A Scalable Two Stage Approach to Computing Optimal Decision Sets*

Alexey Ignatiev[1], Edward Lam[1,2], Peter J. Stuckey[1], Joao Marques-Silva[3]

[1]Monash University, Melbourne, Australia
[2]CSIRO Data61, Melbourne, Australia
[3]ANITI, IRIT, CNRS, Toulouse, France
{alexey.ignatiev,edward.lam,peter.stuckey}@monash.edu, joao.marques-silva@irit.fr

Abstract

Machine learning (ML) is ubiquitous in modern life. Since it is being deployed in technologies that affect our privacy and safety, it is often crucial to understand the reasoning behind its decisions, warranting the need for *explainable AI*. Rule-based models, such as decision trees, decision lists, and *decision sets*, are conventionally deemed to be the most interpretable. Recent work uses propositional satisfiability (SAT) solving (and its optimization variants) to generate minimum-size decision sets. Motivated by limited practical scalability of these earlier methods, this paper proposes a novel approach to learn minimum-size decision sets by enumerating individual rules of the target decision set independently of each other, and then solving a set cover problem to select a subset of rules. The approach makes use of modern maximum satisfiability and integer linear programming technologies. Experiments on a wide range of publicly available datasets demonstrate the advantage of the new approach over the state of the art in SAT-based decision set learning.

1 Introduction

Rapid advances in artificial intelligence and, in particular, in machine learning (ML), have influenced all aspects of human lives. Given the practical achievements and the overall success of modern approaches to ML (LeCun, Bengio, and Hinton 2015; Jordan and Mitchell 2015; Mnih et al. 2015; ACM 2018), one can argue that it will prevail as a generic computing paradigm and it will find an ever growing range of practical applications. Unfortunately, the most widely used ML models are opaque, which makes it hard for a human decision-maker to comprehend the outcomes of such models. This motivated efforts on validating the operation of ML models (Ruan, Huang, and Kwiatkowska 2018; Katz et al. 2017) but also on devising approaches to *explainable artificial intelligence* (XAI) (Ribeiro, Singh, and Guestrin 2016; Lundberg and Lee 2017; Monroe 2018).

One of the major lines of work in XAI is devoted to training logic-based models, e.g. decision trees (Bessiere, Hebrard, and O'Sullivan 2009; Narodytska et al. 2018; Hu, Rudin, and Seltzer 2019; Aglin, Nijssen, and Schaus 2020), decision lists (Angelino et al. 2017; Rudin and Ertekin 2018) or decision sets (Lakkaraju, Bach, and Leskovec 2016; Ignatiev et al. 2018; Malioutov and Meel 2018; Ghosh and Meel 2019; Yu et al. 2020), where concise explanations can be obtained directly from the model. This paper focuses on the decision set (DS) model, which comprises an unordered set of *if-then* rules.

One of the advantages of decision sets over other rule-based models is that rule independence makes it straightforward to explain a prediction: a user can pick any rule that "fires" the prediction and the rule itself serves as the explanation. As a result, generation of minimum-size decision sets is of great interest. Recent work proposed SAT-based methods for generating minimum-size decision sets by solving a sequence of problems that determine whether a decision set of size K exists, with K being the number of rules (Ignatiev et al. 2018) or the number of literals (Yu et al. 2020) s.t. the decision set agrees with the training data. Unfortunately, scalability of both approaches is limited due to the large size of the propositional encoding.

Motivated by this limitation, our work proposes a novel approach that splits the DS generation problem into two parts: (1) exhaustive rule enumeration and (2) computing a subset of rules agreeing with the training data. In general, this novel approach enables a significantly more compact propositional encoding, which makes it scalable for problem instances that are out of reach for the state of the art. The proposed approach is inspired by the standard setup used in two-level logic minimization (Quine 1952, 1955; McCluskey 1956). While the first part can be done using enumeration of maximum satisfiability (MaxSAT) solutions, the second part is reduced to the set cover problem, for which integer linear programming (ILP) is effective. Experiments on a wide range of datasets indicate that this approach outperforms the state of the art. The remainder of this paper presents these developments in detail.

2 Preliminaries

Satisfiability and Maximum Satisfiability. We use the standard definitions for propositional satisfiability (SAT)

*This work is supported in part by the AI Interdisciplinary Institute ANITI, funded by the French program "Investing for the Future - PIA3" under Grant agreement no. ANR-19-PI3A-0004.

and maximum satisfiability (MaxSAT) solving (Biere et al. 2009). SAT and MaxSAT admit Boolean variables. A *literal* over Boolean variable x is either the variable x itself or its negation $\neg x$. (Given a constant parameter $\sigma \in \{0,1\}$, we use notation x^σ to represent literal x if $\sigma = 1$, and to represent literal $\neg x$ if $\sigma = 0$.) A *clause* is a disjunction of literals. A *term* is a conjunction of literals. A propositional formula is said to be in *conjunctive normal form* (CNF) or *disjunctive normal form* (DNF) if it is a conjunction of clauses or disjunction of terms, respectively. Whenever convenient, clauses and terms are treated as sets of literals. Formulas written as sets of sets of literals (either in CNF or DNF) are described as *clausal*.

We will make use of partial maximum satisfiability (Partial MaxSAT) (Biere et al. 2009, Chapter 19), which can be formulated as follows. A partial CNF formula can be seen as a conjunction of *hard* clauses $\mathcal{H}$ (which must be satisfied) and *soft* clauses $\mathcal{S}$ (which represent a preference to satisfy those clauses). The Partial MaxSAT problem consists in finding an assignment that satisfies all the hard clauses and maximizes the total number of satisfied soft clauses.

Classification Problems and Decision Sets. We follow the notation used in earlier work (Bessiere, Hebrard, and O'Sullivan 2009; Lakkaraju, Bach, and Leskovec 2016; Ignatiev et al. 2018; Yu et al. 2020). Consider a set of features $\mathcal{F} = \{1,\ldots,K\}$. The domain of possible values for feature $r \in [K]$ is D_r. The complete space of feature values (or *feature space* (Han, Kamber, and Pei 2012)) is $\mathbb{F} \triangleq \prod_{r=1}^{K} D_r$. The vector $\mathbf{f} = (f_1,\ldots,f_K)$ of K variables $f_r \in D_r$ refers to a point in $\mathbb{F}$. Concrete (constant) points in $\mathbb{F}$ are denoted by $\mathbf{v} = (v_1,\ldots,v_K)$, with $v_r \in D_r$. For simplicity, all the features are assumed to be binary, i.e. $D_r = \{0,1\}, \forall r \in [K]$; categorical and ordinal features can be mapped to binary features using standard techniques (Pedregosa et al. 2011). Therefore, whenever convenient, a Boolean *literal* on a feature r can be represented as f_r (or $\neg f_r$, resp.), denoting that feature f_r takes value 1 (value 0, resp.).

