Learning Optimal Decision Trees with SAT [*]

Nina Narodytska[1], **Alexey Ignatiev**[2,3], **Filipe Pereira**[2], **Joao Marques-Silva**[2]
[1] VMware Research, CA, USA.
[2] LASIGE, Faculty of Science, University of Lisbon
[3] ISDCT SB RAS, Irkutsk, Russia
nnarodytska@vmware.com, {aignatiev,jpms}@ciencias.ulisboa.pt, fcpereira@lasige.di.fc.ul.pt

Abstract

Explanations of machine learning (ML) predictions are of fundamental importance in different settings. Moreover, explanations should be succinct, to enable easy understanding by human decision makers. Decision trees represent an often used approach for developing explainable ML models, motivated by the natural mapping between decision tree paths and rules. Clearly, smaller trees translate directly to smaller rules, and so one challenge is to devise solutions for computing smallest size decision trees given training data. Although simple to formulate, the computation of smallest size decision trees turns out to be an extremely challenging computational problem, for which no practical solutions are known. This paper develops a SAT-based model for computing smallest-size decision trees given training data. In sharp contrast with past work, the proposed SAT model is shown to scale for publicly available datasets of practical interest.

1 Introduction

For a growing number of applications of Machine Learning (ML) and Data Mining (DM) there is a demand to associate explanations with the predictions made, such that these explanations can be interpreted by a human decision maker. The need for explanations associated with ML and DM models in different settings motivates the general area of *eXplainable AI* (XAI). The importance of XAI is illustrated by a range of work in the recent past ([Li *et al.*, 2018; Lakkaraju *et al.*, 2017; Angelino *et al.*, 2017; Lakkaraju *et al.*, 2016; Letham *et al.*, 2015; Lou *et al.*, 2012] among others), by a number of recent events dedicated to the topic [IJCAI XAI Workshop, 2017; ICML WHI Workshop, 2017; NIPS IML Symposium, 2017], but also by recent research programs [DARPA, 2016]. [Biran and Cotton, 2017] provides a recent account of work on XAI. Furthermore, the relevance of XAI is implicit in recent EU-level regulations [EU Data Protection Regulation, 2016], which is motivated in part by a demand for greater transparency in algorithmic decision making [Goodman and Flaxman, 2017; Weller, 2017]. Although XAI can also be seen as controversial in some works [Doshi-Velez and Kim, 2017], it is also apparent that the demand for greater transparency in algorithmic decision making represents one of the main driving forces towards devising explainable ML models.

One approach to achieve explainable ML models is to use models that by construction provide explanations. This is the case with decision trees [Bessiere *et al.*, 2009], but also with rule lists and rule sets [Lakkaraju *et al.*, 2016; Angelino *et al.*, 2017]. Decision trees have been actively investigated since the early 80s [Breiman *et al.*, 1984; Quinlan, 1986; Quinlan, 1993; Mitchell, 1997; Cockett and Hierrera, 1990], and find important practical applications in a wide range of domains, including ML and DM [Quinlan, 1993; Murthy, 1998; Rokach and Maimon, 2015]. Decision trees can naturally associate explanations with predictions made, and so represent a natural choice in the general area of XAI. Indeed, decision trees are by construction capable of providing explanations for predictions made, which are often *easy* to comprehend by human decision makers, *provided* decision trees (and so their paths) are *small* in size. Thus, finding optimal (in size) decision trees can enable computing small-size explanations in XAI settings. We emphases that by optimal decision trees we mean decision trees of the minimum size.

On the negative side, the construction of optimal decision trees is well-known to be NP-hard, for different notions of optimality [Hyafil and Rivest, 1976; Hancock *et al.*, 1996]. As a result, practical algorithms for learning decision trees are heuristic [Breiman *et al.*, 1984; Quinlan, 1986], with no guarantees in terms of size compared to the optimal. One recent exception [Bessiere *et al.*, 2009], based on constraint programming (CP), enables learning optimal decision trees, but existing results provide only approximate solutions. The use of Boolean satisfiability (SAT) for computing optimal decision trees has been shown to scale only for small-size examples [Bessiere *et al.*, 2009], and only for decision tree of fixed structure and size. Not surprisingly, it is generally believed that learning optimal decision trees is impractical except for very small examples [Rokach and Maimon, 2015].

This paper proposes a different take on learning optimal decision trees from examples. The paper identifies a number of fundamental properties of decision trees, which can be

[*]This work was supported by FCT grants SAFETY (SFRH/BPD/120315/2016) and ABSOLV (028986/02/SAICT/2017), and LASIGE Research Unit, ref. UID/CEC/00408/2013.

Ex.	Lecture (L)	Concert (C)	Expo (E)	Shop (S)	Hike (H)
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 classification example.

exploited when developing propositional models for learning decision trees from examples. Furthermore, the paper develops a novel SAT model for deciding the existence of a decision tree consistent with training data, enabling the learning of optimal decision trees. The model is amenable to a number of improvements, but also to a number of variants. The experimental results, all obtained on well-known datasets, for which optimal decision trees were not known, show that the proposed approach can learn optimal decision trees for many representative datasets.

The paper is organized as follows. Section 2 introduces the notation used in the remainder of the paper, and briefly overviews related work. The proposed SAT-based approach for learning optimal decision trees is detailed in Section 3. Section 4 compares the proposed model with earlier work, on a number of datasets. Section 5 concludes the paper.

