Optimizing Binary Decision Diagrams with MaxSAT for Classification

Hao Hu, Marie-José Huguet, Mohamed Siala

LAAS-CNRS, Université de Toulouse, CNRS, INSA, Toulouse, France
{hhu, huguet, siala}@laas.fr

Abstract

The growing interest in explainable artificial intelligence (XAI) for critical decision making motivates the need for interpretable machine learning (ML) models. In fact, due to their structure (especially with small sizes), these models are inherently understandable by humans. Recently, several exact methods for computing such models are proposed to overcome weaknesses of traditional heuristic methods by providing more compact models or better prediction quality.

Despite their compressed representation of Boolean functions, Binary decision diagrams (`BDD`s) did not gain enough interest as other interpretable ML models. In this paper, we first propose SAT-based models for learning optimal `BDD`s (in terms of the number of features) that classify all input examples. Then, we lift the encoding to a MaxSAT model to learn optimal `BDD`s in limited depths, that maximize the number of examples correctly classified. Finally, we tackle the fragmentation problem by introducing a method to merge compatible subtrees for the `BDD`s found via the MaxSAT model. Our empirical study shows clear benefits of the proposed approach in terms of prediction quality and interpretability (i.e., lighter size) compared to the state-of-the-art approaches.

Introduction

Due to the increasing concerns in understanding the reasoning behind AI decisions for critical applications, interpretable Machine Learning (ML) models gained a lot of attention. Examples of such ML applications include job recruitment, bank credit applications, and justice (Voigt and Bussche 2017). Most of traditional approaches for building interpretable models are greedy, for example, decision trees (Breiman et al. 1984; Quinlan 1986, 1993), rule lists (Cohen 1995; Clark and Boswell 1991), and decision sets (Lakkaraju, Bach, and Leskovec 2016). Compared to traditional approaches, exact methods offer guarantee of optimality, such as model size and accuracy. In this context, combinatorial optimisation methods, such as Constraint Programming (Bonfietti, Lombardi, and Milano 2015; Verhaeghe et al. 2020), Mixed Integer Programming (Angelino et al. 2018; Verwer and Zhang 2019; Aglin, Nijssen, and Schaus 2020), or Boolean Satisfiablility (SAT) (Bessiere, Hebrard, and O'Sullivan 2009; Narodytska et al. 2018; Avellaneda 2020; Hu et al. 2020; Janota and Morgado 2020; Yu et al. 2020) have been successfully used to learn interpretable models. These declarative approaches are particularly interesting since they offer certain flexibility to handle additional requirements when learning a model.

By providing compact representations for Boolean functions, Binary Decision Diagrams (`BDD`s) (Akers 1978; Moret 1982; Bryant 1986; Knuth 2009) are widely studied for hardware design, model checking, and knowledge representation. In the context of ML, `BDD` could be viewed as an intrepretable model for binary classification. In addition, they were extended for multi-classification, known as *decision graphs* and heuristic methods were proposed in (Oliver 1992; Kohavi 1994; Kohavi and Li 1995; Mues et al. 2004). Moreover, (Ignatov and Ignatov 2017) proposed *decision stream*, a similar topology to `BDD` based on merging *similar* subtrees in each split made in decision trees to improve the generalization. (Oliver 1992; Kohavi 1994) showed that *decision graphs* could avoid the *replication* problem and *fragmentation* problem of decision trees effectively, which `BDD`s also could avoid in binary classification. This fact indicates that generally in the practice of ML, a `BDD` have a smaller size than the corresponding decision tree.

In this paper, we introduce a SAT-based model for learning optimal `BDD`s with the smallest number of features classifying all examples correctly, and a lifted MaxSAT-based model to learn optimal `BDD`s minimizing the classification error. We assume that all `BDD`s are *ordered* and *reduced*[1], the limitation on the depth for a `BDD`, corresponds to the number of features to be selected by our model. To the best of our knowledge, (Cabodi et al. 2021) is the only exact method of learning optimal `BDD`s in the context of ML. The authors proposed a SAT model to learn optimal `BDD`s with the smallest sizes that correctly classify all examples. In their approach, the depth of the `BDD` is not restrained. In fact, it is possible that the constructed `BDD` is small in size (number of nodes) and high in depth. As the `BDD` is *ordered*, this approach could not limit the number of features used, making it not quite comparable with our proposition. Another related work is in (Hu et al. 2020) where the authors consider a MaxSAT model to learn optimal decision trees minimizing

[1] The two notions are defined in the background

the classification error within a limited depth. The usage of the same solving methodology with the same objective function and the depth limit, makes these two MaxSAT models comparable. Finally, in order to increase the scalability of our approach, we propose a heuristic extension based on a simple pre-processing step. For the sake of space, some details are left in a technical report (Hu, Huguet, and Siala 2022).

Technical Background

Classification

Consider a dataset $\mathcal{E} = \{e_q, \ldots, e_M\}$ with M examples. Each example $e_q \in \mathcal{E}$ is characterized by a list of binary features $\mathcal{L}_q = [f_1, \ldots, f_K]$ and a binary target cl_q, representing the class of the example ($cl_q \in \{0, 1\}$). The data set is partitioned into $\mathcal{E}^+$ and $\mathcal{E}^-$, where $\mathcal{E}^+$ (respectfully $\mathcal{E}^-$) is the set of positive (respectifully negative) examples. That is, $cl_q = 1$ iff $e_q \in \mathcal{E}^+$ and $cl_q = 0$ iff $e_q \in \mathcal{E}^-$. We assume that, $\forall 1 \leq q, q' \leq M$, $\mathcal{L}_q = \mathcal{L}_{q'}$ implies $cl_q = cl_{q'}$.

Let ϕ be the function defined by $\phi(\mathcal{L}_q) = cl_q$, $\forall q \in [1, M]$. The classification problem is to compute a function γ (called a *classifier*) that matches as accurately as possible the function ϕ on examples e_q of the training data and generalizes well on unseen test data.

Binary Decision Diagrams

Binary Decision Diagrams (BDDs) are used to provide compact representation of Boolean functions. Let $[x_1, \ldots, x_n]$ be a sequence of of n Boolean variables. A BDD is a rooted, directed, acyclic graph $\mathcal{G}$. The vertex set $\mathcal{V}$ of $\mathcal{G}$ contains two types of vertices. A *terminal* vertex v is associated to a binary value: *value(v)* $\in \{0, 1\}$. A *nonterminal* vertex v, is associated to a Boolean variable x_i and has two children *left(v)*, *right(v)* $\in \mathcal{V}$. In this case, *index(v)* $= i \in \{1, \ldots, n\}$ is the index of the Boolean variable associated to v.