Consider a standard classification scenario with training data $\mathcal{E} = \{e_1,\ldots,e_M\}$. A data instance (or *example*) $e_i \in \mathcal{E}$ is a pair $(\mathbf{v}_i, c_i)$ where $\mathbf{v}_i \in \mathbb{F}$ is a vector of feature values and $c_i \in \mathcal{C}$ is a class. An example e_i can be seen as *associating* a vector of feature values $\mathbf{v}_i$ with a class $c_i \in \mathcal{C}$. This work focuses on binary classification problems, i.e. $\mathcal{C} = \{\ominus, \oplus\}$ but the proposed ideas are easily extendable to the case of multiple classes. Given example $e_i = (\mathbf{v}_i, c_i) \in \mathcal{E}$ and the r^{th} component v_{ir} of $\mathbf{v}_i$, literal $f_r^{1-v_{ir}}$ is said to *discriminate* e_i because $f_r^{1-v_{ir}} \triangleq \neg v_{ir}$, i.e. $f_r^{1-v_{ir}}$ falsifies example e_i when it is considered as a conjunction of feature literals. The concept of example discrimination can be extended to terms.

Examples $e_i, e_j \in \mathcal{E}$ associating the same set of feature values with the opposite classes are referred to as *overlapping*. We assume wlog. that the training data $\mathcal{E}$ is *perfectly classifiable*, i.e. $\mathcal{E}$ partially defines a Boolean function $\phi : \mathbb{F} \to \mathcal{C}$ — in other words, there are no overlapping examples in $\mathcal{E}$. Otherwise, either e_i or e_j can be removed from dataset $\mathcal{E}$, incurring an error of 1. In general, repeated *"collisions"* can be resolved by taking the majority vote, which results in highest possible accuracy on training data.

The objective of classification in ML is to devise a function $\hat{\phi}$ that matches the actual function ϕ on the training data $\mathcal{E}$ and generalizes *suitably well* on unseen test data (Fürnkranz, Gamberger, and Lavrac 2012; Han, Kamber, and Pei 2012; Mitchell 1997; Quinlan 1993). In many settings, function $\hat{\phi}$ is not required to match ϕ on the complete set of examples $\mathcal{E}$ and instead an *accuracy* measure is considered. Furthermore, in classification problems one conventionally has to optimize with respect to (1) the complexity of $\hat{\phi}$, (2) the accuracy of the learnt function (to make it match the actual function ϕ on a maximum number of examples), or (3) both. As this paper assumes that the training data does not have overlapping instances, we aim solely at minimizing the representation size of the target ML models.

This paper focuses on learning representations of $\hat{\phi}$ corresponding to *decision sets* (DS) (Lakkaraju, Bach, and Leskovec 2016; Ignatiev et al. 2018; Malioutov and Meel 2018; Ghosh and Meel 2019; Yu et al. 2020). A decision set is an *unordered set* of *rules*. Each rule π is from the set $\mathcal{R} = \prod_{r=1}^{K}\{f_r, \neg f_r, \mathfrak{u}\}$, where $\mathfrak{u}$ represents a *don't care* value. For each example $e \in \mathcal{E}$, a rule of the form $\pi \Rightarrow c$, $\pi \in \mathcal{R}$, $c \in \mathcal{C}$ is interpreted as "if the feature values of example e agree with π then the rule predicts that example e has class c". Hereinafter, we will be dealing with learning minimum-size decision sets, with the size measure being either the number of rules in the decision set or the total number of literals in it (sometimes referred to as *total size*). Note that because rules in decision sets are unordered, some rules may *overlap*, i.e. multiple rules $\pi_i \in \mathcal{R}$ may agree with some instance of the feature space $\mathbb{F}$. It may also happen that *none of the rules* of a decision set apply to some instances of $\mathbb{F}$.

Example 1. Consider the following dataset of four data instances representing the *"to date or not to date?"* example by Domingos (2015):

#	Day	Venue	Weather	TV Show	*Date?*
e_1	Weekday	Dinner	Warm	Bad	No
e_2	Weekend	Club	Warm	Bad	Yes
e_3	Weekend	Club	Warm	Bad	Yes
e_4	Weekend	Club	Cold	Good	No

This data serves to predict whether a friend accepts an invitation to go out for a date given various circumstances. An example of a valid decision set for this data is the following:

IF TV Show = Good **THEN** Date = No
IF Day = Weekday **THEN** Date = No
IF TV Show = Bad ∧ Day = Weekend **THEN** Date = Yes

This DS has 3 rules and a *total size* of 7 (1 for each literal on the left and right, or alternatively, 1 for each literal on the left and 1 for each rule). It does not exhibit rule overlap for examples in $\mathbb{F}$ while the following decision set does:

IF TV Show = Good **THEN** Date = No
IF Day = Weekday **THEN** Date = No
IF Weather = Warm ∧ Day = Weekend **THEN** Date = Yes

Here, the first and third rules overlap for all examples with feature values *Weather* = *Warm* and *TV Show* = *Good*. □

3 Related Work

Rule-based ML models can be traced back to around the 70s and 80s (Michalski 1969; Shwayder 1975; Hyafil and Rivest 1976; Breiman et al. 1984; Quinlan 1986; Rivest 1987). To our best knowledge, decision sets first appear as an unordered variant of decision lists (Rivest 1987; Clark and Niblett 1989) in (Clark and Boswell 1991). The use of logic and optimization for synthesizing a disjunction of rules matching a given training dataset was first tried in (Kamath et al. 1992). Recently, (Lakkaraju, Bach, and Leskovec 2016) argued that decision sets are more interpretable than decision trees and decision lists.

