On Computing Probabilistic Explanations for Decision Trees

Marcelo Arenas[1,2,4], **Pablo Barceló**[2,4,5], **Miguel Romero**[3,5], **Bernardo Subercaseaux**[6]

Authors are listed in alphabetical order.
Correspondence at bsuberca@cs.cmu.edu.
[1] Department of Computer Science, PUC Chile
[2] Institute for Mathematical and Computational Engineering, PUC Chile
[3] Faculty of Engineering and Science, UAI Chile
[4] Millenium Institute for Foundational Research on Data, Chile
[5] CENIA Chile
[6] Carnegie Mellon University, Pittsburgh, USA

Abstract

Formal XAI (explainable AI) is a growing area that focuses on computing explanations with mathematical guarantees for the decisions made by ML models. Inside formal XAI, one of the most studied cases is that of explaining the choices taken by decision trees, as they are traditionally deemed as one of the most interpretable classes of models. Recent work has focused on studying the computation of *sufficient reasons*, a kind of explanation in which given a decision tree T and an instance $\boldsymbol{x}$, one explains the decision $T(\boldsymbol{x})$ by providing a subset $\boldsymbol{y}$ of the features of $\boldsymbol{x}$ such that for any other instance $\boldsymbol{z}$ compatible with $\boldsymbol{y}$, it holds that $T(\boldsymbol{z}) = T(\boldsymbol{x})$, intuitively meaning that the features in $\boldsymbol{y}$ are already enough to fully justify the classification of $\boldsymbol{x}$ by T. It has been argued, however, that sufficient reasons constitute a restrictive notion of explanation. For such a reason, the community has started to study their probabilistic counterpart, in which one requires that the probability of $T(\boldsymbol{z}) = T(\boldsymbol{x})$ must be at least some value $\delta \in (0, 1]$, where $\boldsymbol{z}$ is a random instance that is compatible with $\boldsymbol{y}$. Our paper settles the computational complexity of δ-sufficient-reasons over decision trees, showing that both (1) finding δ-sufficient-reasons that are minimal in size, and (2) finding δ-sufficient-reasons that are minimal inclusion-wise, do not admit polynomial-time algorithms (unless PTIME = NP). This is in stark contrast with the deterministic case ($\delta = 1$) where inclusion-wise minimal sufficient-reasons are easy to compute. By doing this, we answer two open problems originally raised by Izza et al., and extend the hardness of explanations for Boolean circuits presented by Wäldchen et al. to the more restricted case of decision trees. On the positive side, we identify structural restrictions of decision trees that make the problem tractable, and show how SAT solvers might be able to tackle these problems in practical settings.

1 Introduction

Context. The trust that AI models generate in people has been repetitively linked to our ability of *explaining* the decision of said models (Arrieta et al., 2020), thus suggesting the area of *explainable AI* (XAI) as fundamental for the deployment of trustworthy models. A sub-area of explainability that has received considerable attention over the last years, showing quick progress in theoretical and practical terms, is that of *local* explanations, i.e., explanations for the outcome of a particular input to an ML model after the model has been trained. Several queries and scores have been proposed to

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

specify explanations of this kind. These include, e.g., queries based on *prime implicants* (Shih et al., 2018) or *anchors* (Ribeiro et al., 2018), which are parts of an instance that are sufficient to explain its classification, as well as scores that intend to quantify the impact of a single feature in the output of such a classification (Lundberg and Lee, 2017; Yan and Procaccia, 2021).

A remarkable achievement of this area of research has been the development of *formal* notions of explainability. The benefits brought about by this principled approach have been highlighted, in a very thorough and convincing way, in a recent survey by Marques-Silva and Ignatiev (2022). A prime example of this kind of approach is given by *sufficient reasons*, which are also known as prime implicant explanations (Shih et al., 2018) or abductive explanations (Ignatiev et al., 2021).

