatively big datasets. We can evaluate these heuristics from the point of view of (i) single-criterion – we can compare the length or coverage of rules constructed by heuristics; and (ii) bi-criteria – we can measure the distance of a point (length, coverage) corresponding to a heuristic from the set of Pareto optimal points. The presented results show that the best heuristics from the point of view of bi-criteria optimization are not always the best ones from the point of view of single-criterion optimization.

Keywords Decision tables with many-valued decisions · Systems of decision rules · Dynamic programming · Pareto optimal points · Greedy heuristics

1 Introduction

Decision rule systems are widely used as a way to solve problems, as classifiers for unseen objects, and as a tool to extract and represent knowledge from decision tables (datasets).

Fawaz Alsolami
Computer Science Department, King Abdulaziz University, 21589 Jeddah, Saudi Arabia
E-mail: falsolami1@kau.edu.sa

Talha Amin · Igor Chikalov · Mikhail Moshkov
Computer, Electrical and Mathematical Sciences and Engineering Division
King Abdullah University of Science and Technology, Thuwal 23955-6900, Saudi Arabia
E-mail: talha.amin@kaust.edu.sa, igor.chikalov@gmail.com, mikhail.moshkov@kaust.edu.sa

For instance, data mining community prefers decision rule systems when the goal is to gain insight into data (Rivest 1987; Bostrom 1995). Many methods are proposed in the literature to construct decision rule systems either directly from data sets, for example, logical analysis of data (Hammer and Bonates 2006; Crama et al 1988; Boros et al 2000), rough set theory (Pawlak 1991; Pawlak and Skowron 2007), sequential covering (Cohen and Singer 1999; Clark and Niblett 1989; Fürnkranz et al 2012; Dembczyński et al 2010; Fürnkranz 1999; Lavrač et al 2010), or from models such as decision trees (Quinlan 1993; Michalski and Pietrzykowski 2007). We consider here extensions of dynamic programming approach and greedy approach that allow us to derive a system of decision rules directly from a input dataset. In Amin et al (2013); Zielosko et al (2014) the authors considered an optimization algorithm relative to only a single objective function (either length or coverage), whereas here we consider bi-criteria optimization. The authors in Azad et al (2013) considered the average relative difference measure to evaluate the quality of the rule heuristics relative to a given cost function. However, here we take advantage of bi-criteria optimization solutions to evaluate the considered rule heuristics algorithms by calculating the normalized Euclidean distances to the set of Pareto optimal points.

Rule coverage, the number of rows (objects) for which a rule applies to, is crucial in order to discover major patterns in a given dataset whereas rule length, the number of conditions (constraints) on the left-hand side, is important for understanding and interpretability. Unfortunately, the problems of constructing rules with maximum coverage (Bonates et al 2008) or rules with minimum length (see for example Moshkov and Zielosko (2011)) are NP-hard. In this paper, we propose research tools based on dynamic programming to study single-criterion and bi-criteria optimization problems relative to the length and coverage of rules for medium-sized datasets, and apply them to study quality of rules constructed by heuristics.

Along with standard datasets where rows are labeled with a single-valued decision, we also consider multi-label datasets. Multi-label datasets (decision tables with many-valued decisions) appear in various problems of discrete optimization, pattern recognition, computational geometry, decision making, rough set theory (Greco et al 2001; Moshkov and Zielosko 2011; Pawlak 1991). In real life applications we can meet multi-label data when we study, e.g., problems of semantic annotation of images (Boutell et al 2004), music categorization into emotions (Wieczorkowska et al 2005), functional genomics (Blockeel et al 2006), and text categorization (Zhou et al 2005). In contrast with single-label data set, each row of multi-label dataset is associated with a set of decisions (labels). In this work, we consider the problems of finding an arbitrary decision from the set of decisions attached to a row.

One of the contributions of this paper is the algorithm which constructs the set of Pareto optimal points for bi-criteria optimization problem relative to length and coverage for decision rules and systems of decision rules. The construction of all Pareto optimal decision rules (patterns) for a given dataset is a computationally expensive task (Hammer et al 2004), however, in this paper we only construct all Pareto optimal points for a given dataset, the quantity of which is essentially less than the number of Pareto optimal rules. We can derive from the set of Pareto optimal points the optimal values for single-criterion optimization problems for length or coverage.

Another contribution of this paper is the investigation of 23 rule heuristics methods (10 methods for decision tables with single-valued decisions and 13 methods for decision tables with many-valued decisions) for construction of decision rules where the goal is to study the influence of such heuristics parameters to the costs of the produced rules. For single-criterion optimization, we compare the performance of rules constructed by heuristics using

ranking to avoid bias caused by differences in the size of the considered tables. For bi-criteria optimization, we can simultaneously evaluate the performance of the rule heuristics with regards to length and coverage using the normalized distance to the Pareto optimal points of a given dataset, where the greedy heuristic with shortest distance to the set of Pareto optimal points is better than other heuristic methods. It is interesting to see that the best heuristics from the point of view of bi-criteria optimization are not always the best ones from the point of view of single-criterion optimization.

This paper consists of seven sections. Section 2 defines main notions connected with decision tables, rules, and systems of rules that employ throughout this paper. In Section 3, we discuss how to describe the set of decision rules and construct the set of Pareto optimal points for systems of rules. A class of rule heuristic methods for decision rules is investigated in Section 4. The experimental setup and the results of the experiments are described in Section 5. Finally, Section 6 concludes the paper.

2 Decision Tables, Rules, and Systems of Rules

In this section, we consider main notions connected with decision tables, rules, and systems of rules.

2.1 Decision Tables

A *decision table with many-valued decisions* is a rectangular table T with $n \geq 1$ columns filled with numbers from the set $\omega = \{0, 1, 2, \ldots\}$ of nonnegative integers. Columns of the table are labeled with *conditional* attributes $f_1, \ldots, f_n$. Rows of the table are pairwise different, and each row r is labeled with a finite nonempty subset $D(r)$ of ω which is interpreted as a *set of decisions*. Rows of the table are interpreted as tuples of values of conditional attributes. We denote by $Row(T)$ the set of rows of T. Let $D(T) = \bigcup_{r \in Row(T)} D(r)$. A *decision table with single-valued decisions* is a decision table with many-valued decisions T in which $|D(r)| = 1$ for any $r \in Row(T)$. In such a table, each row r is usually labeled not with the set $D(r)$ but with the unique decision from $D(r)$.

A decision table can be represented by a word over the alphabet $\{0, 1, ;, :, |\}$ in which numbers from ω are in binary representation (are represented by words over the alphabet $\{0, 1\}$), the symbol ";" is used to separate two numbers from ω, and the symbol "|" is used to separate two rows (we add numbers from $D(r)$ at the end of each row r and separate these numbers from r by the symbol ":"). The length of this word will be called the *size* of the decision table.