We assume that all BDDs are *ordered* and *reduced*. These two restrictions are widely considered in the literature as they guarantee a *unique* BDD for a given Boolean function. The restriction *ordered* indicates that for any *nonterminal* vertex v, *index(v)* $<$ *index(left(v))* and *index(v)* $<$ *index(right(v))*. The restriction *reduced* indicates that the graph contains no *nonterminal* vertex v with *left(v)* $=$ *right(v)*, nor does it contain distinct *nonterminal* vertices v and v' having isomorphic rooted sub-graphs. Therefore, given an *ordered reduced* BDD $\mathcal{G}$ with *root* v, the associated Boolean function can be recursively obtained with the Shannon expansion process (Shannon 1938).

Let g be a Boolean function defined over a sequence $\mathcal{X} = [x_1, \ldots, x_n]$ of n Boolean variables. The function g can be represented by a *truth table* that lists the 2^n values of all assignments of the n variables. The value of the truth table is therefore associated to a string of 2^n binary values. A truth table β of length 2^n is said to be of order n. A truth table β of order $n > 0$ has the form $\beta_0\beta_1$, where β_0 and β_1 are truth tables of order $n - 1$, and β_0 and β_1 are called *subtables* of β. The *subtables* of *subtables* are also considered to be *subtables*, and a table is considered as a *subtable* of itself. A *bead* of order n is a truth table T of order n that does not have the form $\alpha\alpha$ where α is a subtable of T. The *beads* of g are the *subtables* of its truth table that happen to be *beads*. Proposition 1 from (Knuth 2009) relates truth table and binary decision diagram for the same Boolean function.

Proposition 1. *All vertices in $\mathcal{V}$ of a binary decision diagram $\mathcal{G}$, are in one-to-one correspondence with the beads of the Boolean function g it represents.*

Based on Proposition 1, we can produce the ordered and reduced binary decision diagram of a Boolean function by finding its *beads* and combine its *beads* with its sequence of variables.

Example 1. *Consider the Boolean function from (Knuth 2009): $g_1(x_1, x_2, x_3) = (x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge (x_1 \vee x_3)$. The binary string associated to its truth table β is* 00010111. *The beads of β are* $\{00010111, 0001, 0111, 01, 0, 1\}$.

From Proposition 1, we can draw the BDD *with the beads found, shown as the left part of Figure 1. The dashed (solid) line of each vertex indicates the left (right) child. Then, we can replace the beads by vertices associated with the sequence of Boolean variables. The final binary decision diagram for g_1 is shown as the right part of Figure 1.*

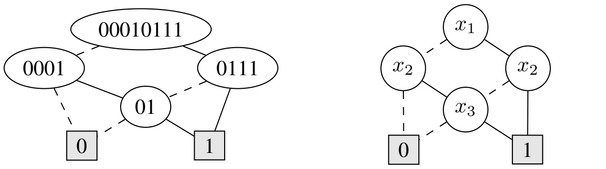

Figure 1: The Binary decision diagram for $g_1(x_1, x_2, x_3)$

Oblivious Read-Once Decision Graphs

Oblivious Read-Once Decision Graphs (OODGs) are proposed in (Kohavi 1994) to overcome some limitations of decision trees for multi-classification, like *replication* and *fragmentation* problem. We refer the readers to (Kohavi and Li 1995; Kohavi 1994) for details on OODGs. An OODG is a rooted, directed, acyclic graph, which contains terminal *category nodes* labelled with classes to make decisions, and non-terminal *branching nodes* labelled with features to make splits. The property "*read-once*" indicates that each feature occurs at most once along any path from the root to a category node. The property "*levelled*" indicates that the nodes are partitioned into a sequence of pairwise disjoint sets, representing the levels, such that outgoing edges from each level terminate at the next level. The property "*oblivious*" extends the idea of "*levelled*" by guaranteeing that all nodes at a given level are labelled by the same feature.

For the classification process, top-down and bottom-up heuristic methods for building OODGs are proposed in (Kohavi and Li 1995; Kohavi 1994). Here, we introduce briefly the top-down heuristic method, which is similar to the heuristic methods C4.5 and CART for computing decision trees. The top-down heuristic induction for OODG with given depth contains three critical phases: (1) selecting a sequence of features with the help of *mutual information* (the difference of *conditional entropy* (Cover and Thomas 2006)); (2)

growing an oblivious decision tree (ODT) by splitting the dataset with features in the sequence selected; and (3) merging *isomorphic* and *compatible* subtrees from top to down to build the OODG. When building the ODT, the algorithm marks nodes that capture no example of the dataset as "*unknwon*". For the merging phase, two subtrees are *compatible* if at least one root is labelled as "*unknown*", or if the two root nodes are labelled with same feature and their corresponding children are the roots of compatible subtrees. The ODT grown could make classifications directly by assigning "*unknown*" nodes with the majority class of their parents.

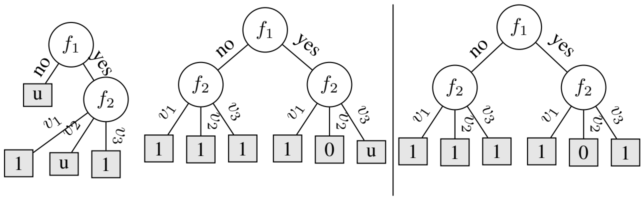

Figure 2: An example of two compatible subtrees (the left two) and the merged tree (the right one) from (Kohavi and Li 1995)

Figure 2 shows an example of two compatible subtrees and the merged tree, where "*unknown*" nodes are labelled as "*u*". Merging compatible subtrees changes the bias by assuming that a "*unknown*" node is likely to behave the same as another child if they belong to compatible subtrees.

In binary classification for binary datasets, OODGs could be considered *equivalent* to BDDs, as the properties "*oblivious*" and "*read-once*" for OODGs are same as property "*ordered*" for BDDs. In addition, the use of merging compatible subtrees could also be applied for BDDs.

SAT and MaxSAT

We use standard terminology for Boolean Satisfiabily (Biere et al. 2009). A *literal* is a Boolean variable or its negation, and a *clause* is a disjunction of literals. An assignment of variables satisfies a clause if one of its literals is true. Given a set of Boolean variables and a set of clauses defined over these variables, the SAT problem can be defined as finding an assignment of the variables such that all the clauses are satisfied. Maximum Satisfiability (MaxSAT) is an optimization version of the SAT problem, where the clauses are partitioned into *hard* and *soft* clauses. Here we consider the Partial MaxSAT problem, that is to find an assignment of the Boolean variables that satisfies all the hard clauses and maximizes the number of satisfied soft clauses.