Given an ML model $\mathcal{M}$ of dimension n and a Boolean input instance $\boldsymbol{x} \in \{0,1\}^n$, a sufficient reason for $\boldsymbol{x}$ under $\mathcal{M}$ is a subset $\boldsymbol{y}$ of the features of $\boldsymbol{x}$, such that any instance $\boldsymbol{z}$ compatible with $\boldsymbol{y}$ receives the same classification result as $\boldsymbol{x}$ on $\mathcal{M}$. In more intuitive words, $\boldsymbol{y}$ is a sufficient reason for $\boldsymbol{x}$ under $\mathcal{M}$ if the features in $\boldsymbol{y}$ suffices to explain the output of $\mathcal{M}$ on $\boldsymbol{x}$. In the formal explainability approach, one then aims to find sufficient reasons $\boldsymbol{y}$ that satisfy one of the following optimality criteria: (a) they are *minimum*, i.e., there are no sufficient reasons with fewer features than $\boldsymbol{y}$, or (b) they are *minimal*, i.e., there are no sufficient reasons that are strictly contained in $\boldsymbol{y}$.

Problem. The XAI community has studied for which Boolean ML models the problem of computing (minimum or minimal) sufficient reasons is computationally tractable and for which it is computationally hard (see, e.g., (Barceló et al., 2020; Marques-Silva et al., 2020, 2021)). It has been argued, however, that for practical applications sufficient reasons might be too *rigid*, as they are specified under worst-case conditions. That is, $\boldsymbol{y}$ is a sufficient reason for $\boldsymbol{x}$ under $\mathcal{M}$ if *every* "completion" of $\boldsymbol{y}$ is classified by $\mathcal{M}$ in the same way as $\boldsymbol{x}$. As several authors have noted already, there is a natural way in which this notion can be relaxed in order to become more suitable for real-world explainability tasks: Instead of asking for each completion of $\boldsymbol{y}$ to yield the same result as $\boldsymbol{x}$ on $\mathcal{M}$, we could allow for a small fragment of the completions of $\boldsymbol{y}$ to be classified differently than $\boldsymbol{x}$ (Izza et al., 2021; Wäldchen et al., 2021; Wang et al., 2021). More precisely, we would like to ensure that a random completion of $\boldsymbol{y}$ is classified as $\boldsymbol{x}$ with probability at least $\delta \in (0,1]$, a threshold that the recipient of the explanation controls. We call $\boldsymbol{y}$ a *δ-sufficient reason for* $\mathbf{e}$ *under* $\mathcal{M}$.

The study of the cost of computing minimum δ-sufficient reasons for expressive Boolean ML models based on propositional formulas was started by Wäldchen et al. (2021). They show, in particular, that the decision problem of checking if $\boldsymbol{x}$ admits a δ-sufficient reason of a certain size k under a model $\mathcal{M}$, where $\mathcal{M}$ is specified as a CNF formula, is $\mathrm{NP}^{\mathrm{PP}}$-complete. This result shows that the problem is very difficult for complex models, at least in theoretical terms. Nonetheless, it leaves the door open for obtaining tractability results over simpler Boolean models, starting from those which are often deemed to be "easy to interpret", e.g., *decision trees* (Gilpin et al., 2018; Izza et al., 2020a; Lipton, 2018). In particular, the study of the cost of computing both minimum and minimal δ-sufficient reasons for decision trees was initiated by Izza et al. (2021, 2022), but nothing beyond the fact that the problem lies in NP has been obtained. Work by Blanc et al. (2021) has shown that it is possible to obtain efficient algorithms that succeed with a certain probability, and that instead of finding a smallest (either cardinality or inclusion-wise) δ-sufficient reason, find δ-sufficient reasons that are small compared to the *average* size of δ-sufficient reasons for the considered model.

Our results. In this paper we provide an in-depth study of the complexity of the problem of minimum and minimal δ-sufficient reasons for decision trees.