A decision table is called *empty* if it has no rows. We denote by $\mathscr{T}$ the set of all decision tables with many-valued decisions and by $\mathscr{T}^+$ – the set of nonempty decision tables with many-valued decisions. Let $T \in \mathscr{T}$. The table T is called *degenerate* if it is empty or has a *common decision* – a decision $t \in D(T)$ such that $t \in D(r)$ for any row r of T. In the latter case, the *minimum common decision* is the minimum decision from $D(T)$ which is a common decision. We denote by $N(T)$ the number of rows in the table T.

For any conditional attribute $f_i \in \{f_1, \ldots, f_n\}$, we denote by $E(T, f_i)$ the set of values of the attribute f_i in the table T. We denote by $E(T)$ the set of conditional attributes for which $|E(T, f_i)| \geq 2$.

Let T be a nonempty decision table. A *subtable* of T is a table obtained from T by removal of some rows. Let $f_{i_1}, \ldots, f_{i_m} \in \{f_1, \ldots, f_n\}$ and $a_1, \ldots, a_m \in \omega$. We denote by

$T(f_{i_1},a_1)\dots(f_{i_m},a_m)$ the subtable of the table T containing the rows from T which at the intersection with the columns $f_{i_1},\dots,f_{i_m}$ have numbers $a_1,\dots,a_m$, respectively. Such nonempty subtables, including the table T, are called *separable* subtables of T. We denote by $SEP(T)$ the set of separable subtables of the table T.

2.2 Decision Rules

Let T be a decision table with n conditional attributes $f_1,\dots,f_n$ and $r=(b_1,\dots,b_n)$ be a row of T. A *decision rule over* T is an expression of the kind

$$(f_{i_1}=a_1)\wedge\dots\wedge(f_{i_m}=a_m)\to t \tag{1}$$

where $f_{i_1},\dots,f_{i_m}\in\{f_1,\dots,f_n\}$, and $a_1,\dots,a_m,t$ are numbers from ω. It is possible that $m=0$. In this case the rule (1) has the form $\emptyset\to t$. We denote $T^0=T$, and if $m>0$ we denote $T^j=T(f_{i_1},a_1)\dots(f_{i_j},a_j)$ for $j=1,\dots,m$ for the considered rule. We will say that the decision rule (1) *covers* the row r if r belongs to T^m, i.e., $b_{i_1}=a_1,\dots,b_{i_m}=a_m$.

A decision rule (1) over T is called a *decision rule for* T if it satisfies the following conditions:

- if $m=0$ then T is degenerate and t is the minimum common decision for T,
- if $m>0$ then T^m is degenerate, t is the minimum common decision for T^m, and, for $j=1,\dots,m$, T^{j-1} is not degenerate and $f_{i_j}\in E(T^{j-1})$.

A decision rule (1) for T is called a *decision rule for T and r* if it covers r. We denote by $DR(T)$ the set of decision rules for T. By $DR(T,r)$ we denote the set of decision rules for T and r.

We now consider a notion of *cost function for decision rules*. This is a function $\psi(T,\rho)$ which is defined on pairs T,ρ, where T is a nonempty decision table and ρ is a decision rule for T, and has values from the set $\mathbb{R}$ of real numbers. This cost function is given by pair of functions $\psi^0:\mathscr{T}^+\to\mathbb{R}$ and $F_\psi:\mathbb{R}\to\mathbb{R}$. The value $\psi(T,\rho)$ is defined by induction:

- If ρ is equal to $\emptyset\to t$ then $\psi(T,t)=\psi^0(T)$.
- If ρ is equal to $(f_i=a)\wedge\beta\to t$ then

$$\psi(T,(f_i=a)\wedge\beta\to t)=F_\psi(\psi(T(f_i,a),\beta\to t)).$$

The cost function ψ is called *strictly increasing* if $F_\psi(x_1)>F_\psi(x_2)$ for any $x_1,x_2\in\mathbb{R}$ such that $x_1>x_2$.

We will consider two strictly increasing cost functions for decision rules:

- The *length* $l(T,\rho)=l(\rho)$ for which $l^0(T)=0$ and $F_l(x)=x+1$. The length of the rule (1) is equal to m.
- The *coverage* $c(T,\rho)$ for which $c^0(T)=N(T)$ and $F_c(x)=x$. The coverage of the rule (1) for table T is equal to $N(T^m)$.

We need to minimize length and maximize coverage. However, we will consider only algorithms for the minimization of cost functions. Therefore, instead of maximization of coverage c we will minimize the negative coverage $-c$ given by pair of functions $-c^0(T)=-N(T)$ and $F_{-c}(x)=x$. The cost functions $-c$ is a strictly increasing cost function.

2.3 Systems of Decision Rules

Let T be a nonempty decision table with n conditional attributes $f_1,\dots,f_n$ and $N(T)$ rows $r_1,\dots,r_{N(T)}$.

A *system of decision rules for* T is an $N(T)$-tuple $S=(\rho_1,\dots,\rho_{N(T)})$ where $\rho_1 \in DR(T,r_1),\dots,\rho_{N(T)} \in DR(T,r_{N(T)})$.

We now consider a notion of *cost function for systems of decision rules*. This is a function $\psi(T,S)$ which is defined on pairs T,S, where T is a nonempty decision table and $S=(\rho_1,\dots,\rho_{N(T)})$ is a system of decision rules for T, and has values from the set $\mathbb{R}$. This function is given by cost function for decision rules ψ. The value $\psi(T,S)$ is equal to $\psi(T,\rho_1)+\ldots+\psi(T,\rho_{N(T)})$.

The cost function for systems of decision rules ψ is called *strictly increasing* if ψ is strictly increasing cost function for decision rules.

3 Construction of the Set of Pareto Optimal Points

In this section, we discuss how to describe the set of decision rules and how to construct the set of Pareto optimal points for systems of rules.

3.1 Directed Acyclic Graph $\Delta(T)$

Let T be a nonempty decision table with n conditional attributes $f_1,\dots,f_n$. We now consider an algorithm $\mathscr{A}_1$ for the construction of a directed acyclic graph $\Delta(T)$ which will be used for the description and study of decision rules. Nodes of this graph are some separable subtables of the table T. During each iteration we process one node. We start with the graph that consists of one node T which is not processed and finish when all nodes of the graph are processed.

Algorithm $\mathscr{A}_1$
Input: A nonempty decision table T with n conditional attributes $f_1,\dots,f_n$.
Output: Directed acyclic graph $\Delta(T)$.

1. Construct the graph that consists of one node T which is not marked as processed.
2. If all nodes of the graph are processed then the work of the algorithm is finished. Return the resulting graph as $\Delta(T)$. Otherwise, choose a node (table) Θ that has not been processed yet.
3. If Θ is degenerate mark the node Θ as processed and proceed to step 2.
4. If Θ is not degenerate then, for each $f_i \in E(\Theta)$, draw a bundle of edges from the node Θ. Let $f_i \in E(\Theta)$and $E(\Theta,f_i)=\{a_1,\dots,a_k\}$. If some of the nodes $\Theta(f_i,a_1),\dots,\Theta(f_i,a_k)$ are not present in the graph then add these nodes to the graph. Then create k edges from Θ to the nodes $\Theta(f_i,a_1),\dots,\Theta(f_i,a_k)$ and label these edges with the pairs $(f_i,a_1),\dots,(f_i,a_k)$, respectively. Mark the node Θ as processed and return to step 2.