Our work builds on (Ignatiev et al. 2018; Yu et al. 2020) where SAT-based models were proposed for training decision sets of smallest size. The method of (Ignatiev et al. 2018) minimized the number of rules in *perfect* decision sets, i.e. those that agree perfectly with the training data, which is assumed to be consistent; it was also shown to significantly outperform the smooth local search approach of (Lakkaraju, Bach, and Leskovec 2016). Rule minimization was then followed by minimization of the total number of literals used in the decision set, which resulted in a lexicographic approach to the minimization problem. In contrast, (Yu et al. 2020) focused on minimizing the total number of literals in the target DS. This work showed that minimizing the number of rules is more scalable for solving the perfect decision set problem since the optimization measure, i.e. the number of rules, is more coarse-grained. However, minimizing the total number of literals was shown to produce significantly smaller and, thus, more *interpretable* target decision sets. Furthermore, they showed that *sparse* decision sets (minimizing either the number of literals or the number or rules) provide a user with yet another way to produce a succinct classifier representation, by trading off its accuracy for smaller size. Sparse decision sets were also considered in (Malioutov and Meel 2018; Ghosh and Meel 2019) where the authors proposed a MaxSAT model for representing one target class of the training data. ILP was also applied to compute a variant of sparse decision sets (Dash, Günlük, and Wei 2018). As was shown in (Yu et al. 2020), sparse decision sets, although are much easier to compute, achieve lower test accuracy compared to perfect decision sets. As a result, the focus of this work is solely on improving scalability of computing perfect decision sets, i.e. sparse models are excluded from consideration.

4 Decision Sets by Rule Enumeration

Similar to the recent logic-based approaches to learning decision sets (Ignatiev et al. 2018; Malioutov and Meel 2018; Ghosh and Meel 2019), our approach builds on state-of-the-art SAT and MaxSAT technology. These prior works consider a SAT or MaxSAT model that determines whether there exists a decision set of size N given training data $\mathcal{E}$ with $|\mathcal{E}| = M$ examples. The problem is solved by iteratively varying size N and making either a series of SAT calls or one MaxSAT call. The main limitation of prior work is the encoding formula size, which is $\mathcal{O}(N \times M \times K)$, where N is the target size of decision set (which is determined either as the number of rules (Ignatiev et al. 2018) or as the total number of literals (Yu et al. 2020)), M is the number of training data instances and K is the number of features in the training data. This limitation significantly impairs scalability of these approaches and hence restricts their practical applicability.

In contrast to the aforementioned works, the approach detailed below does not aim at devising a decision set in one step and instead consists of two phases. The first phase sequentially enumerates all individual minimal rules given an input dataset. The second phase computes a minimum-size subset of rules (either in terms of the number of rules or the total number of literals in use) that *covers* all the training data instances. This way the approach trades off large encoding size and thus potentially hard SAT oracle calls for computing a complete decision set with a (much) larger number of simpler oracle calls, each computing an individual rule, followed by solving the set cover problem. This algorithmic setup is, in a sense, inspired by the effectiveness of the standard clausal formula minimization approach (Quine 1952, 1955; McCluskey 1956; Brayton et al. 1984; Espresso).

4.1 Decision Sets as DNF Formulas

First, recall that we consider binary classification, i.e. $\mathcal{C} = \{\ominus, \oplus\}$ but the ideas of this section can be easily adapted to multi-class problems, e.g. by using *one-hot encoding* (Pedregosa et al. 2011). Next, let us split the set of training examples $\mathcal{E}$ into the sets of examples $\mathcal{E}_\oplus$ and $\mathcal{E}_\ominus$ for the respective classes s.t. $\mathcal{E} = \mathcal{E}_\oplus \cup \mathcal{E}_\ominus$ and $\mathcal{E}_\oplus \cap \mathcal{E}_\ominus = \emptyset$.

Recall that a decision set is an unordered set of *if-then* rules $\pi \Rightarrow c$, each associating a set of literals $\pi \in \mathcal{R}$, $\mathcal{R} = \prod_{r=1}^{K} \{f_r, \neg f_r, \mathrm{u}\}$, over the feature-values present in rule π with the corresponding class $c \in \{\ominus, \oplus\}$. Following (Ignatiev et al. 2018), observe that each set of literals π in the rule forms a term and so every class $c_i \in \{\ominus, \oplus\}$ in a decision set can be represented logically as a disjunction of terms, each term representing a conjunction of literals in π.

Example 2. Consider our example dataset shown in Example 1. Assume that features *Day*, *Venue*, *Weather*, and *TV Show* are represented with Boolean variables f_1, f_2, f_3, and f_4, respectively. Observe that all the features f_r, $r \in [4]$, in the example are binary, and thus each value for feature f_r can be represented either as literal f_r or literal $\neg f_r$. Let us map the original feature values to $\{0, 1\}$ such that the alphabetically-first value is mapped to 0 while the other is mapped to 1. The classes *No* and *Yes* are mapped to $\ominus$ and $\oplus$, respectively. As a result, our dataset becomes

f_1	f_2	f_3	f_4	c
0	1	1	0	$\ominus$
1	0	1	0	$\oplus$
1	0	1	0	$\oplus$
1	0	0	1	$\ominus$

Using this binary dataset and the first decision set from Example 1 the classes $c = \ominus$ and $c = \oplus$ are represented as the DNF formulas $\phi_\ominus \triangleq (f_4) \vee (\neg f_1)$ and $\phi_\oplus \triangleq (\neg f_4 \wedge f_1)$ $\square$

In this work, we follow (Ignatiev et al. 2018; Yu et al. 2020) and compute minimum-size decision sets in the form of disjunctive representations $\phi_\ominus$ and $\phi_\oplus$ of classes $\ominus$ and

$\oplus$. Even though it is simpler to construct rules for one class when there are only two classes (Malioutov and Meel 2018; Ghosh and Meel 2019), computing both $\phi_\ominus$ and $\phi_\oplus$ achieves better interpretability. Specifically, if both classes are explicitly represented, it is relatively easy to extract explicit and succinct explanations for any class, but this is not the case when only one class is computed. This approach also immediately extends to problems with three or more classes.

Without loss of generality, we focus on computing a disjunctive representation $\phi_\oplus$ for the class $c = \oplus$. The same reasoning can be applied to compute $\phi_\ominus$. The target DNF $\phi_\oplus$ must be consistent with the training data, i.e. every term $\pi \in \phi_\oplus$ must (1) *agree with* at least one example $e_i \in \mathcal{E}_\oplus$ and (2) *discriminate* all examples $e_j \in \mathcal{E}_\ominus$. Furthermore, each term $\pi \in \phi_\oplus$ must be *irreducible*, meaning that any subterm $\pi' \subsetneq \pi$ does not fulfill one of the two conditions above.