(Max)SAT-Based Model for Binary Decision Diagrams

In this section, we present our approach for learning BDDs for binary classification using SAT and MaxSAT.

Problem Definition

We firstly consider the following decision problem for classification with BDD in a given depth.

- $P_{bdd}(\mathcal{E}, H)$: *Given a set of examples $\mathcal{E}$, is there a* BDD *of depth H that classifies correctly all examples in $\mathcal{E}$?*

Notice that the algorithm for $P_{bdd}(\mathcal{E}, H)$ can be used to the alternative problem of optimizing a BDD that classifies all examples in the dataset correctly with a minimum depth. For that purpose, one can use a linear search that takes an initial depth H_0 as input and progressively increases or decreases this value depending on the result of solving $P_{bdd}(\mathcal{E}, H)$.

Next, we consider another optimization problem for the classification with BDD in a limited depth.

- $P^*_{bdd}(\mathcal{E}, H)$: *Given a set of examples $\mathcal{E}$, find a* BDD *of depth H that maximises the number of examples in $\mathcal{E}$ that are correctly classified.*

We propose an initial SAT model for the decision problem $P_{bdd}(\mathcal{E}, H)$. Then, we propose an improved version in tighter formula size. Finally, we show how the improved SAT model for $P_{bdd}(\mathcal{E}, H)$ can be used effectively to solve the optimization problem $P^*_{bdd}(\mathcal{E}, H)$ with MaxSAT.

SAT Model for $P_{bdd}(\mathcal{E}, H)$

As shown before, a BDD of depth H could be generated from the combination of a sequence of Boolean variables of size H: $[x_1, \ldots, x_H]$, and a truth table of order H associated to a Boolean function. To solve the classification problem $P_{bdd}(\mathcal{E}, H)$, we then have to find a sequence of binary features of size H that maps one-to-one the sequence of Boolean variables, and a truth table associated to a Boolean function that well-classified all examples. We denote the sequence of binary features found as *feature ordering*. Therefore, the SAT encoding consists of two parts:

- **Part 1:** Constraints for selecting features of the dataset into the feature ordering of size H.
- **Part 2:** Constraints for generating a truth table that classifies all examples of $\mathcal{E}$ correctly with the selected feature ordering.

To realize the SAT encoding, we introduce two sets of Boolean variables as follow:

- a^i_r: the variable a^i_r is 1 iff feature f_r is selected as i-th feature in the feature ordering, where $i = 1, \ldots, H$, $r = 1, \ldots, K$.
- c_j: the variable c_j is 1 iff the j-th value of the truth table is 1, where $j = 1, \ldots, 2^H$.

The set of variables a^i_r guarantees the *ordered* restriction. Then, we introduce two constraints (1) and (2) for the feature ordering. Constraint 1 ensures that any feature f_r can be selected at most once.

$$\sum_{i=1}^{H} a^i_r \leq 1, \quad r = 1, \ldots, K \tag{1}$$

Then, there is exacty one feature selected for each index of the feature ordering.

$$\sum_{r=1}^{K} a^i_r = 1, \quad i = 1, \ldots, H \tag{2}$$

We use the classical sequential counter encoding proposed in (Sinz 2005) to model constraints (1) and (2) as a Boolean formula.

The truth table we are looking for is the binary string of the values of variables $c_1 c_2 \dots c_{2^H}$. To avoid the first feature selected makes useless split, we need to make sure that the truth table is a *bead*.

$$\bigvee_{j=1}^{2^{H-1}} (c_j \oplus c_{j+2^{H-1}}) \tag{3}$$

There is a relationship between the values of a truth table and the assignments of the given sequence of Boolean variables. For example, the first value of a truth table corresponds to the assignment that $x_1 = 0$ and $x_2 = 0$. Therefore, we define the following function to obtain the value of the i-th feature in the feature ordering of size H given the j-th value in the truth table.

$$\mathit{rel}(i,j) = \lfloor \frac{j-1}{2^{H-i}} \rfloor \bmod 2, \quad i \in [1,H], j \in [1,2^H] \tag{4}$$

For an example $e_q \in \mathcal{E}$, we denote the value of the feature f_r as $\sigma(r,q)$. If $\mathit{rel}(i,j) = \sigma(r,q)$, it indicates that for example e_q, the feature f_r can be at the i-th position in the feature ordering to produce the j-th value in the truth table. To classify all examples correctly, we ensure that no example follows an assignment in the truth table leading to its opposite class. Thus, we propose the following constraints for classification. Let $e_q \in \mathcal{E}^+$, for all $j = 1, \dots, 2^H$:

$$\neg c_j \rightarrow \bigvee_{i=1}^{H} \bigvee_{r=1}^{K} (a_r^i \wedge \mathit{rel}(i,j) \oplus \sigma(r,q)) \tag{5}$$

That is, for every positive example e_q, any variable c_j assigned to 0 must be associated to an assignment of features that contains at least one feature-value that is not coherent with e_q. For negative examples, we use a similar idea. Let $e_q \in \mathcal{E}^-$, for all $j = 1, \dots, 2^H$:

$$c_j \rightarrow \bigvee_{i=1}^{H} \bigvee_{r=1}^{K} (a_r^i \wedge \mathit{rel}(i,j) \oplus \sigma(r,q)) \tag{6}$$

$\mathcal{E}_0$	$\boldsymbol{f_1}$	$\boldsymbol{f_2}$	$\boldsymbol{f_3}$	$\boldsymbol{f_4}$	**c**
e_1	1	0	1	0	0
e_2	1	0	0	1	0
e_3	0	0	1	0	1
e_4	1	1	0	0	0
e_5	0	0	0	1	1
e_6	1	1	1	1	0
e_7	0	1	1	0	0
e_8	0	0	1	1	1