Let T be a nonempty decision table with n conditional attributes $f_1,\dots,f_n$ and $G=\Delta(T)$. A node Θ of the graph G is called *terminal* if there are no edges starting in this node. A node Θ of the graph G is terminal if and only if Θ is a degenerate table.

Let τ be a directed path from a node Θ of G to a terminal node Θ' in which edges (in the order from Θ to Θ') are labeled with pairs $(f_{i_1}, c_{i_1}), \ldots, (f_{i_m}, c_{i_m})$, and t be the minimum common decision for Θ'. We denote by $rule(\tau)$ the decision rule over T

$$(f_{i_1} = c_{i_1}) \wedge \ldots \wedge (f_{i_m} = c_{i_m}) \rightarrow t.$$

If $m = 0$ (if $\Theta = \Theta'$) then the rule $rule(\tau)$ is equal to $\emptyset \rightarrow t$.

Let $r = (b_1, \ldots, b_n)$ be a row of T, and Θ be a node of G (subtable of T) containing the row r. We denote by $Rule(G, \Theta, r)$ the set of rules $rule(\tau)$ corresponding to all directed paths τ from Θ to terminal nodes Θ' containing r.

Proposition 1 *Let T be a nonempty decision table with n conditional attributes $f_1, \ldots, f_n$, $r = (b_1, \ldots, b_n)$ be a row of T, $G = \Delta(T)$, and Θ be a node of the graph G containing r. Then the set $Rule(G, \Theta, r)$ coincides with the set of all decision rules for Θ and r, i.e., $Rule(G, \Theta, r) = DR(\Theta, r)$.*

Proof From the definition of the graph G it follows that each rule from $Rule(G, \Theta, r)$ is a decision rule for Θ and r.

Let us consider an arbitrary decision rule ρ for Θ and r:

$$(f_{i_1} = b_{i_1}) \wedge \ldots \wedge (f_{i_m} = b_{i_m}) \rightarrow t.$$

It is easy to show that there is a directed path $\Theta_0 = \Theta, \Theta_1, \ldots, \Theta_m$ in G such that, for $j = 1, \ldots, m$, $\Theta_j = \Theta(f_{i_1}, b_{i_1}) \ldots (f_{i_j}, b_{i_j})$, there is an edge from Θ_{j-1} to Θ_j labeled with (f_{i_j}, b_{i_j}), and Θ_m is a terminal node in G. Therefore $\rho \in Rule(G, \Theta, r)$. □

3.2 Pareto Optimal Points

In this subsection, we consider the notion of Pareto optimal point, and an algorithm which, for a given set of points, constructs all Pareto optimal points for this set.

Let $\mathbb{R}^2$ be the set of pairs of real numbers (*points*). We consider a partial order $\leq$ on the set $\mathbb{R}^2$: $(c, d) \leq (a, b)$ if $c \leq a$ and $d \leq b$. Two points α and β are *comparable* if $\alpha \leq \beta$ or $\beta \leq \alpha$. A subset of $\mathbb{R}^2$ in which no two different points are comparable is called an *antichain*. We will write $\alpha < \beta$ if $\alpha \leq \beta$ and $\alpha \neq \beta$. If α and β are comparable then $\min(\alpha, \beta) = \alpha$ if $\alpha \leq \beta$ and $\min(\alpha, \beta) = \beta$ if $\alpha > \beta$.

Let A be a nonempty finite subset of $\mathbb{R}^2$. A point $\alpha \in A$ is called a *Pareto optimal point for A* if there is no a point $\beta \in A$ such that $\beta < \alpha$. We denote by $Par(A)$ the set of Pareto optimal points for A. It is clear that $Par(A)$ is an antichain.

We now describe an algorithm which, for a given nonempty finite subset A of the set $\mathbb{R}^2$, constructs the set $Par(A)$. We assume that A is a multiset containing, possibly, repeating elements.

Algorithm $\mathscr{A}_2$

Input: A nonempty finite subset A of the set $\mathbb{R}^2$ containing, possibly, repeating elements (multiset).

Output: The set $Par(A)$ of Pareto optimal points for A.

1. Set P equal to the empty set.
2. Construct a sequence B of all points from A ordered according to the first coordinate in the ascending order.

3. If there is only one point in the sequence B, then add this point to P, return P, and finish the work of the algorithm. Otherwise, choose the first $\alpha = (\alpha_1, \alpha_2)$ and the second $\beta = (\beta_1, \beta_2)$ points from B.
4. If α and β are comparable then remove α and β from B, add the point $\min(\alpha, \beta)$ to the beginning of B, and proceed to step 3.
5. If α and β are not comparable (in this case $\alpha_1 < \beta_1$ and $\alpha_2 > \beta_2$) then remove α from B, add the point α to P, and proceed to step 3.

Proposition 2 *Let A be a nonempty finite subset of the set $\mathbb{R}^2$ containing, possibly, repeating elements (multiset). Then the algorithm $\mathscr{A}_2$ returns the set $Par(A)$ of Pareto optimal points for A.*

Proof Let the output set P be equal to $\{(a_1, b_1), \ldots, (a_t, b_t)\}$. It is clear that, for any $\alpha \in A$, $\alpha \notin P$, there exists $\beta \in P$ such that $\beta < \alpha$. From here it follows that $Par(A) \subseteq P$ and P is an antichain. Let us assume that there exists $\gamma \in P$ which does not belong to $Par(A)$. Then there exists $\alpha \in A$ such that $\alpha < \gamma$. Since P is an antichain, $\alpha \notin P$. We know that there exists $\beta \in P$ such that $\beta \leq \alpha$. This results in two different points β and γ from P being comparable, which is impossible. Therefore $P = Par(A)$. □

3.3 Pareto Optimal Points for Decision Rules

Let T be a nonempty decision table with n conditional attributes $f_1, \ldots, f_n$, $r = (b_1, \ldots, b_n)$ be a row of T, and $G = \Delta(T)$. For each node Θ of the graph G containing r, we denote $p(G, \Theta, r) = \{(l(\Theta, \rho), -c(\Theta, \rho)) : \rho \in Rule(G, \Theta, r)\}$. We denote by $Par(p(G, \Theta, r))$ the set of Pareto optimal points for $p(G, \Theta, r)$. We now describe an algorithm $\mathscr{A}_3$ constructing the set $Par(p(G, T, r))$. In fact, this algorithm constructs, for each node Θ of the graph G containing r, the set $B(\Theta, r) = Par(p(G, \Theta, r))$.