4.2 Learning Rules

This section describes the first phase of the proposed approach, namely, how a term π satisfying both of the conditions above can be obtained separately of the other terms. Every term is computed as a MaxSAT solution to a partial CNF formula

$$\psi \triangleq \mathcal{H} \wedge \mathcal{S} \tag{1}$$

with $\mathcal{H}$ and $\mathcal{S}$ being the *hard* and *soft* parts, described below.

Consider two sets of Boolean variables P and N, $|P| = |N| = K$. For every feature f_r, $r \in [K]$, define variables $p_r \in P$ and $n_r \in N$. The idea is inspired by the *dual-rail encoding* (DRE) of propositional formulas (Bryant et al. 1987), e.g. studied in the context of logic minimization (Manquinho et al. 1997; Jabbour et al. 2014). Variables p_r and n_r are referred to as *dual-rail variables*. We assume that $p_r = 1$ iff $f_r = 1$ while $n_r = 1$ iff $f_r = 0$. Moreover, for every feature f_r, $r \in [K]$, a hard clause is added to $\mathcal{H}$ to forbid the feature taking two values at once:

$$\forall_{r\in[K]} \ (\neg p_r \vee \neg n_r) \tag{2}$$

The other combinations of values for p_r and n_r encode the fact that feature r occurs in the target term π positively, negatively, or does not occur at all (when $p_r = n_r = 0$).

The set of soft clauses $\mathcal{S}$ represents a preference to discard the features from a target term π and thus contains a pair of soft unit clauses expressing that preference:

$$\mathcal{S} \triangleq \{(\neg p_r), (\neg n_r) \mid r \in [K]\} \tag{3}$$

By construction of $\mathcal{S}$ and given a MaxSAT solution for (1), the target term π is composed of all features f_r, for which one of the dual-rail variables (either p_r or n_r) is assigned to 1 by the solution, i.e. the corresponding soft clauses are *falsified*.

Discrimination Constraints. Every example $e_j = (\mathbf{v}_j, \ominus)$ from $\mathcal{E}_\ominus$ must be discriminated. This can be enforced by using a clause $(\bigvee_{r\in[K]} f_r^{1-v_{jr}})$, where constant v_{jr} is the value of the r^{th} feature in example e_j. To represent this in the dual-rail formulation, we add the following hard clauses to $\mathcal{H}$:

$$\forall_{j\in[|\mathcal{E}_\ominus|]} \bigvee_{r\in[K]} \delta_{jr}, \tag{4}$$

where δ_{jr} is to be replaced by dual-rail variable p_r if $v_{jr} = 0$ and replaced by the opposite dual-rail variable n_r if $v_{jr} = 1$.

Example 3. Consider instance $e_1 = (f_1 = 0, f_2 = 1, f_3 = 1, f_4 = 0) \in \mathcal{E}_\ominus$ of the running example. To discriminate it, we add a hard clause $(p_1 \vee n_2 \vee n_3 \vee p_4)$. Indeed, to satisfy this clause, we have to pick one of the literals discriminating example e_1, e.g. if $p_1 = 1$ then literal f_1 occurs in term π, which discriminates instance e_1. □

Coverage Constraints. To enforce that every term π covers at least one training instance of $\mathcal{E}_\oplus$, we can use similar reasoning. Observe that a term $\pi \in \mathcal{R}$ covers instance $e_i = (\mathbf{v}_i, c_i) \in \mathcal{E}_\oplus$ iff none of its literals discriminates e_i, i.e. $f_r^{1-v_{jr}} \notin \pi$ for any $r \in [K]$. For each example $e_i = (\mathbf{v}_i, c_i) \in \mathcal{E}_\oplus$, we introduce an auxiliary variable t_i defined by:

$$t_i \leftrightarrow \neg(\bigvee_{r=1}^{K} \delta_{ir}), \tag{5}$$

where δ_{ir} is to be replaced by dual-rail variable p_r if $v_{ir} = 0$ and replaced by the opposite dual-rail variable n_r if $v_{ir} = 1$. Now, variable t_i is true iff term π covers example e_i.

Example 4. Consider instance $e_2 = (f_1 = 1, f_2 = 0, f_3 = 1, f_4 = 0) \in \mathcal{E}_\oplus$ of the running example. Introduce variable $t_2 \leftrightarrow \neg(n_1 \vee p_2 \vee n_3 \vee p_4)$ as shown above. If $t_2 = 1$, the literals in the target term π cannot discriminate example e_2. □

Once auxiliary variables t_i are introduced for each example $e_i \in \mathcal{E}_\oplus$, the hard clause

$$\bigvee\nolimits_{i\in[|\mathcal{E}_\oplus|]} t_i \tag{6}$$

can be added $\mathcal{H}$ to ensure that any term π agrees with at least one of the training data instances.

The overall partial MaxSAT model (1) comprises hard clauses (2), (4), (5), (6) and also soft clauses (3). The number of variables used in the encoding is $\mathcal{O}(K + M)$ while the number of clauses is $\mathcal{O}(K \times M)$. Recall that earlier works proposed encoding with $\mathcal{O}(N \times M \times K)$ variables and clauses, which in some situations makes it hard (or infeasible) to prove optimality of large decision sets.

Example 5. Consider our aim at computing rules for class $\oplus$ in the running example. By applying the DRE, one obtains the formula $\psi = \mathcal{H} \wedge \mathcal{S}$ where

$$\mathcal{H} = \left\{ \begin{array}{c} (\neg p_1 \vee \neg n_1) \wedge (\neg p_2 \vee \neg n_2) \wedge \\ (\neg p_3 \vee \neg n_3) \wedge (\neg p_4 \vee \neg n_4) \wedge \\ \\ (p_1 \vee n_2 \vee n_3 \vee p_4) \wedge \\ (n_1 \vee p_2 \vee p_3 \vee n_4) \wedge \\ \\ [t_2 \leftrightarrow \neg(n_1 \vee p_2 \vee n_3 \vee p_4)] \wedge \\ [t_3 \leftrightarrow \neg(n_1 \vee p_2 \vee n_3 \vee p_4)] \wedge \\ (t_2 \vee t_3) \end{array} \right\}$$

and

$$\mathcal{S} = \left\{ \begin{array}{l} (\neg p_1) \wedge (\neg n_1) \wedge (\neg p_2) \wedge (\neg n_2) \wedge \\ (\neg p_3) \wedge (\neg n_3) \wedge (\neg p_4) \wedge (\neg n_4) \end{array} \right\}$$