Table 1: A binary classification dataset

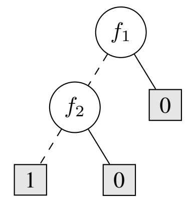

Figure 3: Decision Tree found

Example 2. *Let $\mathcal{E}_0$ be the given set of examples shown in Table 1. Figure 3 shows the corresponding decision tree classifying all examples correctly. We consider to encode a* `BDD` *with depth $H = 2$ classifying all examples of $\mathcal{E}_0$ correctly. The two sets of variables are: $\{a_1^1, a_1^2, a_2^1, a_2^2, a_3^1, a_3^2, a_4^1, a_4^2\}$, and $\{c_1, c_2, c_3, c_4\}$. The constraints 1, 2, and 3 are:*

$$a_1^1 + a_1^2 \leq 1, \quad a_2^1 + a_2^2 \leq 1, \quad a_3^1 + a_3^2 \leq 1, \quad a_4^1 + a_4^2 \leq 1$$
$$a_1^1 + a_2^1 + a_3^1 + a_4^1 = 1, \quad a_1^2 + a_2^2 + a_3^2 + a_4^2 = 1$$
$$(c_1 \oplus c_3) \vee (c_2 \oplus c_4)$$

For classification constraints (i.e., 5 and 6), we show the encoding of $e_1 \in \mathcal{E}^-$ with for value c_1. The encoding for other examples and other values is similar.

$$c_1 \rightarrow (a_1^1 \wedge 0 \oplus 1) \vee (a_2^1 \wedge 0 \oplus 0) \vee (a_3^1 \wedge 0 \oplus 1)$$
$$\vee (a_4^1 \wedge 0 \oplus 0) \vee (a_1^2 \wedge 0 \oplus 1) \vee (a_2^2 \wedge 0 \oplus 0)$$
$$\vee (a_3^2 \wedge 0 \oplus 1) \vee (a_4^2 \wedge 0 \oplus 0)$$

This could be simplified as follow:

$$\neg c_1 \vee a_1^1 \vee a_3^1 \vee a_1^2 \vee a_3^2$$

$x_1 = f_1$	$x_2 = f_2$	
0	0	$c_1 = 1$
0	1	$c_2 = 0$
1	0	$c_3 = 0$
1	1	$c_4 = 0$

Table 2: Truth table solution for `BDD` of depth 2 classifying all example of $\mathcal{E}_0$

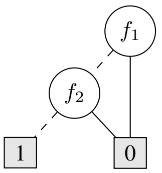

Figure 4: The `BDD` found

The values of truth table found by the SAT model are shown in Table 2, the feature ordering is $[f_1, f_2]$. Moreover, Table 2 illustrates the relationship between the values of truth table and the assignments of the given sequence Boolean variable of size 2. Figure 4 shows the corresponding `BDD`*. This* `BDD` *classifies all examples of the dataset $\mathcal{E}_0$ correctly, also provides more compact representation than the decision tree shown in Figure 3.*

We refer to this first SAT encoding for $P_{bdd}(\mathcal{E}, H)$ as `BDD`1. The size of `BDD`1 is given in Proposition 2.

Proposition 2. *For a $P_{bdd}(\mathcal{E}, H)$ problem with K binary features and M examples, the encoding size (in terms of the number of literals used in the different clauses) of* `BDD1` *is $O(M \times H \times K \times 2^H)$.*

Proof. Notice first that j ranges from 1 to 2^H, i ranges from 1 to H, and r ranges from 1 to K. The term $M \times 2^H$ results from constraint (5) and (6), each contains $O(H \times K)$ literals. For the remaining constraints, it is $O(H \times K)$ for constraints (1) and (2), $O(2^H)$ for constraint (3). □

The size of `BDD`1 is quite huge due to the size of clauses generated by constraints (5) and (6) for classification. This makes `BDD`1 impractical in practice.

An Improved SAT Model for $P_{bdd}(\mathcal{E}, H)$

In order to reduce the size of BDD1, we propose new classification constraints to replace constraints (5) and (6). The idea is that every positive (respectivery negative) example follows an assignment leading to a positive (respectively negative) value of the truth table. We introduce a new set of Boolean variables:

- d_i^q: The variable d_i^q is 1 iff for example e_q the value of the i-th feature selected in feature ordering is 1, where $i = 1, \ldots, H, q = 1, \ldots, M$.

Then, We describe constraints that relate the values of features for each example $e_q \in \mathcal{E}$, for $i = 1, \ldots, H$, $r = 1, \ldots, K$:

$$\begin{aligned} a_r^i &\rightarrow d_i^q \qquad \text{if } \sigma(q, r) = 1 \\ a_r^i &\rightarrow \neg d_i^q \quad \text{if } \sigma(q, r) = 0 \end{aligned} \tag{7}$$

Let $e_q \in \mathcal{E}^+$, we have 2^H constraints for classifying examples correctly:

$$\begin{gathered} \neg d_1^q \wedge \neg d_2^q \wedge \cdots \wedge \neg d_{H-1}^q \wedge \neg d_H^q \rightarrow c_1 \\ \neg d_1^q \wedge \neg d_2^q \wedge \cdots \wedge \neg d_{H-1}^q \wedge d_H^q \rightarrow c_2 \\ \cdots \\ d_1^q \wedge d_2^q \wedge \cdots \wedge d_{H-1}^q \wedge d_H^q \rightarrow c_{2^H} \end{gathered} \tag{8}$$

That is, any positive example follows an assignment of the feature ordering that leads to a positive value in the truth table.

Similarly, for any $e_q \in \mathcal{E}^-$, we also have 2^H constraints:

$$\begin{gathered} \neg d_1^q \wedge \neg d_2^q \wedge \cdots \wedge \neg d_{H-1}^q \wedge \neg d_H^q \rightarrow \neg c_1 \\ \neg d_1^q \wedge \neg d_2^q \wedge \cdots \wedge \neg d_{H-1}^q \wedge d_H^q \rightarrow \neg c_2 \\ \cdots \\ d_1^q \wedge d_2^q \wedge \cdots \wedge d_{H-1}^q \wedge d_H^q \rightarrow \neg c_{2^H} \end{gathered} \tag{9}$$

We refer to this new SAT encoding for $P_{bdd}(\mathcal{E}, H)$ as BDD2. The encoding size of BDD2 is given in Proposition 3.

Proposition 3. *For a $P_{dd}(\mathcal{E}, H)$ problem with K binary features and M examples, the encoding size of the SAT encoding (BDD2) is $O(M \times H \times (2^H + K))$.*

Proof. The term $M \times H \times K$ results from constraint (7). For constraints (8) and (9), for each example, there are 2^H clauses containing $H + 1$ literals. The term $M \times H \times 2^H$ results from that. □

Propositions 2 and 3 show a clear theoretical advantage of BDD2 compared to BDD1 in terms of the encoding size, thus scalability.