2 Preliminaries

2.1 Classification problems

We follow the notation used in earlier work [Bessiere *et al.*, 2009]. We consider a set of features $\mathcal{F} = \{f_1, \ldots, f_K\}$, all of which are assumed to be binary, taking a value in $\{0, 1\}$. When necessary, the fairly standard one-hot-encoding is assumed for handling non-binary categorical features. Numeric features can be handled with standard techniques as well. Furthermore, we start from given training data (or examples) $\mathcal{E} = \{e_1, \ldots, e_M\}$. We consider binary classification, and so $\mathcal{E}$ is partitioned into $\mathcal{E}^+$ and $\mathcal{E}^-$, denoting the examples classified as positive and as negative, respectively. Moreover examples are assumed to be consistent, i.e. a set of features is either associated with the positive or negative clause, but not both. Extensions to this basic formulation, including non-binary features, handling of non-binary classes, and allowing for inconsistent examples are beyond the scope of this paper but are discussed in later sections. Furthermore, in this paper we assume that all features are specified for all examples; the work can be generalized for situations where the value of some features for some examples is left unspecified.

Since all features are binary, a literal on a feature f_r will be represented as f_r, denoting that the feature takes value 1, i.e. $f_r = 1$, or $\neg f_r$, denoting that the feature takes value 0, i.e. $f_r = 0$. Moreover, an example $e_q \in \mathcal{E}$ is represented as a 2-tuple $(\mathcal{L}_q, c_q)$, where $\mathcal{L}_q$ denotes the literals associated with the example and $c_q \in \{0, 1\}$ is the class to which the example belongs. We have $c_q = 1$ if $e_q \in \mathcal{E}^+$ and $c_q = 0$ if $e_q \in \mathcal{E}^-$. A literal l_r on a feature f_r, $l_r \in \{f_r, \neg f_r\}$, *discriminates* an example e_q iff $\neg l_r \in \mathcal{L}_q$.

Example 1. *Table 1 shows a simple classification example. Binary features are* $\mathcal{F} = \{f_1, f_2, f_3, f_4\}$ *with* $f_1 \triangleq$ L, $f_2 \triangleq$ C, $f_3 \triangleq$ E, *and* $f_4 \triangleq$ S. *The example* e_1 *is represented by the 2-tuple* $(\mathcal{L}_1, c_1)$, *with* $\mathcal{L}_1 = \{f_1, \neg f_2, f_3, \neg f_4\}$ *and* $c_1 = 0$. *Moreover, the literals* $\neg f_1$, f_2, $\neg f_3$ *and* f_4 *discriminate* e_1.

The objective of classification is to learn some function $\hat{\phi}$ which matches the actual function ϕ on the training data and generalizes *suitably well* on unseen test data. In this paper, we seek to learn representations of $\hat{\phi}$ represented by (binary) *decision trees* (DT). Many other representations have been studied, including rule lists [Angelino *et al.*, 2017], decision sets [Lakkaraju *et al.*, 2017], and sums of terms (i.e. DNF) [Wang *et al.*, 2015; Hauser *et al.*, 2010], among others. These are of interest, including for XAI, but are currently beyond the scope of this work.

2.2 Boolean satisfiability

We assume notation and definitions standard in the area of Boolean Satisfiability (SAT), i.e. the decision problem for propositional logic [Biere *et al.*, 2009]. Formulas are represented in Conjunctive Normal Form (CNF) and defined over a set of variables $X = \{x_1, \ldots, x_n\}$. A formula $\mathcal{F}$ is a conjunction of clauses, a clause is a disjunction of literals, and a literal is a variable x_i or its complement $\neg x_i$. Where appropriate, formulas are viewed as sets of sets of literals. CNF encodings of cardinality constraints have been studied extensively, and will be assumed throughout [Biere *et al.*, 2009]. Moreover, standard clausification techniques are assumed [Plaisted and Greenbaum, 1986].

2.3 Related work

The complexity of learning optimal decision trees is well-known, for different notions of optimality [Hyafil and Rivest, 1976; Hancock *et al.*, 1996]. For example, finding a decision tree that minimizes the average number of tests necessary to classify an example is NP-hard [Hyafil and Rivest, 1976]. Motivated by these results, and also by existing experimental evidence, the learning of optimal decision trees is often deemed infeasible in practice [Rokach and Maimon, 2015].

As a result, all practical approaches for learning decision trees are heuristic [Breiman *et al.*, 1984; Quinlan, 1986; Quinlan, 1993; Mitchell, 1997; Rokach and Maimon, 2015]. These heuristic approaches often learn reasonably small decision trees [1]. Furthermore, most ML and DM packages implement some algorithm for learning decision trees [Hall *et al.*, 2009; Pedregosa and et al., 2011]. One concrete example used in the paper is ITI [Utgoff *et al.*, 1997a] [2].

Despite the computational difficulty of the problem, there are examples of earlier attempts at learning optimal decision trees [Bessiere *et al.*, 2009; Dufour, 2014]. Earlier work considered SAT and constraint programming (CP) [Bessiere *et al.*, 2009] and, more recently what can be viewed as an incomplete variant of A* [Dufour, 2014], based on breadth-first search, with backtracking, and different admissible heuristics. The use of SAT was shown unrealistic by earlier re-

[1]As we show in the results, for larger data sets learnt decision trees may be significantly larger than the optimal size.