□

Any assignment satisfying the hard clauses $\mathcal{H}$ of the dual-rail MaxSAT formula (1) constructed above defines a term π that discriminates all examples of $\mathcal{E}_\ominus$ and covers at least one example of $\mathcal{E}_\oplus$; soft clauses $\mathcal{S}$ ensure minimality of terms π. More importantly, one can exhaustively enumerate all solutions of (1) to compute the set of all such terms (i.e. one can use the standard trick of adding a hard clause *blocking* the previous solution and ask for a new one until no more solutions can be found). Let us refer to this set of terms as $\mathcal{T}_\oplus$. Finally, we claim that as soon as exhaustive solution enumeration for formula (1) is finished, the set of terms $\mathcal{T}_\oplus$ covers every example $e_i \in \mathcal{E}_\oplus$. The rationale is that if sets $\mathcal{E}_\oplus$ and $\mathcal{E}_\ominus$ do not overlap then for any example $e_i \in \mathcal{E}_\oplus$ there is a way to cover it by a term π s.t. all examples of $\mathcal{E}_\ominus$ are discriminated by π. This means that, by construction of (1), for every variable t_i, the hard part $\mathcal{H}$ of the formula has a satisfying assignment assigning $t_i = 1$. (Recall that we assume training data to be *perfectly* classifiable.)

Example 6. Consider our running example. Observe that a valid solution for formula ψ above is $\{p_1, n_4\}$ from which we can extract a term $\pi = (f_1 \land \neg f_4)$ for the target class $c = \oplus$. The term π is added to $\mathcal{T}_\oplus$. Observe that π covers both examples e_2 and e_3 and discriminates examples e_1 and e_4 from class $\ominus$. □

4.3 Smallest Rule Cover

Once the set $\mathcal{T}_\oplus$ of all terms for class $c = \oplus$ is obtained, the next step of the approach is to compute a *smallest size* cover $\phi_\oplus$ of the training examples $\mathcal{E}_\oplus$. Concretely, the problem is to select the smallest size subset $\phi_\oplus$ of $\mathcal{T}_\oplus$ that covers all the training examples. The size can be the either the number of terms used or the total number of literals used in $\phi_\oplus$. Therefore, the problem to solve is essentially the *set cover problem* (Karp 1972). Assume that $|\mathcal{T}_\oplus| = L$ and create a Boolean variable b_j for every term $\pi_j \in \mathcal{T}_\oplus$, indicating that rule j is selected. Also, consider $L \times M'$, $M' = |\mathcal{E}_\oplus|$, Boolean constant values a_{ij} s.t. $a_{ij} = 1$ iff term j covers example i. Then, the problem of computing the cover with the fewest number of terms can be stated as:

$$\text{minimize} \sum_{j=1}^{L} b_j \tag{7}$$

$$\text{subject to} \sum_{j=1}^{L} a_{ij} \cdot b_j \geq 1, \forall i \in [M'] \tag{8}$$

Alternatively, the objective function can be modified to minimize the total number of literals. Concretely, create a constant $s_j \in \mathbb{Z}$ s.t. $s_j = |\pi_j|$, $\pi \in \mathcal{T}_\oplus$. The problem is then to

$$\text{minimize} \sum_{j=1}^{L} s_j \cdot b_j \tag{9}$$

$$\text{subject to} \sum_{j=1}^{L} a_{ij} \cdot b_j \geq 1, \forall i \in [M'] \tag{10}$$

Example 7. For our running example, $|\mathcal{T}_\oplus| = 4$, with terms being:

$$\mathcal{T}_\oplus = \{(f_1 \land f_3), (f_1 \land \neg f_4), (\neg f_2 \land f_3), (\neg f_2, \neg f_4)\}$$

The set cover problem for $c = \oplus$ can be seen as the table:

	π_1	π_2	π_3	π_4
a_{ij}	1	1	1	1
	1	1	1	1
s_j	2	2	2	2

This example has trivial solutions because every term covers all examples of $\mathcal{E}_\oplus$, i.e., every $a_{i,j} = 1$ When minimizing the number of terms using the set cover problem (7) and (8), every optimal solution contains exactly one term. When minimizing the number of literals using the set cover problem (9) and (10), every optimal solution again contains exactly one term, and this term has size 2.

Now, consider class $c = \ominus$; $|\mathcal{T}_\ominus| = 4$, with terms being:

$$\mathcal{T}_\ominus = \{(\neg f_1), (f_2), (\neg f_3), (f_4)\}$$

Then the set cover problem for $c = \ominus$ can be seen as the following table:

	π_1	π_2	π_3	π_4
a_{ij}	1	1	0	0
	0	0	1	1
s_j	1	1	1	1

In our case, one valid solution picks columns π_1 and π_3 as both of them together cover $\mathcal{E}_\ominus$. Thus, when minimizing the number of terms, the fewest number of columns to pick, such that every row has at least one value, is 2, i.e., any optimal solution has cost 2 $(1+1)$. □

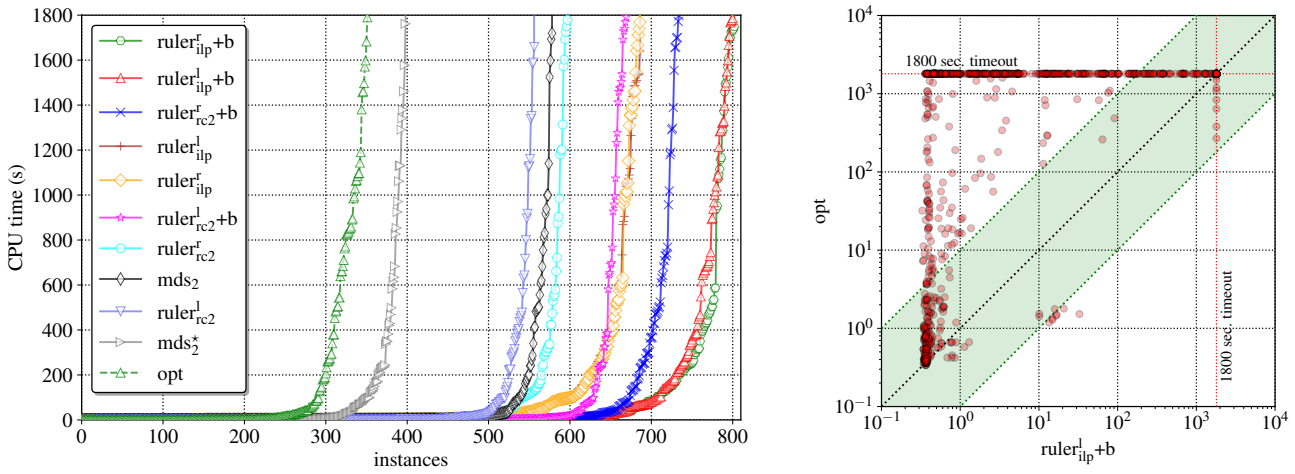
Breaking Symmetric Rules. Observe in Example 7 that the two terms π_1 and π_2 in $\mathcal{T}_\oplus$ cover the same examples from $\mathcal{E}_\oplus$. In the context of the set cover problem, such terms are described as *symmetric*. It is straightforward that at most one term in a set of symmetric terms can appear in a solution because of the minimization in the set cover problem.