For $(a, b) \in \mathbb{R}^2$, we denote by $(a, b)^+$ the pair $(a + 1, b)$. For a nonempty finite subset A of $\mathbb{R}^2$, we denote $A^+ = \{(a, b)^+ : (a, b) \in A\}$.

Algorithm $\mathscr{A}_3$.

Input: A nonempty decision table T with n conditional attributes $f_1, \ldots, f_n$ and a row $r = (b_1, \ldots, b_n)$ of T.

Output: The set $Par(p(G, T, r))$ of Pareto optimal points for the set of pairs $p(G, T, r) = \{(l(T, \rho), -c(T, \rho)) : \rho \in Rule(G, T, r)\}$.

1. Using Algorithm $\mathscr{A}_1$ construct the graph $G = \Delta(T)$.
2. If all nodes in G containing r are processed, then return the set $B(T, r)$. Otherwise, choose in the graph G a node Θ containing r which is not processed yet and which is either a terminal node of G or a nonterminal node of G such that, for every $f_i \in E(\Theta)$, the node $\Theta(f_i, b_i)$ is already processed, i.e., the set $B(\Theta(f_i, b_i), r)$ is already constructed.
3. If Θ is a terminal node, then set $B(\Theta, r) = \{(l^0(\Theta), -c^0(\Theta))\}$. Mark the node Θ as processed and proceed to step 2.
4. If Θ is a nonterminal node, then construct the sets $B(\Theta(f_i, b_i), r)^+$ for every $f_i \in E(\Theta)$, and construct the multiset

$$A(\Theta, r) = \bigcup_{f_i \in E(\Theta)} B(\Theta(f_i, b_i), r)^+$$

by simple transcription of elements from the sets $B(\Theta(f_i, b_i), r)^+$, $f_i \in E(\Theta)$.

5. Apply to the multiset $A(\Theta,r)$ the algorithm $\mathscr{A}_2$ which constructs the set $Par(A(\Theta,r))$. Set $B(\Theta,r) = Par(A(\Theta,r))$. Mark the node Θ as processed and proceed to step 2.

Theorem 1 *Let T be a nonempty decision table with n conditional attributes $f_1,\dots,f_n$, $r = (b_1,\dots,b_n)$ be a row of T, and $G = \Delta(T)$. Then, for each node Θ of the graph G containing r, the algorithm $\mathscr{A}_3$ constructs the set $B(\Theta,r) = Par(p(G,\Theta,r))$.*

Proof We prove the considered statement by induction on nodes of G. Let Θ be a terminal node of G containing r. Then $Rule(G,\Theta,r) = \{\emptyset \to t\}$ where t is the minimum common decision for Θ, $p(G,\Theta,r) = Par(p(G,\Theta,r)) = \{(l^0(\Theta), -c^0(\Theta))\}$, and

$$B(\Theta,r) = Par(p(G,\Theta,r)).$$

Let Θ be a nonterminal node of G containing r such that, for any $f_i \in E(\Theta)$, the considered statement holds for the node $\Theta(f_i,b_i)$, i.e., $B(\Theta(f_i,b_i),r) = Par(p(G,\Theta(f_i,b_i),r))$. It is clear that

$$p(G,\Theta,r) = \bigcup_{f_i \in E(\Theta)} p(G,\Theta(f_i,b_i),r)^+.$$

Let us show that

$$Par(p(G,\Theta,r)) \subseteq \bigcup_{f_i \in E(\Theta)} Par(p(G,\Theta(f_i,b_i),r)^+).$$

Let $\alpha \in p(G,\Theta,r) \setminus \bigcup_{f_i \in E(\Theta)} Par(p(G,\Theta(f_i,b_i),r)^+)$. Then there is $f_i \in E(\Theta)$ such that $\alpha \in p(G,\Theta(f_i,b_i),r)^+$ but $\alpha \notin Par(p(G,\Theta(f_i,b_i),r)^+)$. Therefore there exists a $\beta \in p(G,\Theta(f_i,b_i),r)^+$ such that $\beta < \alpha$. Hence $\alpha \notin Par(p(G,\Theta,r))$, and $Par(p(G,\Theta,r)) \subseteq \bigcup_{f_i \in E(\Theta)} Par(p(G,\Theta(f_i,b_i),r)^+)$.

We now show that $Par(p(G,\Theta(f_i,b_i),r)^+) = Par(p(G,\Theta(f_i,b_i),r))^+$ for any $f_i \in E(\Theta)$. Let $f_i \in E(\Theta)$ and $(a,b),(c,d) \in p(G,\Theta(f_i,b_i),r)$. It is clear that $(c,d) = (a,b)$ if and only if $(c,d)^+ = (a,b)^+$, and $(c,d) < (a,b)$ if and only if $(c,d)^+ < (a,b)^+$. From here it follows that $Par(p(G,\Theta(f_i,b_i),r)^+) = Par(p(G,\Theta(f_i,b_i),r))^+$. Therefore

$$Par(p(G,\Theta,r)) \subseteq \bigcup_{f_i \in E(\Theta)} Par(p(G,\Theta(f_i,b_i),r))^+ \subseteq p(G,\Theta,r).$$

Let us show that

$$Par(p(G,\Theta,r)) = Par\left(\bigcup_{f_i \in E(\Theta)} Par(p(G,\Theta(f_i,b_i),r))^+\right).$$

It is clear that $Par(p(G,\Theta,r)) \subseteq Par\left(\bigcup_{f_i \in E(\Theta)} Par(p(G,\Theta(f_i,b_i),r))^+\right)$. Let us assume that, for some β, $\beta \in Par\left(\bigcup_{f_i \in E(\Theta)} Par(p(G,\Theta(f_i,b_i),r))^+\right)$ and $\beta \notin Par(p(G,\Theta,r))$. Then there exists $\alpha \in p(G,\Theta,r)$ such that $\alpha < \beta$. One can show that there exists

$$\gamma \in Par(p(G,\Theta,r)) \subseteq Par\left(\bigcup_{f_i \in E(\Theta)} Par(p(G,\Theta(f_i,b_i),r))^+\right)$$

such that $\gamma \leq \alpha$. Therefore $\gamma < \beta$ and

$$\beta \notin Par\left(\bigcup_{f_i \in E(\Theta)} Par(p(G,\Theta(f_i,b_i),r))^+\right).$$

Hence $Par(p(G,\Theta,r)) = Par\left(\bigcup_{f_i \in E(\Theta)} Par(p(G,\Theta(f_i,b_i),r))^+\right)$.

Since $B(\Theta,r) = Par\left(\bigcup_{f_i \in E(\Theta)} B(\Theta(f_i,b_i),r)^+\right)$ and

$$B(\Theta(f_i,b_i),r) = Par(p(G,\Theta(f_i,b_i),r))$$

for any $f_i \in E(\Theta)$, we have $B(\Theta,r) = Par(p(G,\Theta,r))$. □

3.4 Pareto Optimal Points for Systems of Decision Rules

Let T be a nonempty decision table with n conditional attributes $f_1,\ldots,f_n$ and $N(T)$ rows $r_1,\ldots,r_{N(T)}$. We denote by $\mathscr{S}(G,T)$ the set $Rule(G,T,r_1) \times \ldots \times Rule(G,T,r_{N(T)})$ of systems of decision rules for T. From Proposition 1 it follows that $\mathscr{S}(G,T) = DR(T,r_1) \times \ldots \times DR(T,r_{N(T)})$.