[2]The use of ITI is motivated by also being used in earlier related work [Bessiere *et al.*, 2009].

sults [Bessiere *et al.*, 2009]. Moreover, the use of CP was shown to enable obtaining approximate solutions for reasonably sized problems, this at the cost of sacrificing achieving optimal solutions [Bessiere *et al.*, 2009].

3 SAT-Based Learning of Decision Trees

This section proposes a different take on the problem of learning a decision tree using SAT-based methods compared to [Bessiere *et al.*, 2009]. Our encoding exploits a structural property of the learning problem. Namely, one way to learn a decision tree with a given number of nodes is to, first, guess a valid binary tree topology and, second, verify that we can classify all positive and negative examples correctly for this topology. Therefore, our encoding consists of two parts: constraints that encode a valid binary tree and constraints that ensure the decision tree is accurate when classifying examples in the training data. We start with the generation of valid binary trees constraints.

We develop the encoding for a specific number of nodes N of the decision tree, but such that any binary tree of size N can be learned. To search for the optimal value, a wealth of approaches can be considered, which have been studied in the context of SAT-based optimization. In this paper we refine upper bounds on the size of the decision tree, until the formula becomes unsatisfiable for a target size N, in which case the optimal size is $N + 2$ as N must be an odd number.

3.1 Encoding valid binary trees

In this section we consider an encoding of a binary tree for a given number of nodes N. We note that we always build a full binary tree where each node has two children. We assume that nodes are numbered in the breadth-first order from left to right. Any decision tree can be represented with this numbering as we can number nodes using the breadth-first order. Namely, the root node of the tree is numbered 1. The two children of a node i can be numbered in the range from $i + 1$ to $\min(2i + 1, N)$. The number of the left child and the number of the right child are consecutive numbers.

For each node we introduce a propositional variable v_i to encode information about internal nodes and leaves. To encode the child-parent relationship, we introduce three sets of propositional variables, l_{ij}, r_{ij}, p_{ji} (see the upper part of Table 2) Note that l_{ij} and r_{ij} are defined for even/odd indices as a left/right child must be an even/odd node. Moreover, the shortcuts $j \in \mathrm{LR}(i)$ and $j \in \mathrm{RR}(i)$ are also defined in Table 2.

Example 2. *Consider an example of an encoding for a tree with $N = 5$ nodes. We introduce four sets of variables: $\{v_1, \ldots, v_5\}$, $\{l_{12}, l_{2,4}, l_{34}\}$, $\{r_{13}, r_{25}, r_{35}\}$ and $\{p_{21}, p_{31}, p_{32}, p_{42}, p_{43}, p_{52}, p_{53}, p_{54}\}$.*

Next we describe the constraints used. We assume a non-trivial learning problem, and so the root node is not a leaf.

$$(\neg v_1) \tag{1}$$

If a node is a leaf node, then it has no children:

$$v_i \rightarrow \neg l_{ij} \qquad j \in \mathrm{LR}(i) \tag{2}$$

The left child and the right child of the ith node are numbered consecutively.

$$l_{ij} \leftrightarrow r_{ij+1} \qquad j \in \mathrm{LR}(i) \tag{3}$$

A non-leaf node must have a child.

$$\neg v_i \rightarrow \left(\sum_{j \in \mathrm{LR}(i)} l_{ij} = 1 \right) \tag{4}$$

If the ith node is a parent then it must have a child.

$$\begin{array}{ll} p_{ji} \leftrightarrow l_{ij}, & j \in \mathrm{LR}(i) \\ p_{ji} \leftrightarrow r_{ij}, & j \in \mathrm{RR}(i) \end{array} \tag{5}$$

The binary tree must be a tree. Hence, all nodes but the first must have a parent:

$$\left(\sum_{i=\lfloor \frac{j}{2} \rfloor}^{\min(j-1,N)} p_{ji} = 1 \right) \qquad \text{with } j = 2, \ldots, N \tag{6}$$

Note that constraints (4) contain the cardinality constraints. We used the sequential counters encoding [Sinz, 2005] to model a cardinality constraint as a Boolean formula.

Example 3. *We continue with Example 2. We will have the following constraints to encode a binary tree construction. First, we enforce that the root is not a leaf by adding $\neg v_1$. Then we encode constraints from* (2)*: $v_1 \rightarrow \neg l_{12}; v_2 \rightarrow \neg l_{24}; v_3 \rightarrow \neg l_{34}$. Second, we add constraints from* (3) *that left and right nodes are numbered consecutively and a non-leaf must have a child (constraints* (4)*):*

$$l_{1,2} \leftrightarrow r_{13}; l_{24} \leftrightarrow r_{25}; l_{34} \leftrightarrow r_{35};$$
$$\neg v_1 \rightarrow (l_{12} = 1); \neg v_2 \rightarrow (l_{24} = 1); \neg v_3 \rightarrow (l_{34} = 1);$$

Next, we encode parent-child relations using (5) *and* (6)*:*

$$p_{21} \leftrightarrow l_{12}; p_{42} \leftrightarrow l_{24}; p_{43} \leftrightarrow l_{34},$$
$$p_{31} \leftrightarrow r_{13}; p_{52} \leftrightarrow r_{25}; p_{53} \leftrightarrow r_{35},$$
$$p_{21} = 1; p_{31} + p_{32} = 1; p_{42} + p_{43} = 1; p_{52} + p_{53} + p_{54} = 1.$$

As can be seen from this example, the encoding is both compact and easy to implement. There are only two valid binary trees. The first one has two internal nodes: v_1 and v_2. The second one has two internal nodes: v_1 and v_3.