MaxSAT Model for $P_{bdd}^*(\mathcal{E}, H)$:

We now present a MaxSAT encoding for the optimization problem $P_{bdd}^*(\mathcal{E}, H)$. That is, given a set of examples $\mathcal{E}$, find a binary decision diagram of depth H that maximises the number of examples correctly classified.

We transform the SAT encoding of BDDs into a MaxSAT encoding following a simple technique. The idea is to keep structural constraints as hard clauses and classification constraints as soft clauses. We consider BDD2 as it has a reduced size. Constraints (1), (2), (3) and (7) are kept as hard clauses. To classify the examples, we declare all clauses of constraints (8) and (9) as soft clauses. For any example e_q, the number of satisfied soft clauses associated to e_q is either 2^H (indicating e_q is classified correctly), or $2^H - 1$ (indicating e_q is classified wrongly). Therefore, the objective of maximising the number of satisfied soft clauses is equivalent to maximise the number of examples that are correctly classified.

Merging Compatible Subtrees

Consider a BDD $\mathcal{G}$ found by a MaxSAT solver and its associated truth table β. Based on the feature ordering of $\mathcal{G}$, it is possible that some values in β capture no (training) example (Equivalent to "*unknown*" nodes for OODG). Such values are decided by the MaxSAT solver in an arbitrary way, which gives a certain bias in generalisation. We propose to merge compatible subtrees in $\mathcal{G}$ in order to handle this bias. This will result in changing some values in the truth table β (i.e. the arbitrary ones decided by MaxSAT).

We propose a post-processing procedure to merge compatible subtrees using the following three phase: (1) update the truth table β by replacing the values of β that capture no examples with a special value "u"; (2) for each level, check the *beads*, where "u" can be used to match 1 or 0, and create a node for each *bead*; (3) for each level, after creating the nodes, check the matches between all subtables of the next level. For matched subtables, update the corresponding *beads* of current level to eliminate the "u" values.

Experimental Results

We present our large experimental study to evaluate empirically our propositions on different aspects[2]. We consider datasets from CP4IM [3]. These datasets are binarized with the one-hot encoding. Preliminary experiments on the decision problem $P_{bdd}(\mathcal{E}, H)$ confirm the great improvements the encoding size of BDD2 compared to BDD1, as shown in propositions 2 and 3. However, solving the optimization problem of finding a BDD that classifies all examples in the dataset correctly with a minimum depth using linear calls to BDD1 and BDD2 was hard. We therefore focus on the optimisation problem $P_{bdd}^*(\mathcal{E}, H)$. This is also motivated by the fact that classifying all examples correctly induces overfitting.

At first, we evaluate the prediction performance between the proposed MaxSAT-BDD model and the heuristic method, ODT and OODG. Next, we compare our model with an exact method for building decision trees using MaxSAT (Hu et al. 2020) in terms of prediction quality, model size, and encoding size. Finally, we propose and evaluate a simple heuristic version of our encoding to tackle scalability. For each dataset, we use random 5-fold cross-validation with 5 different seeds. All experiments were run on a cluster using Xeon E5-2695 v3@2.30GHz CPU and running

[2] The source code and the datasets are available online at https://gitlab.laas.fr/hhu/bddencoding

[3] https://dtai.cs.kuleuven.be/CP4IM/datasets/

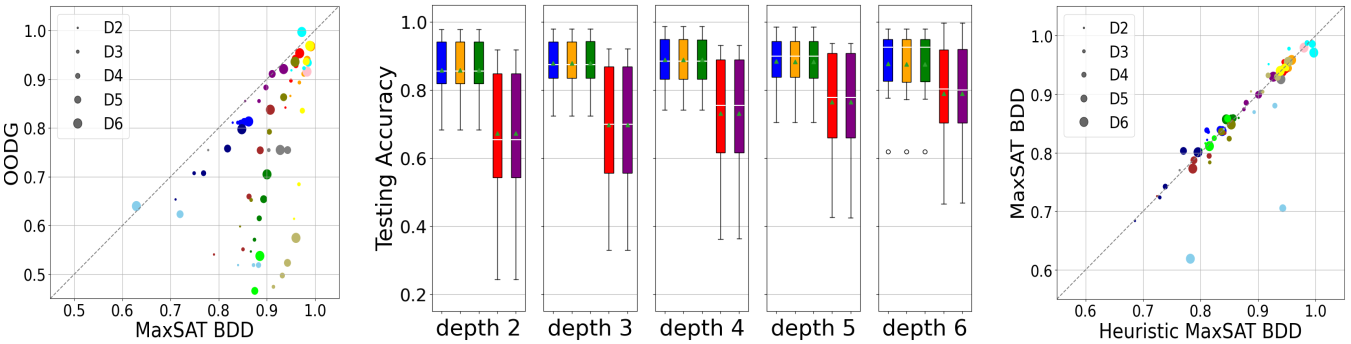

Figure 5: The left scatter shows the average training accuracy of OODG and MaxSAT model. The middle boxplots show the average testing accuracy with different biases: MaxSAT BDD-P(blue), MaxSAT BDD-C(yellow), MaxSAT BDD-S(green), ODT(red), OODG(purple). The right scatter shows the average testing accuracy of MaxSAT-BDD and its heuristic approach.

xUbuntu 16.04.6 LTS. The MaxSAT solver we used is Loandra (Berg, Demirović, and Stuckey 2019), an efficient incomplete MaxSAT solver that return the best solution found within a limited computation time or report optimality. For each experiment, the time limit for generating formulas and the time limit for solver are set to 15 minutes.

Comparison with Existing Heuristic Approaches

We consider the $P^*_{bdd}(\mathcal{E}, H)$ problem with 5 different depths $H \in \{2, 3, 4, 5, 6\}$. We compare our MaxSAT-BDD model with the approach proposed in (Kohavi and Li 1995) to learn ODT and OODG. For the heuristic methods, as described in the background section, after merging *isomorphic* and *compatible* subtrees of ODT, the corresponding OODG changes the bias for those "*unknown*" nodes. In fact, different bias affects the prediction for unseen examples, but *not* for learned examples. Therefore, the training accuracy of ODT and OODG are equals, but the testing accuracy could be different. This fact also suits to the MaxSAT-BDD with different biases for the "*unknown*" nodes. In this experiment, we consider three different biases: assigning for each unknown node the majority class of its branch (denoted as **MaxSAT BDD-P**), merging compatible subtrees (**MaxSAT BDD-C**), and the class decided by the MaxSAT solver (**MaxSAT BDD-S**).