Symmetric terms can become an issue if the total number of terms is *exponential* on the number of features. This kind of repetition can be avoided by using the instance coverage variables t_i. Concretely, given a term $\pi \in \mathcal{T}_\oplus$ covering a set $\mathcal{E}'_\oplus \subset \mathcal{E}_\oplus$ of data instances, one can add a clause $(\bigvee_{i \in \mathcal{E}_\oplus \setminus \mathcal{E}'_\oplus} t_i)$ enforcing that any terms discovered later must cover at least one instance e_i uncovered by term π.

While the terms are symmetric for objective (7), there is a dominance relation for objective (9). Consider a term π with the same coverage as another term ρ s.t. $|\rho| \geq |\pi|$ — term π *dominates* term ρ. Given a set cover solution $S \cup \{\rho\}$, we can always replace ρ by π to get a *no worse* solution $S \cup \{\pi\}$. Therefore, term ρ can be ignored during selection.

Because term enumeration is done with MaxSAT, i.e. smaller terms come first, we can use the same method above for symmetry to eliminate dominated terms, since a dominating term (a smallest term with the same coverage) will always be discovered first. Clearly, optimality for objectives (7) and (9) is still guaranteed if breaking symmetric rules is applied.

Example 8. By breaking symmetric terms, the enumeration procedure computes only one term for class $\oplus$ and two terms for class $\ominus$. (Recall that we previously got $|\mathcal{T}_\oplus| = |\mathcal{T}_\ominus| = 4$.) Note that all of them are included in the solutions for the corresponding set cover problems. □



(a) Raw performance

(b) Detailed runtime comparison of $ruler^{l}_{ilp}$ vs. *opt*

Figure 1: Scalability of the competitors.

5 Experimental Results

This section evaluates the proposed rule enumeration based approach in terms of scalability and compares it with the state of the art of SAT-based learning of minimum-size decision sets on a variety of publicly available datasets. The experiments were performed in Debian Linux on an Intel Xeon Silver-4110 2.10GHz processor with 64GByte of memory. Following the setup of recent work (Yu et al. 2020), the time limit was set to 1800s for each individual process to run. The memory limit was set to 8GByte per process.

Prototype Implementation and Selected Competition. A prototype[1] of our rule enumeration based approach was developed as a set of Python scripts, in the following referred to as *ruler*. The implementation of rule enumeration was done with the use of the state-of-the-art MaxSAT solver RC2 (Ignatiev, Morgado, and Marques-Silva 2018, 2019), which proved to be the most effective in MaxSAT model enumeration (MaxSAT Evaluation 2020). As a result, the terms are computed in a sorted fashion, i.e. the smallest ones come first. For the second phase of the approach, i.e. computing the set cover, we attempted to solve the problem both (1) with the RC2 MaxSAT solver and (2) with the Gurobi ILP solver (Gurobi Optimization 2020). The corresponding configurations of the prototype are called $ruler^{*}_{rc2}$ and $ruler^{*}_{ilp}$, where '$*$' can either be *'r'* or *'l'* meaning that the solver minimizes either the number of rules or the total number of literals.[2] Configurations $ruler^{*}_{*}$+b apply symmetry breaking constraints to reduce the number of implicant rules computed in the first phase of the approach. Finally, all configurations were set to compute explicit optimal DNF representations for all classes given a dataset.

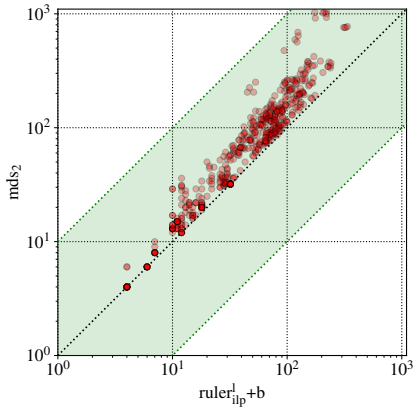
The competiting approaches include SAT-based methods (Ignatiev et al. 2018) $MinDS_2$ and $MinDS^{\star}_2$ referred to as mds_2 and $mds^{\star}_2$. The former tool computes the fewest number of rules while the latter lexicographically minimizes the number of rules and then the number of literals. The second competitor is a recent SAT-based approach that minimizes the total number of literals in the model (Yu et al. 2020), in the following referred to as *opt*. Note that as the main objective of this experimental assessment is to demonstrate *scalability* of the proposed approach, the methods for computing sparse decision sets (Malioutov and Meel 2018; Ghosh and Meel 2019; Yu et al. 2020) are intentionally excluded due to the significant difference in the problem they tackle.

Benchmarks. All datasets considered in the evaluation were adopted from (Yu et al. 2020) and used unchanged. These datasets originated from the UCI Machine Learning Repository (UCI) and the Penn Machine Learning Benchmarks (PennML). The total number of datasets is 1065. The number of *one-hot encoded* (Pedregosa et al. 2011) features (training instances, resp.) per dataset in the benchmark suite varies from 3 to 384 (from 14 to 67557, resp.). Also, since *ruler* can handle only perfectly classifiable data, it processes each training dataset by keeping the largest consistent (non-overlapping) set of examples. This technique is applied in (Ignatiev et al. 2018; Yu et al. 2020) as well, which enables one to achieve the highest possible accuracy on the training data. Motivated by one of the conclusions of (Yu et al. 2020) stating that perfectly accurate decision sets, if successfully computed, are significantly more accurate than sparse and also heuristic models, here we do not compare test accuracy of the competitors – we assume test accuracy to be (close to) identical for all the considered approaches.