We also ensure a simple approach for breaking some symmetries. Concretely, for any internal node of a binary tree, the left branch will *always* be associated with some feature being assigned value 0, whereas the right branch will *always* be associated with that feature being assigned value 1.

3.2 Computing decision trees with SAT

Given a valid binary tree, each leaf node will be associated with the positive or the negative class. If the node's class is positive, then the path in the tree, and the literals associated with each branch, must *discriminate* all the negative examples; otherwise the classification would not be (100%) accurate. If the node's class is negative, then the path in the tree, and the literals associated with each branch, must discriminate all the positive examples; otherwise the classification would not be (100%) accurate. This section develops the constraints that achieve the target discrimination. (As indicated earlier, it is possible to allow for some examples not to be discriminated, i.e. simply let some examples not to be discriminated, but this is beyond the scope of the paper.)

Besides the variables used to generate valid binary trees, we use additional variables to capture the constraints that the decision tree must correctly classify all examples in $\mathcal{E}$. (The lower part of Table 2 summarizes these variables.) Five addi-

Var	Description of variables
v_i	1 iff node i is a leaf node, $i = 1, \ldots, N$,
l_{ij}	1 iff node i has node j as the left child, with $j \in \textsc{lr}(i)$, where $\textsc{lr}(i) = \mathsf{even}([i+1, \mathsf{min}(2i, N-1)])$, $i = 1, \ldots, N$,
r_{ij}	1 iff node i has node j as the right child, with $j \in \textsc{rr}(i)$, where $\textsc{rr}(i) = \mathsf{odd}([i+2, \mathsf{min}(2i+1, N)])$, $i = 1, \ldots, N$,
p_{ji}	1 iff the parent of node j is node i, $j = 2, \ldots, N$, $i = 1, \ldots, N-1$,
a_{rj}	1 iff feature f_r is assigned to node j, $r = 1, \ldots, K$, $j = 1, \ldots, N$,
u_{rj}	1 iff feature f_r is being discriminated against by node j, $r = 1, \ldots, K$, $j = 1, \ldots, N$,
d^0_{rj}	1 iff feature f_r is discriminated for value 0 by node j, or by one of its ancestors, $r = 1, \ldots, K$, $j = 1, \ldots, N$,
d^1_{rj}	1 iff feature f_r is discriminated for value 1 by node j, or by one of its ancestors, $r = 1, \ldots, K$, $j = 1, \ldots, N$,
c_j	1 iff class of leaf node j is 1, $j = 1, \ldots, N$.

Table 2: Description of propositional variables.

tional sets of variables are used. Variables a_{rj} and u_{rj} encode information about the binary features that was branched on at node j. Variables a_{rj} signal that the rth feature is branched on in node j. Variables u_{rj} store information whether the rth feature is discriminated at any node on the path from this node to the root. Variables d^0_{rj} and d^1_{rj} are used to remember if a feature was discriminated positively or negatively along the path from the root to the jth node. Concretely, any example exhibiting $f_r = 0$ (resp. $f_r = 1$) will be discriminated by node j or by one of its ancestors iff $d^0_{rj} = 1$ (resp. $d^1_{rj} = 1$).

Next we present the constraints used. To discriminate a feature for value 0 at node j, $j = 2, \ldots, N$:

$$d^0_{rj} \leftrightarrow \left(\bigvee_{i=\lfloor \frac{j}{2} \rfloor}^{j-1} \left((p_{ji} \wedge d^0_{ri}) \vee (a_{ri} \wedge r_{ij}) \right) \right); \; d^0_{r,1} = 0. \quad (7)$$

To discriminate a feature for value 1 at node j, $j = 2, \ldots, N$:

$$d^1_{rj} \leftrightarrow \left(\bigvee_{i=\lfloor \frac{j}{2} \rfloor}^{j-1} \left((p_{ji} \wedge d^1_{ri}) \vee (a_{ri} \wedge l_{ij}) \right) \right); \; d^1_{r,1} = 0. \quad (8)$$

Using a feature r at node j, with $r = 1, \ldots, K, j = 1, \ldots, N$:

$$\begin{aligned} &\bigwedge_{i=\lfloor \frac{j}{2} \rfloor}^{j-1} (u_{ri} \wedge p_{ji} \rightarrow \neg a_{rj}) \\ &u_{rj} \leftrightarrow \left(a_{rj} \vee \bigvee_{i=\lfloor \frac{j}{2} \rfloor}^{j-1} (u_{ri} \wedge p_{ji}) \right) \end{aligned} \quad (9)$$

For a non-leaf node j, exactly one feature is used:

$$\neg v_j \rightarrow \left(\sum_{r=1}^{K} a_{rj} = 1 \right) \qquad \text{with } j = 1, \ldots, N \quad (10)$$

For a leaf node j, no feature is used:

$$v_j \rightarrow \left(\sum_{r=1}^{K} a_{rj} = 0 \right) \qquad \text{with } j = 1, \ldots, N \quad (11)$$

Let $e_q \in \mathcal{E}^+$, and let the sign of the literal on feature f_r for e_q be $\sigma(r, q) \in \{0, 1\}$. For every leaf node j, $j = 1, \ldots, N$:

$$v_j \wedge \neg c_j \rightarrow \bigvee_{r=1}^{K} d^{\sigma(r,q)}_{r,j} \quad (12)$$

i.e. any positive example must be discriminated if the leaf node is associated with the negative class.