The left scatter plot in Figure 5 presents the comparison of the average training accuracy between OODG and MaxSAT-BDD model. In this figure, different datasets are marked with different colors, and different depths are labelled with points of different sizes. From the scatter plot, we observe that the average training accuracy of both approaches increase with the increase of depth. Overall, the MaxSAT-BDD model performs better than the heuristic OODG in training accuracy.

The middle boxplots of Figure 5 show the average testing accuracy of MaxSAT-BDD with different biases, ODT, and OODG using different depths averaged over all datasets. The white line and green triangle of each box indicate the median and the average value, respectively. Several observations from the boxplots are presented as follows. At first, for each bias, increasing the depth could improve the prediction performance. However, compared to ODT and OODG, all biases chosen for MaxSAT-BDD get less improvements with increasing depths. Next, there are slight differences for different biases of MaxSAT-BDD, indicating that it is quite robust. Then, generally, MaxSAT-BDD gets better prediction performance than ODT and OODG, in particular when depths are small. We noticed also that when the depth is 2, all datasets (except one), MaxSAT-BDD reports optimality.

Comparison with an Exact Decision Tree Approach

The purpose of this experiment is to compare our proposition with the exact method for learning decision trees using the same solving approach (MaxSAT). For MaxSAT-BDD, we consider only the bias of merging compatible subtrees (**MaxSAT BDD-C**) since no substantial difference was observed. We consider different values for depth: $H \in \{2, 3, 4, 5, 6\}$. For MaxSAT-BDD, the depth also corresponds to the number of selected features, whereas for MaxSAT-DT the depth chosen here indicate *maximum depth* of the BDD.

Table 3 presents the results of evaluation. In the column "Dataset", the dataset size (left) and the number of binary feature (right) are shown under the name. The column "Size" and "E_S" indicate the number of nodes of model and the encoding size (number of literals in 100 thousands). The best values are marked in bold.

The results in Table 3 show that the MaxSAT-BDD is competitive to MaxSAT-DT in terms of prediction quality. In most cases, the training and testing accuracy of these two approaches are close. However, the size of the models are always smaller with MaxSAT-BDD. The difference grows bigger when the depth increases. The reduction in model size provides better intrepretability. Moreover, sometimes, compared to the optimal BDDs found via MaxSAT-BDD, the optimal decision trees found via MaxSAT-DT make useless splits improving no prediction performance, as we can see in the case of the datasets "*car*" and "*hypothyroid*" with depth 2. We observe also that MaxSAT-BDD has always a (far) lighter encoding size than MaxSAT-DT which makes it easier to handle and to report optimality in limited time.

Datasets	H	MaxSAT BDD-C				MaxSAT-DT			
		Train	Test	Size	E_S	Train	Test	Size	E_S
anneal (812/89)	2	82.92	**82.19**	**5**	**2.41**	**83.18**	82.14	6.84	5.27
	3	84	83.55	**7**	**3.72**	**85.07**	**84.66**	12.68	12.62
	4	84.58	83.84	**9.4**	**5.21**	**86.05**	**84.78**	18.68	31.55
	5	85.33	83.92	**11.72**	**7.11**	**86.44**	**84.88**	23.88	86.53
	6	86.26	83.70	**14.68**	**9.95**	**87.6**	**85.76**	39.16	266.67
audiology (216/146)	2	94.91	**94.92**	**4**	**1.06**	**95.49**	**94.92**	7	3.14
	3	96.78	**95.84**	**5.04**	**1.64**	**97.82**	95.56	11.56	8.88
	4	97.73	**95.56**	**6.96**	**2.26**	**99.51**	94.54	19.08	27.21
	5	98.40	**94.44**	**9.88**	**2.98**	**99.95**	93.98	27	91.53
	6	99.17	**95.84**	**14.28**	**3.96**	**99.86**	94.08	24.12	332.36
australian (653/124)	2	86.70	**85.94**	**4.72**	**2.68**	**86.93**	85.33	6.68	5.97
	3	87.45	84.81	**5.32**	**4.11**	**88.09**	**84.87**	13.08	14.61
	4	88.45	**86.03**	**7.4**	**5.68**	**88.74**	85.18	17.48	37.76
	5	**89.36**	**85.91**	**10.44**	**7.59**	89.28	84.75	22.52	107.64
	6	**90.05**	**85.7**	**17.32**	**10.25**	89.49	84.84	27.08	343.36
cancer (683/89)	2	93.88	93.59	**4**	**2.03**	**94.91**	**94.2**	7	4.56
	3	95.02	93.91	**5.84**	**3.14**	**96.6**	**94.73**	15	11.09
	4	96.06	**95.49**	**7.96**	**4.39**	**97.34**	94.17	21	28.38
	5	95.94	93.74	**10.68**	**5.99**	**97.99**	**94.35**	29.32	80.09
	6	96.84	**94.35**	**14.8**	**8.38**	**98.87**	93.41	45.72	253.69
car (1728/21)	2	**85.53**	**85.53**	**4**	**1.33**	**85.53**	**85.53**	6.84	3.2
	3	88.40	87.41	**5.08**	**2.2**	**89.25**	**87.45**	12.68	7.18
	4	89.84	88.54	**6.84**	**3.44**	**91.62**	**89.68**	20.36	16.25
	5	91.13	89.91	**9.6**	**5.58**	**93.78**	**92.77**	29.56	39.0
	6	93.51	92.99	**13.36**	**9.71**	**95.8**	**95.06**	31.96	104.45
hypothyroid (3247/86)	2	**97.84**	**97.84**	**4**	**9.27**	**97.84**	**97.84**	5.96	18.22
	3	98.09	**98.04**	**5.12**	**14.28**	**98.14**	97.82	9.72	40.3
	4	98.27	**98.13**	**6.72**	**20.01**	**98.38**	98.01	15.40	88.55
	5	98.30	**98.05**	**9.28**	**27.4**	**98.45**	98	20.04	201.63
	6	98.37	**97.95**	**13.68**	**38.54**	**98.46**	97.91	33.16	495.76
mushroom (8124/112)	2	95.13	95.13	**4**	**29.92**	**96.9**	**96.9**	7	56.53
	3	97.74	97.77	**6.8**	**45.81**	**99.9**	**99.9**	13.72	122.72
	4	98.78	98.74	**9**	**63.51**	**100**	**100**	19.80	260.39
	5	98.63	98.57	**11.32**	**85.37**	**100**	**100**	23.40	557.11
	6	97.28	97.10	**14.6**	**116.59**	**100**	**100**	27.56	1237.69
tic-tac-toe (958/27)	2	71.05	**68.35**	**4**	**0.93**	**71.1**	67.49	5.96	2.23
	3	74.91	72.36	**6.16**	**1.5**	**77.15**	**73.55**	11.48	5.2
	4	76.87	74.22	**8.84**	**2.29**	**82.47**	**78.68**	20.60	12.51
	5	81.86	**80.31**	**13.88**	**3.57**	**83.08**	79.50	28.44	32.83
	6	**84.82**	80.08	**24.16**	**5.95**	84.25	**80.86**	38.12	97.95