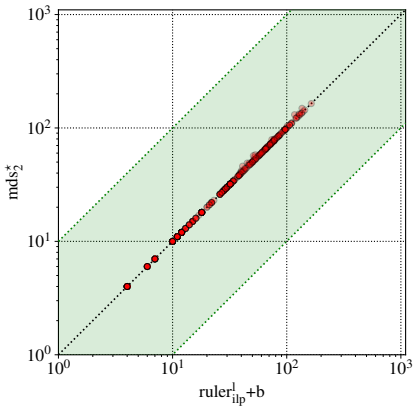
Raw Performance. Figure 1a shows scalability of all the selected approaches. Observe that the mixed solution $ruler^{r}_{ilp}$+b demonstrates the best performance being able to train decision sets for 802 datasets. Second best approach is $ruler^{l}_{ilp}$+b and copes with 800 benchmarks. Pure MaxSAT-based $ruler^{r}_{rc2}$+b and $ruler^{l}_{rc2}$+b perform worse with 734

[1] Available as part of https://github.com/alexeyignatiev/minds.

[2] We also tried using Gurobi for the first phase but it was significantly outperformed by the MaxSAT-based solution.



(a) Literals or rules: $ruler^l_{ilp}$ vs. mds_2



(b) Literals or lexicographic: $ruler^l_{ilp}$ vs. $mds^\star_2$

Figure 2: Model size comparison.

and 669 instances solved. This is not surprising because the structure of set cover problems naturally fits the capabilities of modern ILP solvers.

Disabling symmetry breaking constraints affects the performance of all configurations of $ruler^*_*$, which drops significantly. When it is disabled, the best such configuration ($ruler^l_{ilp}$) solves 686 benchmarks while the worst one ($ruler^l_{rc2}$) tackles 556. Here, we should say that the *maximum* and *average* number of rules enumerated if symmetry breaking is disabled is 326399 and 19604.4, respectively. Breaking symmetric rules decreases these numbers to 8865 and 563.7, respectively.

As for the rivals of the proposed approach, the best of them (mds_2) is far behind $ruler^*_{ilp}$+b and successfully trains 578 models even though it targets rule minimization, which is arguably a much simpler problem. Another competitor ($mds^\star_2$) lexicographically minimizes the number of rules and then the number of literals, which is unsurprisingly harder to deal with, as it solves 398 benchmarks. Finally, the worst performance is demonstrated by *opt*, which learns 351 models. We reemphasize that both *opt* and $ruler^l_{ilp}$+b compute minimum-size decision sets in terms of the number of literals. However, the proposed solution outperforms the competition by 449 benchmarks. Note that due to the large encoding size, *opt* (mds_2, resp.) is practically limited to optimal models having a few dozens of literals (rules, resp.). There is no such limitation in $ruler^*_*$ – in our experiments, it could obtain minimum-size models having thousands of literals in total within the given time limit. The runtime comparison for $ruler^l_{ilp}$+b and *opt* is detailed in Figure 1b – observe that except for a few outliers, $ruler^l_{ilp}$+b outperforms the rival by up to four orders of magnitude.

Rules vs. Literals. Here we demonstrate that literal minimization results in smaller and thus more interpretable models than rule minimization. Figure 2a compares the total number of literals in the models of mds_2 and $ruler^l_{ilp}$+b while Figure 2b compares the model sizes for $mds^\star_2$ and $ruler^l_{ilp}$+b. To make a valid comparison, we used only instances solved by both approaches in each pair. Among the datasets used in of Figure 2a, the average number of literals obtained by mds_2 and $ruler^l_{ilp}$+b is 116.2 and 62.2, respectively – the advantage of literal minimization is clear. Also, as shown in Figure 2b, lexicographic optimization results in models almost identical in size to the models produced by $ruler^l_{ilp}$+b. This suggests that applying the approach of $mds^\star_2$ may in general pay off in terms of solution size, by significantly sacrificing scalability compared to mds_2 and, more importantly, to $ruler^l_{ilp}$+b (398 vs. 578 vs. 800 instances solved, which represents 37.4%, 54.7%, and 75.1% of all 1065 benchmarks, respectively).

6 Conclusions

This paper has introduced a novel approach to learning minimum-size decision sets based on individual rule enumeration. The proposed approach has been motivated by the standard twofold methods applied in two-level logic minimization (Quine 1952, 1955; McCluskey 1956) and split the problem into two parts: (1) exhaustively enumerating individual rules followed by (2) solving the set cover problem. The basic approach has been additionally augmented with symmetry breaking, enabling us to significantly reduce the number of rules produced. The approach has been applied to computing minimum-size decision sets both in terms of the number of rules and in terms of the total number of literals. The proposed approach has been shown to outperform the state of the art in logic-based learning of minimum-size decision sets by a few orders of magnitude.

As the proposed approach targets computing *perfectly accurate* decision sets, a natural line of future work is to examine ways of applying it to computing *sparse* decision sets that trade off accuracy for size. Another line of work is to address the issue of potential rule overlap wrt. the proposed approach. Finally, it is of interest to apply similar rule enumeration techniques for devising other kinds of rule-based ML models, e.g. decision lists and decision trees.

References

ACM. 2018. Fathers of the Deep Learning Revolution Receive ACM A.M. Turing Award. http://tiny.cc/9plzpz.

Aglin, G.; Nijssen, S.; and Schaus, P. 2020. Learning Optimal Decision Trees Using Caching Branch-and-Bound Search. In *AAAI*, 3146–3153.

Angelino, E.; Larus-Stone, N.; Alabi, D.; Seltzer, M.; and Rudin, C. 2017. Learning Certifiably Optimal Rule Lists. In *KDD*, 35–44.

Bessiere, C.; Hebrard, E.; and O'Sullivan, B. 2009. Minimising Decision Tree Size as Combinatorial Optimisation. In *CP*, 173–187.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2009. *Handbook of Satisfiability*. IOS Press. ISBN 978-1-58603-929-5.

Brayton, R. K.; Hachtel, G. D.; McMullen, C.; and Sangiovanni-Vincentelli, A. 1984. *Logic minimization algorithms for VLSI synthesis*, volume 2. Springer Science & Business Media.