Let $e_q \in \mathcal{E}^-$, and let the sign of the literal on feature f_r for e_q be $\sigma(r, q)$. Then, for every leaf node j, with $j = 1, \ldots, N$:

$$v_j \wedge c_j \rightarrow \bigvee_{r=1}^{K} d^{\sigma(r,q)}_{r,j} \quad (13)$$

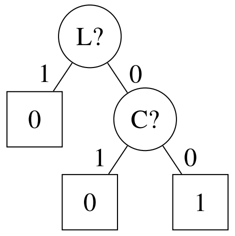

Figure 1: Decision tree for the example in Table 1

i.e. any negative example must be discriminated if the leaf node is associated with the positive class.

Proposition 1 (Propositional model size)**.** *For a DT learning problem with K binary features, $M = |\mathcal{E}|$ examples, and a target decision tree with N nodes, the proposed encoding size (on the number of literals) is in $\mathcal{O}(K \times N^2 + M \times N \times K)$.*

Proof. (Sketch) By inspection of the constraints proposed in this section, noting that r ranges from 1 to K, i, j range from 1 to N, and the the size of $\mathcal{E}$ is M. The term $M \times N$ results from (12) and (13), each contains $O(K)$ literals. The term $K \times N^2$ depends on the remaining constraints. □

Remark 1. *The proposed model is far tighter than the one proposed in [Bessiere* et al.*, 2009] in terms of the size of the SAT encoding, i.e. in $\mathcal{O}(K \times N^2 \times M^2 + N \times K^2 + K \times N^3)$, which exhibits the additional drawback of requiring a fixed target binary tree. Moreover, in practice one would in general expect M to be far larger than either K or N. Thus, in practice the proposed model will in general be orders of magnitude tighter than the model proposed in earlier work [Bessiere* et al.*, 2009]. Our experimental results confirm this observation (see Table 3).*

Example 4. *Figure 1 shows a binary tree of size 5 that classifies examples from Table 1 correctly. We observe that this tree is one of two valid binary trees (see Example 3).* (7)– (13) *rule out the second valid binary tree as it does not ensure correct classification of examples.*

3.3 Additional inference constraints

We propose additional constraints that aim at pruning the search space, by filtering as soon as possible tree arrangements that are invalid. During the search, a partial structure of the tree is constructed. Hence, we can reduce the search space by pruning invalid extensions of this partial tree. We recall that, in general, for a node i, the left child ranges from $i + 1$ to $\mathsf{min}(2i, N - 1)$, among even-numbered nodes, and the right child ranges from $i+2$ to $\min(2i+1, N)$, among odd-numbered nodes.

For each k, with $k < i$, such that v_k holds, then the upper bound on the number of left and right children of i is reduced. Likewise, if v_k does not hold, then the lower bound on the number of left and right children of i is increased. The following example demonstrates our intuition.

Example 5. *Suppose we are looking for a binary tree of size $N = 7$. We focus on the third node, $i = 3$. In general, nodes $4, 5, 6$ or 7 can be children of this node. If we have a balanced binary tree then nodes 6 and 7 are children of node 3. If we have highly a unbalanced binary tree where each left branch is of length one then nodes 4 and 5 are children of node 3.*

Suppose that during the search we learn that there exists (at least) one node that is a leaf and it is numbered 3 or less. In this case, node 2 is a leaf and we conclude that only nodes 4 and 5 can be children of node 3.

Let $\lambda_{t,i}$ denote the number of leaf nodes until (and including) node i. Clearly, $0 \leq t \leq \lfloor \frac{i}{2} \rfloor$. $\lambda_{t,i}$ is defined inductively:

1. $\lambda_{0,i} = 1$, for $1 \leq i \leq N$.
2. $\lambda_{t,i} \leftrightarrow (\lambda_{t,i-1} \vee \lambda_{t-1,i-1} \wedge v_i)$,$i = 1..N, t = 1..\lfloor \frac{i}{2} \rfloor$.

Similarly, we can define the number of non-leaf (tree) nodes until node i, $\tau_{t,i}$, with $0 \leq t \leq i$. Again, $\tau_{t,i}$ is defined inductively as follows:

1. $\tau_{0,i} = 1$, for $1 \leq i \leq N$.
2. $\tau_{t,i} \leftrightarrow (\tau_{t,i-1} \vee \tau_{t-1,i-1} \wedge \neg v_i)$, for $i = 1..N, t = 1..i$.

Proposition 2 (Refine *upper bound* on descendants' numbers)**.** *If $\lambda_{t,i} = 1$, with $0 < t \leq \lfloor \frac{i}{2} \rfloor$, then $l_{i,2(i-t+1)} = 0$ and $r_{i,2(i-t+1)+1} = 0$.*

Proposition 3 (Refine *lower bound* on descendants' numbers)**.** *If $\tau_{t,i} = 1$, with $\lceil \frac{i}{2} \rceil < t \leq i$, then $l_{i,2(t-1)} = 0$ and $r_{i,2t-1} = 0$.*

Overall, additional inference constraints give 10% to 50% speed up in our experiments.