1. We start by pinpointing the exact computational complexity of these problems by showing that, under the assumption that PTIME $\neq$ NP, none of them can be solved in polynomial time. We start with minimum δ-sufficient reasons and show that the problem is hard even if δ is an arbitrary fixed constant in $(0,1]$. Our proof takes as basis the fact that, assuming PTIME $\neq$ NP, the problem of computing minimum sufficient reasons for decision trees is not tractable (Barceló et al., 2020). The reduction, however, is non-trivial and requires several involved constructions and a careful analysis. The proof for minimal δ-sufficient reasons is even more difficult, and the result more surprising, as in this case we cannot start from a similar problem over decision trees: computing minimal sufficient reasons over decision trees (or, equivalently, minimal δ-sufficient reasons for $\delta = 1$) admits a simple polynomial time algorithm. Our result then implies that such a good behavior is lost when the input parameter δ is allowed to be smaller than 1.

2. To deal with the high computational complexity of the problems, we look for structural restrictions of it that, at the same time, represent meaningful practical instances and ensure that these problems can be solved in polynomial time. The first restriction, called *bounded split number*, assumes there is a constant $c \geq 1$ such that, for each node u of a decision tree T of dimension n, the number of features that are mentioned in both T_u, the subtree of T rooted at u, and $T - T_u$, the subtree of T obtained by removing T_u, is at most c. We show that the problems of computing minimum and minimal δ-sufficient reasons over decision trees with bounded split number can be solved in polynomial time. The second restriction is *monotonicity*. Intuitively, a Boolean ML model $\mathcal{M}$ is *monotone*, if the class of instances that are classified positively by $\mathcal{M}$ is closed under the operation of replacing 0s with 1s. For example, if $\mathcal{M}$ is of dimension 3 and it classifies the input $(1, 0, 0)$ as positive, then so it does for all the instances in $\{(1, 0, 1), (1, 1, 0), (1, 1, 1)\}$. We show that computing minimal δ-sufficient reasons for monotone decision trees is a tractable problem. (This good behavior extends to any class of monotone ML models for which counting the number of positive instances is tractable; e.g., monotone *free binary decision diagrams* (Wegener, 2004)).

3. In spite of the intractability results we obtain in the paper, we show experimentally that our problems can be solved over practical instances by using SAT solvers. This requires finding efficient encodings of such problems as conjunctive normal form (CNF) formulas, which we then check for satisfiability. This is particularly non-trivial for probabilistic sufficient reasons, as it requires dealing with the arithmetical nature of the probabilities involved through a Boolean encoding.

Organization of the paper. We introduce the main terminology used in the paper in Section 2, and we define the problems of computing minimum and minimal δ-sufficient reasons in Section 3. The intractability of the these problems is proved in Section 4, while some tractable restrictions of them are provided in Section 5. Our Boolean encodings can be found in Section 6. Finally, we discuss some future work in Section 7.

2 Background

An *instance* of dimension n, with $n \geq 1$, is a tuple $\boldsymbol{x} \in \{0, 1\}^n$. We use notation $\boldsymbol{x}[i]$ to refer to the i-th component of this tuple, or equivalently, its i-th feature. Moreover, we consider an abstract notion of a model of dimension n, and we define it as a Boolean function $\mathcal{M} : \{0, 1\}^n \to \{0, 1\}$. That is, $\mathcal{M}$ assigns a Boolean value to each instance of dimension n, so that we focus on binary classifiers with Boolean input features. Restricting inputs and outputs to be Boolean makes our setting cleaner while still covering several relevant practical scenarios.