Datasets	H	MaxSAT BDD-C				MaxSAT-DT			
		Train	Test	Size	E_S	Train	Test	Size	E_S
cleveland (296/95)	2	79.04	72.57	**4**	**0.95**	**80.76**	**72.84**	7	2.56
	3	85.07	**83.37**	**6**	**1.47**	**85.68**	76.55	12.84	6.89
	4	86.32	**79.46**	**7.84**	**2.06**	**86.77**	76.75	17.80	20.08
	5	**88.65**	**78.72**	**13.08**	**2.79**	87.26	74.45	23.96	64.68
	6	**90.74**	**77.29**	**21.04**	**3.87**	88.58	75.81	28.84	228.48
kr-vs-kp (3196/73)	2	77.83	77.01	**4**	**7.79**	**86.92**	**86.92**	7	15.51
	3	90.43	90.43	**5.28**	**12.05**	**93.81**	**93.79**	12.44	34.3
	4	94.09	94.09	**7.56**	**17.03**	**94.32**	**94.14**	17.24	75.38
	5	94.34	94.18	**9.52**	**23.64**	**94.85**	**94.69**	25.40	171.71
	6	92.80	92.55	**11.52**	**33.94**	**93.91**	**93.69**	29.32	422.77
lymph (148/68)	2	84.46	**83.23**	**4**	**0.35**	**86.01**	79.27	7	1.23
	3	86.76	78.35	**5.92**	**0.56**	**91.93**	**80.54**	14.68	3.67
	4	90.54	**82.4**	**8.72**	**0.79**	**94.56**	78.46	20.20	11.79
	5	93.51	**83.6**	**13.52**	**1.09**	**97.09**	82.46	27.08	41.31
	6	95.88	**84.82**	**17.64**	**1.57**	**99.59**	80.92	46.60	155.03
tumor (336/31)	2	82.80	**81.6**	**4**	**0.37**	**82.92**	81.01	6.76	1.05
	3	83.84	80.43	**5.3**	**0.6**	**86.16**	**82.97**	13.88	2.72
	4	85.52	82.49	**8.64**	**0.9**	**87.89**	**82.85**	20.92	7.64
	5	87.51	**85.83**	**13.32**	**1.38**	**90.1**	79.34	47.80	23.91
	6	88.57	81.12	**19.84**	**2.24**	**90.34**	**81.31**	37.32	83.86
soybean (630/50)	2	90.48	90.48	**4**	**1.08**	**91.27**	**91.27**	7	2.56
	3	91.39	90.41	**6.52**	**1.7**	**95.45**	**94.7**	15	6.23
	4	93.24	93.21	**9.04**	**2.45**	**97.25**	**95.9**	22.20	16.02
	5	94.31	92.95	**11.92**	**3.53**	**97.96**	**95.3**	40.60	45.53
	6	96.07	95.52	**14.88**	**5.34**	**98.27**	**96.03**	33.40	145.99
splice-1 (3190/287)	2	84.04	**84.04**	**4**	**29.66**	**84.22**	83.17	6.92	55.52
	3	87.25	86.94	**5.44**	**44.9**	**87.79**	**87.37**	11.32	123.16
	4	**88.3**	**88.04**	**7.24**	**60.83**	86.52	85.64	16.60	271.79
	5	71.99	70.53	**10.28**	**78.39**	**77.37**	**76.32**	21.88	622.68
	6	**62.92**	**61.89**	**16.28**	**99.63**	60.36	58.95	29.40	1540.61
vote (435/48)	2	95.68	**95.22**	**3.76**	**0.72**	**96.21**	95.03	7	1.83
	3	96.69	**94.57**	**5.56**	**1.14**	**97.39**	93.79	13.96	4.66
	4	97.40	94.39	**8.16**	**1.65**	**98.62**	**94.57**	21.16	12.65
	5	98.21	**94.57**	**12.4**	**2.38**	**99.47**	93.84	30.52	38.2
	6	98.93	93.98	**18.44**	**3.62**	**99.62**	**94.76**	35.40	129.24

Table 3: Comparison Results between MaxSAT-BDD and MaxSAT-DT.

Evaluation of a Heuristic MaxSAT-BDD Method

To increase the scalability of our model, we propose a heuristic version of MaxSAT-BDD using CART (Breiman et al. 1984), an efficient and scalable heuristic for learning decision trees. The idea is to run CART as a pre-processing step in order to choose a subset of (important) features. Then apply the MaxSAT-BDD approach using only the selected subset of features.

We first evaluate the prediction quality of this heuristic method with the original MaxSAT-BDD. The size of the encoding is in favor of the heuristic approach as expected, making large problems treatable. The results of average testing accuracy are shown in the right scatter of Figure 5. This heuristic approach is clearly very competitive to the original MaxSAT-BDD in generalization performance. Especially, for those datasets with more features, the heuristic approach obtains better prediction performance than the original within the same limited time. As a consequence, compared to OODG, our heuristic approach performs much better in terms of prediction quality. We also observed no significant prediction quality difference with CART.

Conclusion

We propose exact and heuristic methods for optimizing binary decision diagrams (BDDs) based on the (Maximum) Boolean Satisfiability framework. Our large experimental studies show clear benefits of the proposed framework in terms of prediction quality and interpretability compared to the existing heuristic approach. Besides, our approach has competitive prediction performance with far lighter encoding size compared to a similar approach for building decition trees.

In the future, it would be interesting to extend the proposed approach for multi-valued classification. Moreover, a deeper investigation of BDDs with other interpretable models (such as decision rules and decision sets) will be beneficial since it enhances alternatives for explainable AI.

References

Aglin, G.; Nijssen, S.; and Schaus, P. 2020. Learning Optimal Decision Trees Using Caching Branch-and-Bound Search. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, 3146–3153. AAAI Press.

Akers. 1978. Binary Decision Diagrams. *IEEE Transactions on Computers*, C-27(6): 509–516.

