Optimizing Binary Decision Diagrams with MaxSAT for Classification
Hao Hu, Marie-José Huguet, Mohamed Siala

To cite this version:
Hao Hu, Marie-José Huguet, Mohamed Siala. Optimizing Binary Decision Diagrams with MaxSAT for Classification. 36th AAAI Conference on Artificial Intelligence, Feb 2022, Vancouver, Canada. hal-03667549

HAL Id: hal-03667549
https://laas.hal.science/hal-03667549
Submitted on 13 May 2022
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 (BDDs) did not gain enough interest as other interpretable ML models. In this paper, we first propose SAT-based models for learning optimal BDDs (in terms of the number of features) that classify all input examples. Then, we lift the encoding to a MaxSAT model to learn optimal BDDs 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 BDDs 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 Satisfiability (SAT) (Bessiere, Hebrard, and O’Sullivan 2009; Narodytska et al. 2018; Avelleda 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 (BDDs) (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 interpretable 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 BDDs 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 BDDs with the smallest number of features classifying all examples correctly, and a lifted MaxSAT-based model to learn optimal BDDs minimizing the classification error. We assume that all BDDs 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 BDDs in the context of ML. The authors proposed a SAT model to learn optimal BDDs 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 func-
tion 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 \( \mathcal{E} = \{e_1, \ldots, e_M\} \) 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 \( c_l_q \), rep-
resenting the class of the example (\( c_l_q \in \{0, 1\} \)). The data set
is partitioned into \( \mathcal{E}^+ \) and \( \mathcal{E}^- \), where \( \mathcal{E}^+ \) (respectfully \( \mathcal{E}^- \))
is the set of positive (respectfully negative) examples. That
is, \( c_l_q = 1 \) iff \( e_q \in \mathcal{E}^+ \) and \( c_l_q = 0 \) iff \( e_q \in \mathcal{E}^- \). We assume that,
\( 1 \leq q, 0 \leq M \), \( \mathcal{L}_q = \mathcal{L}_q' \) implies \( c_l_q = c_l_q' \).

Let \( \phi \) be the function defined by \( \phi(\mathcal{L}_q) = c_l_q \), \( \forall q \in \{1, M\} \).
The classification problem is to compute a function \( \gamma \) (called a classifier)
that matches as accurately as possible the function \( \phi \) 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 com-
 pact representation of Boolean functions. Let \( \{x_1, \ldots, x_n\} \) be a sequence of \( n \) Boolean variables. A BDD is a rooted,
directed, acyclic graph \( G \). The vertex set \( \mathcal{V} \) of \( G \) contains
two types of vertices. A terminal vertex \( v \) is associated to a
binary value: \( \text{value}(v) \in \{0, 1\} \). A nonterminal vertex \( v \),
is associated to a Boolean variable \( x_i \) and has two children
\( \text{left}(v), \text{right}(v) \in \mathcal{V} \). In this case, \( \text{index}(v) = i \in \{1, \ldots, n\} \)
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 func-
tion. The restriction ordered indicates that for any non-
terminal vertex \( v \), \( \text{index}(v) < \text{index}(\text{left}(v)) \) and \( \text{index}(v) \)< \( \text{index}(\text{right}(v)) \). The restriction reduced indicates that the
graph contains no nonterminal vertex \( v \) with \( \text{left}(v) = \text{right}(v) \), 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 Shan-
non 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 \( \beta \) of length \( 2^n \) is said to be of order \( n \). A truth table \( \beta \) of
order \( n > 0 \) has the form \( \beta_0 \beta_1 \), where \( \beta_0 \) and \( \beta_1 \) are truth
tables of order \( n - 1 \), and \( \beta_0 \) and \( \beta_1 \) are called subtables
of \( \beta \). 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 \( \alpha \beta \alpha \) where \( \alpha \) 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 dia-
gram \( \mathcal{G} \), 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): \( g_1(x_1, x_2, x_3) = (x_1 \lor x_2) \land (x_2 \lor x_3) \land (x_1 \lor x_3)
\). The binary string associated to its truth table \( \beta \) is 00010111.
The beads of \( \beta \) 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 se-
quence 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 \( g_1(x_1, x_2, x_3) \)](image)

Oblivious Read-Once Decision Graphs
Oblivious Read-Once Decision Graphs (OODGs) are pro-
posed 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 termi-
nal 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 “obli-
vious” 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 (Ko-
havi and Li 1995; Kohavi 1994). Here, we introduce briefly
the top-down heuristic method, which is similar to the
heuristic methods \( \mathcal{C}_4.5 \) and \( \mathcal{A}_4 EXT \) 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 differ-
ence 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 “unknown”. 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 “t”. 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 Satisfiability (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.