Extending the model. As indicated earlier, the model can be extended to handle non-binary features, e.g. by exploiting the well-known one-hot-encoding to replace non-binary features with a set of binary features. The model naturally enables non-binary classes, simply by ensuring that a node of a given class discriminates all others. Finally, the model can be extended to accommodate for inconsistent training data, i.e. multiple classes associated with the same assignment of feature. This requires allowing some classes not to be discriminated in some decision tree path. This solution represents a more challening combinatorial optimization problem.

4 Experimental Results

We perform an experimental evalution on a set of benchmarks from [Bessiere *et al.*, 2009; Olson *et al.*, 2017]. We ran our experiments on Intel(R) Xeon(R) CPU 3.50GHz. The Glucose3 SAT solver [Audemard and Simon, 2009] was used, with a total timeout of 1000 sec in all experiments. The memory limit is 2GB. We also employed the ITI algorithm to produce decision trees from labeled examples [Utgoff *et al.*, 1997a; Utgoff *et al.*, 1997b]. The ITI algorithm does not guarantee optimality, so the size of the tree produced by ITI serves as an upper bound on the size of the optimal tree in our experiments. We refer to our encoding as DT1.

SAT	Weather	Mouse	Cancer	Car	Income
DT2*	27K	3.5M	92G	842M	354G
DT1	190K	1.2M	5.2M	4.1M	1.2G

Table 3: Comparison of sizes of CNF encodings. *Note that DT2 assumes a fixed tree structure. DT1 searches for the best tree structure, so it encodes more general problem.

4.1 Comparison with existing SAT encodings

Our first experiment is to compare our SAT encoding (DT1) with results for the SAT model reported in [Bessiere *et al.*, 2009] that we refer to as DT2. The encodings are compared in terms of their size and performance on a subset of benchmarks from [Bessiere *et al.*, 2009] that the authors kindly shared with us. For a given tree size of depth 4, we build a CNF encoding of the problem. The first row shows results reported in [Bessiere *et al.*, 2009]. Note that results for the Income benchmark were reported for 10% of data samples in Table 1 in [Bessiere *et al.*, 2009], so we report on the same amount of data randomly sampled. We would like to point out that our encoding is more general compared to DT2 that only solves the classification problem for a fixed binary tree. In our case, we encode both the tree construction for a given number of nodes and the classification task. The second row shows our results. As can be seen from Table 3, our encoding is more compact compared to the CNF encoding in [Bessiere *et al.*, 2009]. As can be observed, the encoding size can be 1000 times smaller than DT2 on some examples. Next, we discuss the performance of these encodings. DT2 was able to find the optimal only for the two small instances Weather and Mouse with the total time to prove optimality 0.37s and 577.27s, respectively. Using our encoding, the total time to prove optimality are 0.05s and 12.94s. Even though these results were obtained on different machines, it is plain that our encoding is more effective in finding and proving optimality.

4.2 Comparison on extended set of benchmarks

We perform a series of experiments on an extended set of benchmarks. The goal is to demonstrate that for problems that admit relatively small decision trees we can find the optimal decision tree and prove its optimality. We start with toy benchmarks from [Bessiere *et al.*, 2009; Olson *et al.*, 2017] that admit a small DT (less than 15 nodes). Table 4 shows our results. The second column ($\#nd^{ub}$) shows the size of the decision tree produced by ITI. The third column ($\#nd$) shows the size of optimal tree that we find. To find the optimal tree we perform a linear search on the total number of nodes starting from the upper bound provided by the ITI tree. We only consider trees with an odd number of nodes. The 'time' column shows the total time in seconds to perform this linear search procedure. As can be seen from the table, our encoding is very efficient and we can improve results of ITI and prove optimality for these benchmarks.

Next we consider larger benchmarks. Finding an optimal tree is a hard combinatorial problem as the search space is huge even for trees of size around 100 nodes. Exploring such search space is a challenging problem. However, [Bessiere *et*

Name	$\#nd^{ub}$	$\#nd$	time
Mouse	15	15	12.94
Weather	13	9	0.05
Irish	7	7	0.10
Corral	13	13	0.001
Mux6	35	15	0.11

Table 4: Results on small benchmarks.

al., 2009] showed that it is often sufficient to learn a tree on a small fraction of dataset randomly sampled, i.e. considering a reduced-size training data given the universe of samples. This gives smaller and more understandable DT with a small loss in training accuracy. We perform a similar experiment here. We randomly sampled a fraction r of samples from datasets. For each sub-sampled benchmark, we find a DT using ITI and use it as an upper bound on the size of the tree. Then we perform linear search from this upper bound as described above. Table 5 shows our results. We sample 20 benchmarks for each fraction value r and show averaged results in columns $\#nd^{ub}$, $\#nd$ and 'avg. time'. The next column shows the number of instances where we prove optimality. The next column show the number of samples ($\#b$) and the number of features ($\#f$) in the corresponding benchmarks. The last column 'test acc. ITI/DT1' shows the accuracy of the decision trees on the test set which was 25% of all datapoints for ITI and DT1 averaged over all benchmarks where DT1 proved optimality. As can be seen from the table, we can improve ITI's decision trees by up to 30%, which is a significant improvement. Contrasting our results with results in [Bessiere *et al.*, 2009], we note that the CP model does not prove optimality in any of tested benchmarks so our results are a step toward to tackling this problem. On the other hand, the CP model scales to much larger trees, up to 300 nodes, which is not feasible for our model at the moment. The key component of the CP model is an information gain-based decision heuristic, which gives an important guidance to the search procedure. It is a subject of future research how to combine this heuristic and VSIDS to provide similar guidance for the SAT solver. Another point to observe is that accuracy on the test set is around 50% for the trees that ITI/DT1 generate. The reason for this is that we build a tree using a small subsample of the benchmark set at the moment. Note that in most cases larger training sets lead to better accuracy is on the test data. For example, consider the 'Car' benchmark. If we use $r = 0.1$ fraction of the data to build a tree the accuracy is 0.55 which is 0.08 higher compared to trees built with $r = 0.05$. Comparing the accuracy of DT1 and ITI, we can see that on the majority of benchmarks DT1 accuracy is the similar as ITI while the size of trees is smaller.