Angelino, E.; Larus-Stone, N.; Alabi, D.; Seltzer, M.; and Rudin, C. 2018. Learning certifiably optimal rule lists for categorical data. *Journal of Machine Learning Research*, 18(234): 1–78.

Avellaneda, F. 2020. Efficient Inference of Optimal Decision Trees. In *Proceedings of the Thirty-Fourth Conference on Artificial Intelligence (AAAI)*. New York, USA.

Berg, J.; Demirović, E.; and Stuckey, P. J. 2019. Core-Boosted Linear Search for Incomplete MaxSAT. In *Proceedings of the 16th International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research - CPAIOR*, 39–56.

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*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. ISBN 978-1-58603-929-5.

Bonfietti, A.; Lombardi, M.; and Milano, M. 2015. Embedding Decision Trees and Random Forests in Constraint Programming. In Michel, L., ed., *Integration of AI and OR Techniques in Constraint Programming*, 74–90. Cham: Springer International Publishing. ISBN 978-3-319-18008-3.

Breiman, L.; Friedman, J. H.; Olshen, R. A.; ; and Stone, C. J. 1984. *Classification and Regression Trees*. Chapman & Hall/CRC, 1st edition. ISBN 0-534-98053-8.

Bryant, R. E. 1986. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Trans. Computers*, 35(8): 677–691.

Cabodi, G.; Camurati, P. E.; Ignatiev, A.; Marques-Silva, J.; Palena, M.; and Pasini, P. 2021. Optimizing Binary Decision Diagrams for Interpretable Machine Learning Classification. In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2021, Grenoble, France, February 1-5, 2021*, 1122–1125. IEEE.

Clark, P.; and Boswell, R. 1991. Rule induction with CN2: Some recent improvements. In Kodratoff, Y., ed., *Machine Learning — EWSL-91*, 151–163. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-46308-5.

Cohen, W. W. 1995. Fast Effective Rule Induction. In Prieditis, A.; and Russell, S. J., eds., *Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995*, 115–123. Morgan Kaufmann.

Cover, T. M.; and Thomas, J. A. 2006. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. USA: Wiley-Interscience. ISBN 0471241954.

Hu, H.; Huguet, M.; and Siala, M. 2022. Optimizing Binary Decision Diagrams with MaxSAT for classification. *CoRR*, abs/2203.11386.

Hu, H.; Siala, M.; Hebrard, E.; and Huguet, M. 2020. Learning Optimal Decision Trees with MaxSAT and its Integration in AdaBoost. In Bessiere, C., ed., *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, 1170–1176. ijcai.org.

Ignatov, D. Y.; and Ignatov, A. D. 2017. Decision Stream: Cultivating Deep Decision Trees. In *29th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2017, Boston, MA, USA, November 6-8, 2017*, 905–912. IEEE Computer Society.

Janota, M.; and Morgado, A. 2020. SAT-Based Encodings for Optimal Decision Trees with Explicit Paths. In Pulina, L.; and Seidl, M., eds., *Theory and Applications of Satisfiability Testing - SAT 2020 - 23rd International Conference, Alghero, Italy, July 3-10, 2020, Proceedings*, volume 12178 of *Lecture Notes in Computer Science*, 501–518. Springer.

Knuth, D. E. 2009. *The Art of Computer Programming, Volume 4, Fascicle 1: Bitwise Tricks & Techniques; Binary Decision Diagrams*. Addison-Wesley Professional, 12th edition. ISBN 0321580508.

Kohavi, R. 1994. Bottom-Up Induction of Oblivious Read-Once Decision Graphs. In Bergadano, F.; and Raedt, L. D., eds., *Machine Learning: ECML-94, European Conference on Machine Learning, Catania, Italy, April 6-8, 1994, Proceedings*, volume 784 of *Lecture Notes in Computer Science*, 154–169. Springer.

Kohavi, R.; and Li, C. 1995. Oblivious Decision Trees, Graphs, and Top-Down Pruning. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes*, 1071–1079. Morgan Kaufmann.

Lakkaraju, H.; Bach, S. H.; and Leskovec, J. 2016. Interpretable Decision Sets: A Joint Framework for Description and Prediction. In Krishnapuram, B.; Shah, M.; Smola, A. J.; Aggarwal, C. C.; Shen, D.; and Rastogi, R., eds., *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, 1675–1684. ACM.

Moret, B. M. E. 1982. Decision Trees and Diagrams. *ACM Comput. Surv.*, 14(4): 593–623.

Mues, C.; Baesens, B.; Files, C. M.; and Vanthienen, J. 2004. Decision Diagrams in Machine Learning: An Empirical Study on Real-Life Credit-Risk Data. In Blackwell, A. F.; Marriott, K.; and Shimojima, A., eds., *Diagrammatic Representation and Inference, Third International Conference, Diagrams 2004, Cambridge, UK, March 22-24, 2004, Proceedings*, volume 2980 of *Lecture Notes in Computer Science*, 395–397. Springer.

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

Lang, J., ed., *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, 1362–1368. ijcai.org.

Oliver, J. 1992. *Decision graphs: an extension of decision trees*. Citeseer.

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 Kaufmann Publishers Inc. ISBN 1-55860-238-0.

Shannon, C. E. 1938. A symbolic analysis of relay and switching circuits. *Electrical Engineering*, 57(12): 713–723.

Sinz, C. 2005. Towards an Optimal CNF Encoding of Boolean Cardinality Constraints. In van Beek, P., ed., *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*, volume 3709 of *Lecture Notes in Computer Science*, 827–831. Springer.

Verhaeghe, H.; Nijssen, S.; Pesant, G.; Quimper, C.; and Schaus, P. 2020. Learning optimal decision trees using constraint programming. *Constraints An Int. J.*, 25(3-4): 226–250.

Verwer, S.; and Zhang, Y. 2019. Learning optimal classification trees using a binary linear program formulation. In *Proceedings of the Thirty-Third National on Artificial Intelligence, AAAI-19*, 1625–1632. AAAI Press.

Voigt, P.; and Bussche, A. v. d. 2017. *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Springer Publishing Company, Incorporated, 1st edition. ISBN 3319579584.

Yu, J.; Ignatiev, A.; Stuckey, P. J.; and Bodic, P. L. 2020. Computing Optimal Decision Sets with SAT. In Simonis, H., ed., *Principles and Practice of Constraint Programming - 26th International Conference, CP 2020, Louvain-la-Neuve, Belgium, September 7-11, 2020, Proceedings*, volume 12333 of *Lecture Notes in Computer Science*, 952–970. Springer.