A *partial instance* of dimension n is a tuple $\boldsymbol{y} \in \{0, 1, \bot\}^n$. Intuitively, if $\boldsymbol{y}[i] = \bot$, then the value of the i-th feature is undefined. Notice that an instance is a particular case of a partial instance where all features are assigned value either 0 or 1. Given two partial instances $\boldsymbol{x}, \boldsymbol{y}$ of dimension n, we say that $\boldsymbol{y}$ is *subsumed* by $\boldsymbol{x}$, denoted by $\boldsymbol{y} \subseteq \boldsymbol{x}$, if for every $i \in \{1, \ldots, n\}$ such that $\boldsymbol{y}[i] \neq \bot$, it holds that $\boldsymbol{y}[i] = \boldsymbol{x}[i]$. That is, $\boldsymbol{y}$ is subsumed by $\boldsymbol{x}$ if it is possible to obtain $\boldsymbol{x}$ from $\boldsymbol{y}$ by replacing some undefined values. Moreover, we say that $\boldsymbol{y}$ is *properly subsumed* by $\boldsymbol{x}$, denoted by $\boldsymbol{y} \subsetneq \boldsymbol{x}$, if $\boldsymbol{y} \subseteq \boldsymbol{x}$ and $\boldsymbol{y} \neq \boldsymbol{x}$. Notice that a partial instance $\boldsymbol{y}$ can be thought of as a compact representation of the set of instances $\boldsymbol{z}$ such that $\boldsymbol{y}$ is subsumed by $\boldsymbol{z}$, where such instances $\boldsymbol{z}$ are called the *completions* of $\boldsymbol{y}$ and are grouped in the set $\textsc{Comp}(\boldsymbol{y})$.

A *binary decision diagram* (BDD) of dimension n is a rooted directed acyclic graph $\mathcal{M}$ with labels on edges and nodes such that: (i) each leaf (a node with no outgoing edges) is labeled with $\mathbf{true}$ or $\mathbf{false}$; (ii) each internal node (a node that is not a leaf) is labeled with a feature $i \in \{1, \ldots, n\}$; and (iii) each internal node has two outgoing edges, one labeled 1 and the other one labeled 0. Every instance $\boldsymbol{x} \in \{0, 1\}^n$ defines a unique path $\pi_{\boldsymbol{x}} = u_1 \cdots u_k$ from the root u_1 to a leaf u_k of $\mathcal{M}$ such that: if the label of u_i is $j \in \{1, \ldots, n\}$, where $i \in \{1, \ldots, k-1\}$, then the edge from u_i to u_{i+1} is labeled with $\boldsymbol{x}[j]$. Moreover, the instance $\boldsymbol{x}$ is positive, denoted by $\mathcal{M}(\boldsymbol{x}) = 1$, if the label of u_k is $\mathbf{true}$; otherwise the instance $\boldsymbol{x}$ is negative, which is denoted by $\mathcal{M}(\boldsymbol{x}) = 0$. A BDD $\mathcal{M}$ is *free* if for every path from the root to a leaf, no two nodes on that path have the same label. A *decision tree* is simply a free BDD whose underlying directed acyclic graph is a rooted tree.

3 Probabilistic Sufficient Reasons

Sufficient reasons are partial instances obtained by removing from an instance $\boldsymbol{x}$ components that do not affect the final classification. Formally, fix a dimension n. Given a decision tree T, an instance $\boldsymbol{x}$, and a partial instance $\boldsymbol{y}$ with $\boldsymbol{y} \subseteq \boldsymbol{x}$, we call $\boldsymbol{y}$ a *sufficient reason* for $\boldsymbol{x}$ under T if $T(\boldsymbol{x}) = T(\boldsymbol{z})$ for every $\boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})$. In other words, the features of $\boldsymbol{y}$ that take value either 0 or 1 explain the decision taken by T on $\boldsymbol{x}$, as $T(\boldsymbol{x})$ would not change if the remaining features (i.e., those that are undefined in $\boldsymbol{y}$) were to change in $\boldsymbol{x}$, thus implying that the classification $T(\boldsymbol{x})$ is a consequence of the features defined in $\boldsymbol{y}$. We say that a sufficient reason $\boldsymbol{y}$ for $\boldsymbol{x}$ under T is *minimal*, if it is minimal under the order induced by $\subseteq$, that is, if there is no sufficient reason $\boldsymbol{y}'$ for $\boldsymbol{x}$ under T such that $\boldsymbol{y}' \subsetneq \boldsymbol{y}$. Also, we define a *minimum* sufficient reason for $\boldsymbol{x}$ under T as a sufficient reason $\boldsymbol{y}$ for $\boldsymbol{x}$ under T that maximizes the value $|\boldsymbol{y}|_\perp := |\{i \in \{1, \dots, n\} \mid \boldsymbol{y}[i] = \perp\}|$.