We describe now an algorithm which constructs the set of Pareto optimal points for the set of pairs $p(G,T) = \{(l(T,S),-c(T,S)) : S \in \mathscr{S}(G,T)\}$. Let A,B be nonempty finite subsets of the set $\mathbb{R}^2$. We denote by $A \langle+\rangle B$ the set $\{(a+c,b+d) : (a,b) \in A,(c,d) \in B\}$.

Algorithm $\mathscr{A}_4$.

Input: A nonempty decision table T with n conditional attributes $f_1,\ldots,f_n$ and $N(T)$ rows $r_1,\ldots,r_{N(T)}$.

Output: The set $Par(p(G,T))$ of Pareto optimal points for the set of pairs $p(G,T) = \{(l(T,S),-c(T,S)) : S \in \mathscr{S}(G,T)\}$.

1. Using Algorithm $\mathscr{A}_1$ construct the graph $G = \Delta(T)$.
2. For $i = 1,\ldots,N(T)$, use algorithm $\mathscr{A}_3$ to create the set $Par(P_i)$ where $P_i = p(G,T,r_i) = \{(l(T,\rho),-c(T,\rho)) : \rho \in Rule(G,T,r_i)\}$.
3. Set $B_1 = Par(P_1)$ and $i = 2$.
4. Construct the multiset $A_i = B_{i-1} \langle+\rangle Par(P_i) = \{(a+c,b+d) : (a,b) \in B_{i-1},(c,d) \in Par(P_i)\}$ – we will not remove equal pairs from the constructed set.
5. Using algorithm $\mathscr{A}_2$, construct the set $B_i = Par(A_i)$.
6. If $i = N(T)$ then return $C(G,T) = B_i$ and finish the work of the algorithm. Otherwise, set $i = i+1$ and proceed to step 4.

Theorem 2 *Let T be a decision table with n conditional attributes $f_1,\ldots,f_n$ and $N(T)$ rows $r_1,\ldots,r_{N(T)}$, and $G = \Delta(T)$. Then the algorithm $\mathscr{A}_4$ constructs the set $C(G,T) = Par(p(G,T))$.*

Proof For $i = 1,\ldots,N(T)$, denote $P_i = p(G,T,r_i)$. During the second step, the algorithm $\mathscr{A}_4$ constructs (using the algorithm $\mathscr{A}_3$) the sets $Par(P_1),\ldots,Par(P_{N(T)})$ (see Theorem 1).

Let us show that during next steps, the algorithm $\mathscr{A}_4$ constructs the set $C(G,T) = Par(Q_{N(T)})$ where $Q_1 = P_1$, and, for $i = 2,\ldots,N(T)$, $Q_i = Q_{i-1} \langle+\rangle P_i$. We will prove by induction on i that, for $i = 1,\ldots,N(T)$, the set B_i (see the description of the algorithm $\mathscr{A}_4$) is equal to the set $Par(Q_i)$. Since $B_1 = Par(P_1)$ and $Q_1 = P_1$, we have $B_1 = Par(Q_1)$. Let

for some $i-1$, $2 \leq i \leq N(T)$, the considered statement hold, i.e., $B_{i-1} = Par(Q_{i-1})$. Then $B_i = Par(B_{i-1} \langle + \rangle Par(P_i)) = Par(Par(Q_{i-1}) \langle + \rangle Par(P_i))$. We know that $Q_i = Q_{i-1} \langle + \rangle P_i$.

Let us show that $Par(Q_i) \subseteq Par(Q_{i-1}) \langle + \rangle Par(P_i)$. Let $\beta \in Par(Q_i) = Par(Q_{i-1} \langle + \rangle P_i)$ and $\beta = (a+c, b+d)$ where $(a,b) \in Q_{i-1}$ and $(c,d) \in P_i$. One can show that there exist $(a',b') \in Par(Q_{i-1})$ and $(c',d') \in Par(P_i)$ such that $(a',b') \leq (a,b)$ and $(c',d') \leq (c,d)$. It is clear that $\alpha = (a'+c', b'+d') \leq (a+c, b+d) = \beta$, and $\alpha \in Par(Q_{i-1}) \langle + \rangle Par(P_i)$. Since $\beta \in Par(Q_{i-1} \langle + \rangle P_i)$, we have $\beta = \alpha$. Therefore $Par(Q_i) \subseteq Par(Q_{i-1}) \langle + \rangle Par(P_i)$.

Let us now show that $Par(Q_i) = Par(Par(Q_{i-1}) \langle + \rangle Par(P_i))$. We can clearly see that $Par(Q_i) \subseteq Par(Par(Q_{i-1}) \langle + \rangle Par(P_i))$. Now we assume that, for some β, $\beta \notin Par(Q_i)$ and $\beta \in Par(Par(Q_{i-1}) \langle + \rangle Par(P_i))$. Then there exists $\alpha \in Q_i$ such that $\alpha < \beta$. One can show that there exists

$$\gamma \in Par(Q_i) \subseteq Par(Q_{i-1}) \langle + \rangle Par(P_i)$$

such that $\gamma \leq \alpha$. Therefore $\gamma < \beta$ and $\beta \notin Par(Par(Q_{i-1}) \langle + \rangle Par(P_i))$. Hence $Par(Q_i) = Par(Par(Q_{i-1}) \langle + \rangle Par(P_i))$. Therefore $B_i = Par(Q_i)$. So we have $B_{N(T)} = Par(Q_{N(T)})$ and the algorithm $\mathscr{A}_4$ returns the set $C(G,T) = Par(Q_{N(T)})$. One can show that $Q_{N(T)} = p(G,T)$. Therefore $C(G,T) = Par(p(G,T))$. □

3.5 On Complexity of Algorithms $\mathscr{A}_1$, $\mathscr{A}_3$, and $\mathscr{A}_4$

One can show that the time complexity of each of the algorithms $\mathscr{A}_1$, $\mathscr{A}_3$, and $\mathscr{A}_4$ is bounded from above by a polynomial on the size of the input table T and the number $|SEP(T)|$ of different separable subtables of T.

In Moshkov (2007); Moshkov and Chikalov (2000) infinite sets of attributes (so-called restricted information systems) were studied for each of which the number of separable subtables in decision tables over the considered set of attributes is bounded from above by a polynomial on the number of attributes in the table. For decision tables over such set of attributes, the time complexity of each of the algorithms $\mathscr{A}_1$, $\mathscr{A}_3$, and $\mathscr{A}_4$ is bounded from above by a polynomial on the size of the input table.