Another powerful approach to perform dimensionality reduction of data is to extract relevant features. We performed the feature selection using scikit-learn [Pedregosa and et al., 2011]. Namely, we use SelectKBest method to select the most important features. The value of K, which is the number of features, was selected based on the loss of accuracy on the training set. In all tested benchmarks, we lost at most 15% of accuracy due to the dimensionality reduction. Table 6 shows our results. The second column ($\#nd_o^{ub}$) shows the number of nodes in an ITI decision tree on the original dataset where all features are present. The third column ($\#nd^{ub}$) shows the number of nodes in an ITI decision tree on reduced dataset projected on important features. The fourth column ($\#nd$) shows the number of nodes in the optimal tree that is constructed using important features. The fifth column shows the time in seconds to prove optimality. The last column show the number of samples ($\#b$), the number of features ($\#f_o$) in original benchmarks and the number of features ($\#f_r$) in the reduced benchmarks. First, we observe that dimensionality reduction reduces the decision tree size in all benchmarks. Second, our encoding is capable to find much smaller trees and prove optimality on these benchmarks.

Name	r	$\#nd^{ub}$	$\#nd$	avg. time	#o	$\#b/\#f$	test acc. ITI/DT1
Car	0.05	22.30	18.80	75.41	20	86 / 21	0.69/0.47
	0.1	33.67	23.67	684.12	3	172 / 21	0.69/0.55
Cancer	0.1	10.50	9	5.60	20	44 / 89	0.50/ 0.45
	0.2	17.71	12.86	110.11	14	89 / 89	0.49/0.46
	0.25	18.33	14.33	281.96	3	112 / 89	0.50/0.51
Shuttle	0.05	10.23	10.23	142.35	13	725 / 691	0.79/0.66
	0.1	11.67	11.67	409.19	9	1450 / 691	0.79/0.56
Colic	0.05	6.90	6.30	2.11	20	17 / 415	0.46/0.43
	0.1	11.92	9.92	130.48	13	35 / 415	0.55/0.53
Appen-tis	0.3	11.25	10.62	117.62	12	31 / 530	0.27/0.35
	0.4	9.40	9.40	204.71	5	42 / 530	0.34/0.23
Australian	0.05	12.05	10.89	271.45	19	34 / 1163	0.48/0.41
	0.1	17	13	719.22	1	69 / 1163	0.44/0.44
Backache	0.3	13	9.50	116.40	8	54 / 475	0.59/0.51
	0.4	12.33	11.67	519.27	3	72 / 475	0.38/0.17
Cleve	0.1	12.90	10.80	42.65	14	30 / 395	0.50/0.51

Table 5: Results on subsampled instances solved to optimality.

Name	$\#nd_o^{ub}$	$\#nd^{ub}$	$\#nd$	time	$\#b/\#f_o/\#f_r$
Appen-tis	39	25	25	176.51	106 / 530 / 40
Australian	203	21	19	2.97	690 / 1163 / 23
Backache	49	23	23	2.24	180 / 475 / 15
Car	95	19	19	0.24	1728 / 21 / 8
Cancer	73	41	35	198.71	683 / 89 / 9
Colic	95	21	19	0.49	368 / 415 / 10
Cleve	117	15	15	0.01	303 / 395 / 6
Haberman	171	31	25	37.37	306 / 92 / 15
Heart-statlog	99	27	25	4.76	270 / 381 / 8
Hepatitis	57	7	7	0.001	155 / 361 / 7
HouseVotes	67	29	27	2.84	435 / 16 / 7
Hungarian	101	21	19	1.40	294 / 330 / 10
New-thyroid	101	5	5	0.001	215 / 334 / 3
Promoters	25	25	23	0.31	106 / 334 / 6
Shuttle	67	7	7	0.001	14500 / 691 / 5
Spect	117	33	23	11.56	267 / 22 / 9

Table 6: Results on reduced instances solved to optimality.

5 Conclusions

This paper develops novel SAT-based solutions for computing optimal decision trees. The proposed approach exploits a number of well-established techniques for SAT-based problem solving, including exploiting properties of decision trees, breaking symmetries in the problem formulation, and using tight encodings for standard constraints, including AtMost1 constraints. In contrast with earlier work [Bessiere *et al.*, 2009], the paper shows that optimal decision trees can be computed for datasets of reasonable size. To our best knowledge, this paper presents for the first time the size of optimal decision trees for several well-known datasets.

Despite the promising results, it is also the case that learning optimal decision trees remains a challenging combinatorial problem, for which a number of additional optimations can be envisioned, and which are expected to enable targeting even more challenging datasets.

References

[Angelino *et al.*, 2017] Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin. Learning certifiably optimal rule lists. In *KDD*, 2017.