It turns out that minimal sufficient reasons can be computed efficiently for decision trees with a very simple algorithm, assuming a sub-routine to check whether a given partial instance is a sufficient reason (not necessarily minimal) of another given instance. As shown in Algorithm 1, the idea of the algorithm is as follows: start with a candidate answer $\boldsymbol{y}$ which is initially equal to $\boldsymbol{x}$, the instance to explain, and maintain the invariant that $\boldsymbol{y}$ is a sufficient reason for $\boldsymbol{x}$, while trying to remove defined components from $\boldsymbol{y}$ until no longer possible. It is not hard to see that one can check whether a partial instance $\boldsymbol{y}$ is a sufficient reason for an instance $\boldsymbol{x}$ in linear time over decision trees (Audemard et al., 2021). This algorithm is well known (see e.g., (Huang et al., 2021)), and relies on the following simple observation, tracing back to Goldsmith et al. (2005).

Observation 1. *For any class of models $\mathfrak{C}$, if a partial instance $\boldsymbol{y}$ of dimension n is a sufficient reason for an instance $\boldsymbol{x}$ under a model $\mathcal{M} \in \mathfrak{C}$, but not a minimal sufficient reason, then there is a partial instance $\hat{\boldsymbol{y}}$ which is equal to $\boldsymbol{y}$ except that $\hat{\boldsymbol{y}}[i] = \perp, \boldsymbol{y}[i] \neq \perp$ for some $i \in \{1, \dots, n\}$ which is also a sufficient reason for $\boldsymbol{x}$ under $\mathcal{M}$.*

Algorithm 1: Minimal Sufficient Reason

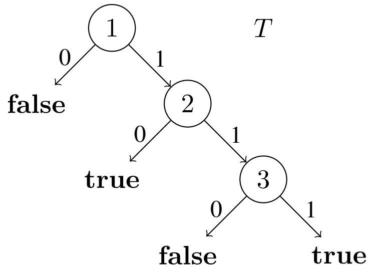
Input: Decision tree T and instance $\boldsymbol{x}$, both of dimension n
Output: A minimal sufficient reason $\boldsymbol{y}$ for $\boldsymbol{x}$ under T.

```
$\boldsymbol{y} \leftarrow \boldsymbol{x}$
while true do
    reduced ← false
    for $i \in \{1, \dots, n\}$ do
        $\hat{\boldsymbol{y}} \leftarrow \boldsymbol{y}$
        $\hat{\boldsymbol{y}}[i] \leftarrow \perp$
        if CheckSufficientReason$(T, \hat{\boldsymbol{y}}, \boldsymbol{x})$ then
            $\boldsymbol{y} \leftarrow \hat{\boldsymbol{y}}$
            reduced ← true
            break
    if (¬reduced) or $|\boldsymbol{y}|_\perp = n$ then
        return $\boldsymbol{y}$
```

The following theorem shows a stark contrast between the complexity of computing minimal and minimum sufficient reasons over decision trees.

Theorem 1 (Barceló et al. (2020))**.** *Assuming* PTIME $\neq$ NP*, there is no polynomial-time algorithm that, given a decision tree T and an instance $\boldsymbol{x}$ of the same dimension, computes a minimum sufficient reason for $\boldsymbol{x}$ under T.*

Arguably, the notion of sufficient reason is a natural notion of explanation for the result of a classifier. However, such a concept imposes a severe restriction by asking all completions of a partial instance to be classified in the same way. To overcome this limitation, a probabilistic generalization of sufficient reasons was proposed by Wäldchen et al. (2021) and Izza et al. (2021). More precisely, this notion allows to settle a confidence $\delta \in (0, 1]$ on the fraction of completions of a partial instance that yield the same classification.