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

Notice that the algorithm for \( P_{\text{bdd}}(E, H) \) 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 \( P_{\text{bdd}}(E, H) \).

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

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

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

**SAT Model for \( P_{\text{bdd}}(E, H) \)**

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 \( P_{\text{bdd}}(E, H) \), 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 \( E \) 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 exactly 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 \(c_1, c_2, \ldots, c_{2^H}\). To avoid the first feature selected makes useless split, we need to make sure that the truth table is a *bead*.

\[
\bigvee_{j=1}^{2^H-1} (c_j \oplus c_{j+2^{H-1}})
\]  

(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.

\[
\text{rel}(i, j) = \left\lfloor \frac{j - 1}{2^{H-1}} \right\rfloor \mod 2, \quad i \in [1, H], j \in [1, 2^H]
\]

(4)

For an example \(e_q \in \mathcal{E}\), we denote the value of the feature \(f_r\) as \(\sigma(r, q)\). If \(\text{rel}(i, j) = \sigma(r, q)\), 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 \(e_q \in \mathcal{E}^+, \text{for all } j = 1, \ldots, 2^H:\)

\[
\neg c_j \rightarrow \bigvee_{i=1}^{H} \bigvee_{r=1}^{K} (a_i^r \land \text{rel}(i, j) \oplus \sigma(r, q))
\]

(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 \(e_q \in \mathcal{E}^-, \text{for all } j = 1, \ldots, 2^H:\)

\[
c_j \rightarrow \bigvee_{i=1}^{H} \bigvee_{r=1}^{K} (a_i^r \land \text{rel}(i, j) \oplus \sigma(r, q))
\]

(6)

The two sets of variables are: \(\{a_1^1, a_2^1, a_2^2, a_3^1, a_3^2, a_3^3\}\) and \(\{c_1, c_2, c_3, c_4\}\). The constraints 1, 2, and 3 are:

\[
a_1^1 + a_2^2 \leq 1, \quad a_1^2 + a_2^3 \leq 1, \quad a_1^3 + a_2^4 \leq 1
\]

\[
a_1^4 + a_2^3 + a_3^1 + a_3^2 = 1
\]

(7)

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

\[
c_1 \rightarrow (a_1^4 \land 0 \lor 1) \lor (a_2^3 \land 0 \lor 0) \lor (a_3^2 \land 0 \lor 1)
\]

\[
\lor (a_1^3 \land 0 \lor 0) \lor (a_2^2 \land 0 \lor 1) \lor (a_3^1 \land 0 \lor 0)
\]

This can be simplified as follow:

\[
\neg c_1 \lor a_1^1 \lor a_3^1 \lor a_1^2 \lor a_3^2
\]

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 \(P_{\text{bdd}}(\mathcal{E}, H)\) as \(\text{BDD}1\). The size of \(\text{BDD}1\) is given in Proposition 2.

**Proposition 2.** For a \(P_{\text{bdd}}(\mathcal{E}, H)\) problem with \(K\) binary features and \(M\) examples, the encoding size (in terms of the number of literals used in the different clauses) of \(\text{BDD}1\) 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 \(\text{BDD}1\) is quite huge due to the size of clauses generated by constraints (5) and (6) for classification. This makes \(\text{BDD}1\) impractical in practice.
An Improved SAT Model for $P_{\text{bdd}}(E, H)$

In order to reduce the size of $\text{BDD}1$, we propose new classification constraints to replace constraints (5) and (6). The idea is that every positive (respectively 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 $e_q \in E$, for $i = 1, \ldots, H$, $r = 1, \ldots, K$:

$$a_r \rightarrow d_i^q \quad \text{if } \sigma(q, r) = 1$$

$$a_r \rightarrow \neg d_i^q \quad \text{if } \sigma(q, r) = 0 \quad (7)$$

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

$$\neg d_i^1 \land \neg d_i^2 \land \cdots \land \neg d_i^{H-1} \land \neg d_i^H \rightarrow c_1$$

$$\neg d_i^1 \land \neg d_i^2 \land \cdots \land \neg d_i^{H-1} \land d_i^H \rightarrow c_2$$

$$\ldots$$

$$d_i^1 \land d_i^2 \land \cdots \land d_i^{H-1} \land d_i^H \rightarrow c_{2^H} \quad (8)$$

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 $e_q \in E^-$, we also have $2^H$ constraints:

$$\neg d_i^1 \land \neg d_i^2 \land \cdots \land \neg d_i^{H-1} \land \neg d_i^H \rightarrow \neg c_1$$

$$\neg d_i^1 \land \neg d_i^2 \land \cdots \land \neg d_i^{H-1} \land d_i^H \rightarrow \neg c_2$$

$$\ldots$$

$$d_i^1 \land d_i^2 \land \cdots \land d_i^{H-1} \land d_i^H \rightarrow \neg c_{2^H} \quad (9)$$

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

**Proposition 3.** For a $P_{\text{bdd}}(E, H)$ problem with $K$ binary features and $M$ examples, the encoding size of the SAT encoding ($\text{BDD2}$) 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 $\text{BDD2}$ compared to $\text{BDD1}$ in terms of the encoding size, thus scalability.