In general case, algorithms $\mathscr{A}_1$, $\mathscr{A}_3$, and $\mathscr{A}_4$ have exponential time complexity depending on the size of the input tables. However, the considered algorithms are applicable to medium sized decision tables which allows us to use them for the study of heuristics for decision rule construction (see next section).

4 Greedy Heuristics

In this section, we describe greedy heuristics which, for a given decision table T and its row r, construct a decision rule for T and r. Due to the differences in handling decision tables with single-valued and many-valued decisions, some of the heuristics can only be calculated for one or the other. However, the main framework of the algorithm remains the same for both types of decision tables.

The 23 heuristics we consider all operate in a similar fashion. Each heuristic inductively creates a decision rule ρ for a given row $r = (b_1, ..., b_n)$ of decision table T and a decision t based on a function $F(T,r,t)$ that selects a conditional attribute from $E(T)$. A decision rule for a given row is incrementally built by starting with an empty rule and adding terms to the left-hand side. To create a system of decision rules, we construct rules for each row and combine them together as a system. The algorithm used to construct a rule for a given row is as follows:

Algorithm $\mathscr{A}_5$

Input: A decision table T with n conditional attributes $f_1,\ldots,f_n$, a row $r=(b_1,\ldots,b_n)$ from T, a decision t from $D(r)$ and a heuristic function $F(T,r,t)$.

Output: A decision rule for T and r.

1. Create a rule ρ_0 of the form $\emptyset \to t$.
2. Assume ρ_k is already defined for some $k \geq 0$, as $(f_{i_1}=b_{i_1}) \wedge \ldots \wedge (f_{i_k}=b_{i_k}) \to t$ (Step 1 defines for $k=0$.)
3. Define a subtable T^k such that $T^k = T(f_{i_1},b_{i_1})\ldots(f_{i_k},b_{i_k})$.
4. If T^k has a common decision t' (it is possible that $t \neq t'$), then replace the right-hand side of ρ_k with t' and end the algorithm with ρ_k as the constructed rule.
5. Select an attribute $f_{i_{k+1}} \in E(T^k)$ such that $F(T^k,r,t)=f_{i_{k+1}}$.
6. Define ρ_{k+1} by adding the term $(f_{i_{k+1}}=b_{i_{k+1}})$ to the left-hand side of ρ_k.
7. Repeat from step 2 with $k=k+1$.

Now we will define the heuristic functions $F(T,r,t)$ used by our algorithm. Given that T is a decision table with conditional attributes $f_1,\ldots,f_n$, $r=(b_1,\ldots,b_n)$ is a row T, and $t \in D(r)$, let $C(T,t)$ be the number of rows r' in T such that $t \in D(r')$. Let $M(T,t)=N(T)-C(T,t)$ and $U(T,t)=C(T,t)/N(T)$. For any attribute $f_j \in E(T)$, we define $a(T,r,f_j,t)=C(T,t)-C(T(f_j,b_j),t)$ and $b(T,r,f_j,t)=M(T,t)-M(T(f_j,b_j),t)$. By $mcd(T)$, we denote the *most common decision* for T which is the minimum decision t_0 from $D(T)$ such that $C(T,t_0)=\max\{C(T,t): t \in D(T)\}$

In this case we can define the following heuristics that will be used in all of our experiments:

1. *poly:* $F(T,r,t)=f_j$ such that $f_j \in E(T)$ and $\frac{b(T,r,f_j,t)}{a(T,r,f_j,t)+1}$ is maximized.
2. *log:* $F(T,r,t)=f_j$ such that $f_j \in E(T)$ and $\frac{b(T,r,f_j,t)}{\log_2(a(T,r,f_j,t)+2)}$ is maximized.
3. *maxCov:* $F(T,r,t)=f_j$ such that $f_j \in E(T)$, $b(T,r,f_j,t)>0$ and $a(T,r,f_j,t)$ is minimized.
4. *M:* $F(T,r,t)=f_j$ such that $f_j \in E(T)$, $T'=T(f_j,b_j)$ and $M(T',t)$ is minimized.
5. *RM:* $F(T,r,t)=f_j$ such that $f_j \in E(T)$, $T'=T(f_j,b_j)$ and $\frac{M(T',t)}{N(T')}$ is minimized.
6. *me:* $F(T,r,t)=f_j$ such that $f_j \in E(T)$, $T'=T(f_j,b_j)$ and $M(T',mcd(T'))$ is minimized.
7. *mep:* $F(T,r,t)=f_j$ such that $f_j \in E(T)$, $T'=T(f_j,b_j)$ and $\frac{M(T',mcd(T'))}{N(T')}$ is minimized.

We also use some heuristics that can only be calculated for decision tables with single-valued decisions:

8. *ent:* $F(T,r,t)=f_j$ such that $f_j \in E(T)$, $T'=T(f_j,b_j)$ and $\sum_{t \in D(T')} U(T',t)\log_2 U(T',t)$ is maximized.
9. *gin:* $F(T,r,t)=f_j$ such that $f_j \in E(T)$, $T'=T(f_j,b_j)$ and $1-\sum_{t \in D(T')} U(T',t)^2$ is minimized.
10. *RT:* $F(T,r,t)=f_j$ such that $f_j \in E(T)$, $T'=T(f_j,b_j)$ and the number of unordered pairs of rows of T' labeled with different decisions is minimized (note that this value is equal to $N(T)^2(1-\sum_{t \in D(T')} U(T',t)^2)/2$).

There is also one heuristic that we only calculate for decision tables with many-valued decisions:

11. *abs:* $F(T,r,t)=f_j$ such that $f_j \in E(T)$, $T'=T(f_j,b_j)$ and $\prod_{t \in D(T')}(1-U(T',t))$ is minimized.

For the first five heuristic functions (poly, log, maxCov, M, and RM), we apply this algorithm using each decision $t \in D(r)$. As a result, we obtain $|D(r)|$ rules. Depending on our aim, we either choose from among these rules a single rule with minimum length (we denote this algorithm by ⟨name of heuristic⟩_L) or a single rule with maximum coverage (we denote this algorithm by ⟨name of heuristic⟩_C). Note that for decision tables with single-valued decisions $|D(r)| = 1$ so there is no need to append "_C" or "_L" to the heuristic name.

For the rest of the heuristics (6-11), either they only care about $mcd(T)$ or they only apply to decision tables with single-valued decisions. As such, algorithm $\mathscr{A}_5$ is only applied to each row r once using any given $t \in D(r)$ since the resulting rule will be the same regardless of the input decision.

Overall, this results in 10 systems of decision rules for decision tables with single-valued decisions (constructed by heuristics: poly, log, maxCov, M, RM, me, mep, ent, gin, and RT) as well as 13 systems of decision rules for decision tables with many-valued decisions (constructed by heuristics: poly_C, poly_L, log_C, log_L, maxCov_C, maxCov_L, M_C, M_L, RM_C, RM_L, me, mep, and abs).