[Audemard and Simon, 2009] Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern SAT solvers. In Craig Boutilier, editor, *IJCAI*, 2009.

[Bessiere *et al.*, 2009] Christian Bessiere, Emmanuel Hebrard, and Barry O'Sullivan. Minimising decision tree size as combinatorial optimisation. In *CP*, 2009.

[Biere *et al.*, 2009] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*. IOS Press, 2009.

[Biran and Cotton, 2017] Or Biran and Courtenay Cotton. Explanation and justification in machine learning: A survey. In *IJCAI-17 Workshop on Explainable AI*, 2017.

[Breiman *et al.*, 1984] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[Cockett and Hierrera, 1990] J. Robin B. Cockett and J. A. Hierrera. Decision tree reduction. *J. ACM*, 37(4), 1990.

[DARPA, 2016] DARPA. DARPA explainable Artificial Intelligence (XAI) program. https://www.darpa.mil/program/explainable-artificial-intelligence, 2016.

[Doshi-Velez and Kim, 2017] Finale Doshi-Velez and Been Kim. A roadmap for a rigorous science of interpretability. *CoRR*, abs/1702.08608, 2017.

[Dufour, 2014] David Dufour. Finding cost-efficient decision trees. Master's thesis, University of Waterloo, 2014.

[EU Data Protection Regulation, 2016] EU Data Protection Regulation. Regulation (EU) 2016/679 of the European Parliament and of the Council, 2016.

[Goodman and Flaxman, 2017] Bryce Goodman and Seth R. Flaxman. European Union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine*, 38(3):50–57, 2017.

[Hall *et al.*, 2009] Mark A. Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[Hancock *et al.*, 1996] Thomas R. Hancock, Tao Jiang, Ming Li, and John Tromp. Lower bounds on learning decision lists and trees. *Inf. Comput.*, 126(2):114–122, 1996.

[Hauser *et al.*, 2010] John Hauser, Olivier Toubia, Theodoros Evgeniou, Rene Befurt, and Daria Dzyabura. Disjunctions of conjunctions, cognitive simplicity, and consideration sets. *J. of Marketing Research*, 47(3), 2010.

[Hyafil and Rivest, 1976] Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision trees is NP-complete. *Inf. Process. Lett.*, 5(1):15–17, 1976.

[ICML WHI Workshop, 2017] ICML WHI Workshop. ICML workshop on human interpretability in ML, 2017.

[IJCAI XAI Workshop, 2017] IJCAI XAI Workshop. IJCAI workshop on explainable artificial intelligence, 2017.

[Lakkaraju *et al.*, 2016] Himabindu Lakkaraju, Stephen Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *KDD*, 2016.

[Lakkaraju *et al.*, 2017] Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Jure Leskovec. Interpretable & explorable approximations of black box models. *CoRR*, abs/1707.01154, 2017.

[Letham *et al.*, 2015] Benjamin Letham, Cynthia Rudin, Tyler H McCormick, David Madigan, et al. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, 9(3):1350–1371, 2015.

[Li *et al.*, 2018] Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *AAAI*, 2018.

[Lou *et al.*, 2012] Yin Lou, Rich Caruana, and Johannes Gehrke. Intelligible models for classification and regression. In *KDD*, pages 150–158, 2012.

[Mitchell, 1997] Tom M. Mitchell. *Machine learning*. McGraw-Hill, 1997.

[Murthy, 1998] Sreerama K. Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Min. Knowl. Discov.*, 2(4):345–389, 1998.

[NIPS IML Symposium, 2017] NIPS IML Symposium. NIPS interpretable ML symposium, 2017.

[Olson *et al.*, 2017] Randal Olson, William La Cava, Patryk Orzechowski, Ryan Urbanowicz, and Jason Moore. PMLB: a large benchmark suite for machine learning evaluation and comparison. *BioData Mining*, 10(1):36, 2017.

[Pedregosa and et al., 2011] Fabian Pedregosa and et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[Plaisted and Greenbaum, 1986] David A. Plaisted and Steven Greenbaum. A structure-preserving clause form translation. *J. Symb. Comput.*, 2(3):293–304, 1986.

[Quinlan, 1986] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[Quinlan, 1993] J. Ross Quinlan. *C4.5: Programs for machine learning*. Morgan Kauffmann, 1993.

[Rokach and Maimon, 2015] Lior Rokach and Oded Maimon. *Data Mining with Decision Trees*. WorldScientific, 2015.

[Sinz, 2005] Carsten Sinz. Towards an optimal CNF encoding of boolean cardinality constraints. In *CP*, 2005.

[Utgoff *et al.*, 1997a] Paul E. Utgoff, Neil C. Berkman, and Jeffery A. Clouse. Decision tree induction based on efficient tree restructuring. *Machine Learning*, 29(1), 1997.

[Utgoff *et al.*, 1997b] Paul E. Utgoff, Neil C. Berkman, and Jeffery A. Clouse. Incremental decision tree induction. http://www-lrn.cs.umass.edu/iti/index.html, 1997.

[Wang *et al.*, 2015] Tong Wang, Cynthia Rudin, Finale Doshi-Velez, Yimin Liu, Erica Klampfl, and Perry MacNeille. Or's of And's for interpretable classification, with application to context-aware recommender systems. *CoRR*, abs/1504.07614, 2015.

[Weller, 2017] Adrian Weller. Challenges for transparency. *CoRR*, abs/1708.01870, 2017.