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

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

We transform the SAT encoding of $\text{BDDs}$ 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 $\text{BDD2}$ 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 $\text{BDD} \ G$ found by a MaxSAT solver and its associated truth table $\beta$. Based on the feature ordering of $G$, it is possible that some values in $\beta$ capture no (training) example (Equivalent to “unknown” nodes for $\text{OODG}$). 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 $\ G$ in order to handle this bias. This will result in changing some values in the truth table $\beta$ (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 $\beta$ by replacing the values of $\beta$ 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 subtalebs of the next level. For matched subtalebs, 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. We consider datasets from CP4IM. These datasets are binarized with the one-hot encoding. Preliminary experiments on the decision problem $P_{\text{bdd}}(E, H)$ confirm the great improvements the encoding size of $\text{BDD2}$ compared to $\text{BDD1}$, as shown in propositions 2 and 3. However, solving the optimization problem of finding a $\text{BDD}$ that classifies all examples in the dataset correctly with a minimum depth using linear calls to $\text{BDD1}$ and $\text{BDD2}$ was hard. We therefore focus on the optimisation problem $P_{\text{bdd}}^*(E, H)$. 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-$\text{BDD}$ model and the heuristic method, ODT and $\text{OODG}$. 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

---

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

3https://ditai.cs.kuleuven.be/CP4IM/datasets/
Comparison with Existing Heuristic Approaches

We consider the $P_{\text{add}}(\mathcal{E}, H)$ 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 (fair) lighter encoding size than MaxSAT-DT which makes it easier to handle and to report optimality in limited time.
### Table 3: Comparison Results between MaxSAT-BDD and MaxSAT-DT.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>H</th>
<th>MaxSAT-BDD-C</th>
<th>MaxSAT-DT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train</td>
<td>Test</td>
<td>Size</td>
</tr>
<tr>
<td>anneal</td>
<td>2</td>
<td>82.92</td>
<td>82.19</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>84</td>
<td>83.55</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>84.58</td>
<td>83.84</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>85.33</td>
<td>83.92</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>86.26</td>
<td>83.70</td>
</tr>
<tr>
<td></td>
<td>99.17</td>
<td>94.44</td>
<td>18.28</td>
</tr>
<tr>
<td>audiology</td>
<td>2</td>
<td>94.91</td>
<td>94.92</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>96.78</td>
<td>95.84</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>97.73</td>
<td>95.56</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>98.40</td>
<td>94.44</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>99.17</td>
<td>94.8</td>
</tr>
<tr>
<td>Australian</td>
<td>2</td>
<td>86.70</td>
<td>85.94</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>87.45</td>
<td>84.81</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>88.45</td>
<td>86.03</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>89.36</td>
<td>85.91</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>90.05</td>
<td>85.7</td>
</tr>
<tr>
<td>cancer</td>
<td>2</td>
<td>93.88</td>
<td>93.59</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>95.02</td>
<td>93.91</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>96.06</td>
<td>94.19</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>95.94</td>
<td>93.74</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>96.84</td>
<td>94.35</td>
</tr>
<tr>
<td>car</td>
<td>2</td>
<td>85.53</td>
<td>85.55</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>86.40</td>
<td>87.41</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>89.84</td>
<td>88.54</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>91.13</td>
<td>89.91</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>93.51</td>
<td>92.99</td>
</tr>
<tr>
<td>hypothyroid</td>
<td>2</td>
<td>97.84</td>
<td>97.84</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>98.09</td>
<td>98.04</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>98.27</td>
<td>98.13</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>98.30</td>
<td>98.05</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>98.37</td>
<td>97.95</td>
</tr>
<tr>
<td>mushroom</td>
<td>2</td>
<td>95.13</td>
<td>95.13</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>97.74</td>
<td>97.77</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>98.78</td>
<td>98.84</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>98.63</td>
<td>98.57</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>97.28</td>
<td>97.10</td>
</tr>
<tr>
<td>tic-tac-toe</td>
<td>2</td>
<td>71.05</td>
<td>68.35</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>74.91</td>
<td>72.36</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>76.87</td>
<td>74.22</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>81.86</td>
<td>80.33</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>84.82</td>
<td>80.08</td>
</tr>
</tbody>
</table>

### 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 decision trees.

In the future, it would be interesting to extend the proposed approach for multi-classified values. 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


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