5 Experimental Results

We carried out our experiments using decision tables from the UCI ML Repository (Lichman 2013). Details regarding these decision tables can be found in Table 1. The number of attributes and rows of each decision table can be found in columns "Attr." and "Rows", respectively. In column "# POPs", we can see the number of Pareto optimal points that exist for systems of decision rules for each table.

In order to obtain decision tables with many-valued decisions, we removed some attributes from the original tables. For example, the "breast-cancer-5" decision table is obtained from the "breast-cancer" dataset by the removal of 5 conditional attributes. The positions of these attributes in the original dataset can be found under the column "Indices of Removed Attr." (this column is left blank for all original decision tables with single-valued decisions). The resulting table contains groups of equal rows possibly with different decisions. We keep a single row from each group and label it with the set of all decisions attached to the rows in its group. As a result, we obtain a decision table with many-valued decisions.

Looking at the data present in Table 1, we can find several interesting facts. Firstly, half of the tables have only a single Pareto optimal point (there exists for each row a rule which simultaneously has both minimum length and maximum coverage). Another observation is that having many Pareto optimal points seems to have a closer correlation to the number of attributes of the table rather than the number of rows. Also, we can clearly see that using tables with many-valued decisions can greatly reduce the number of Pareto optimal points. This could quite possibly be related to reducing the number of attributes.

Our work with greedy heuristics gave interesting results. We decided to measure how well a heuristic performs by three metrics. The first two simply involved comparing the length and coverage, respectively, of the decision rule systems developed by the heuristics. For the third metric, we wanted to compare bi-criteria performance. In this case, we utilized Pareto optimal points. We measured the distance between a greedy solution and the nearest Pareto optimal point to get a good idea of its performance. In order to remove any bias due to differing values for different cost functions (length only varies from 0 to the number attributes while coverage varies from 0 to the number of rows) we normalized the distance along each axis first before calculating the distances. Figure 1 depicts the Pareto optimal points and greedy heuristic results for one of the decision tables with many-valued decisions

Table 1: Details regarding initial and modified tables from UCI ML Repository

Table Name	Attr.	Rows	# POPs	Indices of Removed Attr.
adult-stretch	4	16	1	
balance-scale	4	625	1	
breast-cancer	9	266	204	
breast-cancer-1	8	193	169	3
breast-cancer-5	4	98	16	4,5,6,8,9
cars	6	1728	1	
cars-1	5	432	1	1
flags	26	194	423	
hayes-roth-data	4	69	1	
house-votes	16	279	275	
lenses	4	24	1	
lymphography	18	148	140	
monks1-test	6	432	1	
monks1-train	6	124	5	
monks2-test	6	432	1	
monks2-train	6	169	13	
monks3-test	6	432	1	
monks3-train	6	122	1	
mushroom	22	8124	10822	
mushroom-5	17	4078	3203	5,8,11,13,22
nursery	8	12960	1	
nursery-1	7	4320	1	1
nursery-4	4	240	1	1,5,6,7
shuttle-landing	6	15	4	
soybean-small	35	47	11	
spect-test	22	169	83	
teeth	8	23	1	
teeth-1	7	22	1	1
teeth-5	3	14	1	2,3,4,5,8
tic-tac-toe	9	958	17	
zoo-data	16	59	20	
zoo-data-5	11	42	4	2,9,10,13,14

before normalization. We can see a similar result for a decision table with single-valued decisions in Figure 2.

In order to reduce bias from table sizes and to avoid penalizing heuristics for doing very badly on a few datasets, we can consider using rankings rather than using exact values. Lower ranks are better (1 is the best rank). Furthermore, all heuristics that are tied will receive the same, averaged rank. For example, if two heuristics tie for the first and second rank, then they will both be ranked 1.5 instead. This helps us obtain reasonable results when considering the few but greatly varying datasets from the UCI ML Repository.

In Tables 2 and 3 we can see the ranks of greedy heuristics averaged across all decision tables with many-valued and single-valued decisions, respectively. We can also see the overall rank distinguishing how well each heuristic performed for the given criteria.

One thing of note is that heuristics that perform well for one cost function don't necessarily perform well for another. Looking at Table 2, for example, we can see that "poly_C" and "poly_L" perform very well for coverage, but quite badly for length. Similarly "M_L" and "RM_L" are the best heuristics for length but don't give good results for coverage. Another interesting observation is that the heuristic that performed best for bi-criteria optimization (closest to being Pareto optimal) is "log_L". This heuristic may not be the best for either

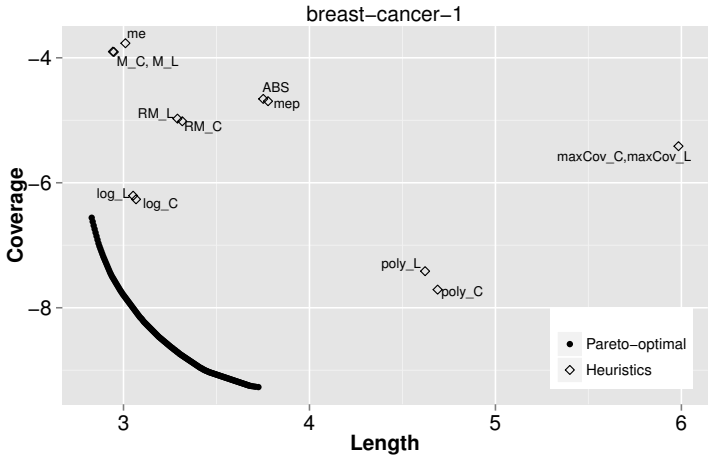

Fig. 1: Pareto optimal points and rule heuristics solutions for a decision table with many-valued decisions

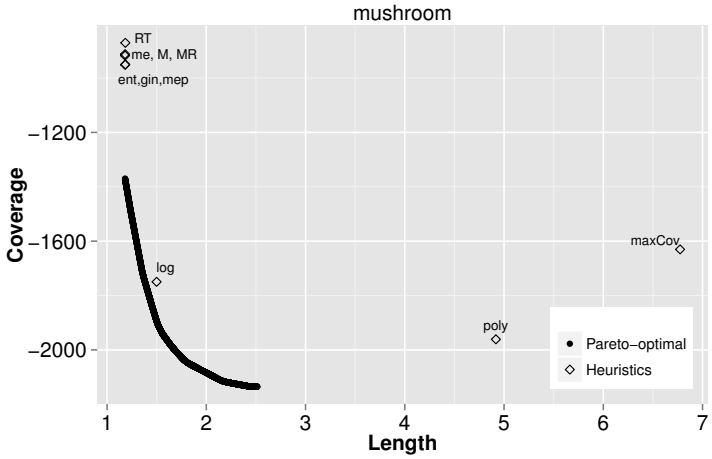

Fig. 2: Pareto optimal points and rule heuristics solutions for a decision table with single-valued decisions

length or coverage, but it gives reasonable results for both at the same time. A similar result can be seen in Table 3, where “log” is the best heuristic from a bi-criteria perspective.