Breiman, L.; Friedman, J. H.; Olshen, R. A.; and Stone, C. J. 1984. *Classification and Regression Trees*. Wadsworth. ISBN 0-534-98053-8.

Bryant, R. E.; Beatty, D. L.; Brace, K. S.; Cho, K.; and Sheffler, T. J. 1987. COSMOS: A Compiled Simulator for MOS Circuits. In *DAC*, 9–16.

Clark, P.; and Boswell, R. 1991. Rule Induction with CN2: Some Recent Improvements. In *EWSL*, 151–163.

Clark, P.; and Niblett, T. 1989. The CN2 Induction Algorithm. *Machine Learning* 3: 261–283.

Dash, S.; Günlük, O.; and Wei, D. 2018. Boolean Decision Rules via Column Generation. In *NeurIPS*, 4660–4670.

Domingos, P. 2015. *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World*. Basic Books. ISBN 978-0465065707.

Espresso. 1993. Espresso — Multi-valued PLA minimization. http://tiny.cc/txdnsz.

Fürnkranz, J.; Gamberger, D.; and Lavrac, N. 2012. *Foundations of Rule Learning*. Springer. ISBN 978-3-540-75196-0.

Ghosh, B.; and Meel, K. S. 2019. IMLI: An Incremental Framework for MaxSAT-Based Learning of Interpretable Classification Rules. In *AIES*, 203–210.

Gurobi Optimization, L. 2020. Gurobi Optimizer Reference Manual. URL http://www.gurobi.com.

Han, J.; Kamber, M.; and Pei, J. 2012. *Data Mining: Concepts and Techniques, 3rd edition*. Morgan Kaufmann. ISBN 978-0123814791.

Hu, X.; Rudin, C.; and Seltzer, M. 2019. Optimal Sparse Decision Trees. In *NeurIPS*, 7265–7273.

Hyafil, L.; and Rivest, R. L. 1976. Constructing Optimal Binary Decision Trees is NP-Complete. *Inf. Process. Lett.* 5(1): 15–17.

Ignatiev, A.; Morgado, A.; and Marques-Silva, J. 2018. PySAT: A Python Toolkit for Prototyping with SAT Oracles. In *SAT*, 428–437.

Ignatiev, A.; Morgado, A.; and Marques-Silva, J. 2019. RC2: an Efficient MaxSAT Solver. *J. Satisf. Boolean Model. Comput.* 11(1): 53–64.

Ignatiev, A.; Pereira, F.; Narodytska, N.; and Marques-Silva, J. 2018. A SAT-Based Approach to Learn Explainable Decision Sets. In *IJCAR*, 627–645.

Jabbour, S.; Marques-Silva, J.; Sais, L.; and Salhi, Y. 2014. Enumerating Prime Implicants of Propositional Formulae in Conjunctive Normal Form. In *JELIA*, 152–165.

Jordan, M. I.; and Mitchell, T. M. 2015. Machine learning: Trends, perspectives, and prospects. *Science* 349(6245): 255–260.

Kamath, A. P.; Karmarkar, N.; Ramakrishnan, K. G.; and Resende, M. G. C. 1992. A continuous approach to inductive inference. *Math. Program.* 57: 215–238.

Karp, R. M. 1972. Reducibility Among Combinatorial Problems. In *Complexity of Computer Computations*, 85–103.

Katz, G.; Barrett, C. W.; Dill, D. L.; Julian, K.; and Kochenderfer, M. J. 2017. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In *CAV*, 97–117.

Lakkaraju, H.; Bach, S. H.; and Leskovec, J. 2016. Interpretable Decision Sets: A Joint Framework for Description and Prediction. In *KDD*, 1675–1684.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *nature* 521(7553): 436.

Lundberg, S. M.; and Lee, S. 2017. A Unified Approach to Interpreting Model Predictions. In *NIPS*, 4765–4774.

Malioutov, D.; and Meel, K. S. 2018. MLIC: A MaxSAT-Based Framework for Learning Interpretable Classification Rules. In *CP*, 312–327.

Manquinho, V.; Flores, P.; Marques-Silva, J.; and Oliveira, A. 1997. Prime Implicant Computation Using Satisfiability Algorithms. In *ICTAI*, 232–239.

MaxSAT Evaluation 2020. 2020. MaxSAT Evaluation 2020. https://maxsat-evaluations.github.io/2020/.

McCluskey, E. J. 1956. Minimization of Boolean Functions. *Bell system technical Journal* 35(6): 1417–1444.

Michalski, R. S. 1969. On the quasi-minimal solution of the general covering problem. In *International Symposium on Information Processing*, 125–128.

Mitchell, T. M. 1997. *Machine learning*. McGraw-Hill.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540): 529.

Monroe, D. 2018. AI, Explain Yourself. *Commun. ACM* 61(11): 11–13.

Narodytska, N.; Ignatiev, A.; Pereira, F.; and Marques-Silva, J. 2018. Learning Optimal Decision Trees with SAT. In *IJCAI*, 1362–1368.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12: 2825–2830.

PennML. 2020. Penn Machine Learning Benchmarks. https://github.com/EpistasisLab/penn-ml-benchmarks.

Quine, W. V. 1952. The problem of simplifying truth functions. *American mathematical monthly* 521–531.

Quine, W. V. 1955. A way to simplify truth functions. *American mathematical monthly* 627–631.

Quinlan, J. R. 1986. Induction of Decision Trees. *Machine Learning* 1(1): 81–106.

Quinlan, J. R. 1993. *C4.5: Programs for machine learning*. Morgan Kauffmann.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *KDD*, 1135–1144.

Rivest, R. L. 1987. Learning Decision Lists. *Machine Learning* 2(3): 229–246.

Ruan, W.; Huang, X.; and Kwiatkowska, M. 2018. Reachability Analysis of Deep Neural Networks with Provable Guarantees. In *IJCAI*, 2651–2659.

Rudin, C.; and Ertekin, S. 2018. Learning customized and optimized lists of rules with mathematical programming. *Mathematical Programming Computation* 10: 659–702.

Shwayder, K. 1975. Combining Decision Rules in a Decision Table. *Commun. ACM* 18(8): 476–480.

UCI. 2020. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml.

Yu, J.; Ignatiev, A.; Stuckey, P. J.; and Le Bodic, P. 2020. Computing Optimal Decision Sets with SAT. In *CP*, 952–970.