6 Conclusions

In this paper, we developed an algorithm which can find all possible Pareto optimal points for systems of decision rules with respect to coverage and length. Using the results of this algorithm, we were able to compare greedy heuristics to find which ones perform best when considering two criteria at once. We were able to draw some interesting conclusions from

Table 2: Comparing heuristics for decision tables with many-valued decisions

	Average Rank			Overall Rank		
Heuristic Name	Cov	Len	Bi-criteria	Cov	Len	Bi-criteria
poly_C	2.61	7.5	7.39	**1**	10	9
poly_L	3.72	7.17	6.39	**2**	8	5.5
log_C	4.06	5.94	4.61	**3**	7	**3**
log_L	5.17	5.44	3.61	4	5	**1**
maxCov_C	8.17	12.44	12.39	7	13	13
maxCov_L	8.61	12.33	11.17	10	12	12
M_C	8.5	4.61	6.39	8.5	**3**	5.5
M_L	9.61	4.5	6.50	12	**1.5**	7
RM_C	5.78	4.83	4.94	5	4	4
RM_L	6.94	4.5	4.33	6	**1.5**	**2**
ABS	8.78	8.56	8.33	11	11	11
me	10.56	5.83	7.83	13	6	10
mep	8.5	7.33	7.11	8.5	9	8

Table 3: Comparing heuristics for decision tables with single-valued decisions

	Average Rank			Overall Rank		
Heuristic Name	Cov	Len	Bi-criteria	Cov	Len	Bi-criteria
poly	2.22	5.98	3.07	**1**	6	**2**
log	2.61	3.93	2.2	**2**	**3**	**1**
maxCov	5.83	8.67	6.91	5	10	8
ent	5.93	6.09	5.72	6	7	5
gin	6.11	6.48	6	7	8	6
me	8	5.35	7.76	9	5	9
mep	6.39	6.85	6.22	8	9	7
M	5.7	2.98	5.33	4	**1**	4
RM	3.76	3.72	3.63	**3**	**2**	**3**
RT	8.46	4.96	8.17	10	4	10

our work. Firstly, we found that many datasets have only a single Pareto optimal point. We also found that greedy heuristics which perform well for bi-criteria optimization do not necessarily perform well when considering each criterion independently.

Acknowledgements Research reported in this publication was supported by King Abdullah University of Science and Technology (KAUST). We are greatly indebted to the anonymous reviewer for useful comments and suggestions.

References

Amin T, Chikalov I, Moshkov M, Zielosko B (2013) Dynamic programming approach for exact decision rule optimization. In: Rough Sets and Intelligent Systems, Springer Berlin Heidelberg, vol 42, pp 211–228

Azad M, Chikalov I, Moshkov M (2013) Optimization of decision rule complexity for decision tables with many-valued decisions. In: Systems, Man, and Cybernetics, IEEE International Conference on, pp 444–448

Blockeel H, Schietgat L, Struyf J, Džeroski S, Clare A (2006) Decision trees for hierarchical multilabel classification: A case study in functional genomics. In: European Conference on Principles and Practice of Knowledge Discovery in Databases, Springer Berlin Heidelberg, Lecture Notes in Computer Science, vol 4213, pp 18–29

Bonates T, Hammer PL, Kogan A (2008) Maximum patterns in datasets. Discrete Applied Mathematics 156(6):846–861

Boros E, Hammer P, Ibaraki T, Kogan A, Mayoraz E, Muchnik I (2000) An implementation of logical analysis of data. Knowledge and Data Engineering, IEEE Transactions on 12(2):292–306

Bostrom H (1995) Covering vs divide-and-conquer for top-down induction of logic programs. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence, vol 2, pp 1194–1200

Boutell MR, Luo J, Shen X, Brown CM (2004) Learning multi-label scene classification. Pattern Recognition 37(9):1757–1771

Clark P, Niblett T (1989) The cn2 induction algorithm. Machine Learning 3:261–283

Cohen WW, Singer Y (1999) A simple, fast, and effective rule learner. In: Proceedings of the Sixteenth National Conference on Artificial intelligence, American Association for Artificial Intelligence, AAAI '99, pp 335–342

Crama Y, Hammer P, Ibaraki T (1988) Cause-effect relationships and partially defined boolean functions. Annals of Operations Research 16(1):299–325

Dembczyński K, Kotłowski W, Słowiński R (2010) Ender: a statistical framework for boosting decision rules. Data Mining and Knowledge Discovery 21(1):52–90

Fürnkranz J (1999) Separate-and-conquer rule learning. Artificial Intelligence Review 13:3–54

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

Greco S, Matarazzo B, Słowiński R (2001) Rough sets theory for multicriteria decision analysis. European Journal of Operational Research 129(1):1–47

Hammer P, Bonates T (2006) Logical analysis of dataan overview: From combinatorial optimization to medical applications. Annals of Operations Research 148(1):203–225

Hammer PL, Kogan A, Simeone B, Szedmk S (2004) Pareto-optimal patterns in logical analysis of data. Discrete Applied Mathematics 144(12):79–102

Lavrač N, Fürnkranz J, Gamberger D (2010) Explicit feature construction and manipulation for covering rule learning algorithms. In: Advances in Machine Learning I, Springer, Studies in Computational Intelligence, vol 262, pp 121–146

Lichman M (2013) UCI Machine Learning Repository. URL http://archive.ics.uci.edu/ml

Michalski S, Pietrzykowski J (2007) iAQ: A program that discovers rules. AAAI-07 AI Video Competition

Moshkov M (2007) On the class of restricted linear information systems. Discrete Mathematics 307(22):2837–2844

Moshkov M, Chikalov I (2000) On algorithm for constructing of decision trees with minimal depth. Fundam Inform 41(3):295–299

Moshkov M, Zielosko B (2011) Combinatorial Machine Learning, Studies in Computational Intelligence, vol 360. Springer Berlin Heidelberg

Pawlak Z (1991) Rough Sets: Theoretical Aspects of Reasoning About Data. Kluwer Academic Publishers, Bosten, Dordrecht

Pawlak Z, Skowron A (2007) Rough sets and boolean reasoning. Information Sciences 177(1):41–73

Quinlan JR (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann

Rivest RL (1987) Learning decision lists. Machine Learning 2:229–246

Wieczorkowska A, Synak P, Lewis RA, Raś ZW (2005) Extracting emotions from music data. In: Foundations of Intelligent Systems, Springer Berlin Heidelberg, Lecture Notes in Computer Science, vol 3488, pp 456–465

Zhou ZH, Jiang K, Li M (2005) Multi-instance learning based web mining. Applied Intelligence 22(2):135–147

Zielosko B, Chikalov I, Moshkov M, Amin T (2014) Optimization of decision rules based on dynamic programming approach. In: Innovations in Intelligent Machines-4, Springer International Publishing, vol 514, pp 369–392