$p(\emptyset) = \frac{3}{8}$	$p(\{1,2\}) = \frac{1}{2}$
$p(\{1\}) = \frac{3}{4}$	$p(\{1,3\}) = 1$
$p(\{2\}) = \frac{1}{4}$	$p(\{2,3\}) = \frac{1}{2}$
$p(\{3\}) = \frac{1}{2}$	$p(\{1,2,3\}) = 1$

Figure 1: The decision tree T and the values $p(X)$ from Example 1.

Definition 1 (Probabilistic sufficient reasons). *Given a value $\delta \in (0,1]$, a δ-*sufficient reason *(δ-SR for short) for an instance $\boldsymbol{x}$ under a decision tree T is a partial instance $\boldsymbol{y}$ such that $\boldsymbol{y} \subseteq \boldsymbol{x}$ and*

$$\mathrm{Pr}_{\boldsymbol{z}}[T(\boldsymbol{z}) = T(\boldsymbol{x}) \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})] \ := \ \frac{|\{\boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}) \mid T(\boldsymbol{z}) = T(\boldsymbol{x})\}|}{2^{|\boldsymbol{y}|_\perp}} \ \geq \ \delta.$$

Minimal and minimum δ-sufficient reasons are defined analogously as in the case of minimal and minimum sufficient reasons.

Example 1. Consider the decision tree T over features $\{1,2,3\}$ shown in Figure 1 and the input instance $\boldsymbol{x} = (1,1,1)$. Notice that $T(\boldsymbol{x}) = 1$. For each $X \subseteq \{1,2,3\}$, we show the probability $p(X) := \mathrm{Pr}_{\boldsymbol{z}}[T(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}_X)]$, where $\boldsymbol{y}_X$ is the partial instance that is obtained from $\boldsymbol{x}$ by fixing $\boldsymbol{y}_X[i] = \perp$ for each $i \notin X$. We observe, for instance, that $\boldsymbol{x}$ itself is neither a minimum nor a minimal 1-SR for $\boldsymbol{x}$ under T, as $\boldsymbol{y}_{\{1,3\}} = (1,\perp,1)$ is also a 1-SR. In turn, $\boldsymbol{y}_{\{1,3\}}$ is both a minimal and a minimum 1-SR for $\boldsymbol{x}$ under T. The partial instance $\boldsymbol{y}_{\{1,3\}}$ is not, however, a minimal or a minimum $3/4$-SR for $\boldsymbol{x}$ under T, as $\boldsymbol{y}_{\{1\}} = (1,\perp,\perp)$ is also a $3/4$-SR. □

Example 2. Consider the decision tree T over features $\{1,2,3\}$ shown in Figure 2 and the input instance $\boldsymbol{x} = (1,1,1)$. Notice that $T(\boldsymbol{x}) = 1$. Exactly as in Example 1, we display as well the probabilities $p(X)$ for each $X \subseteq \{1,2,3\}$. Interestingly, this example illustrates that Observation 1 does not hold when $\delta < 1$. Indeed, consider that $\boldsymbol{y}_{\{1,2,3\}}$ is a $5/8$-SR which is not minimal, as $\boldsymbol{y}_\emptyset$ is also a $5/8$-SR, but if we remove any single feature from $\boldsymbol{y}_{\{1,2,3\}}$, we obtain a partial instance which is not a $5/8$-SR. □

As illustrated on Example 2, it is not true in general that if $\boldsymbol{y}' \subset \boldsymbol{y}$ then

$$\mathrm{Pr}_{\boldsymbol{z}}[T(\boldsymbol{z}) = T(\boldsymbol{x}) \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}')] \leq \mathrm{Pr}_{\boldsymbol{z}}[T(\boldsymbol{z}) = T(\boldsymbol{x}) \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})],$$

which means that standard algorithms for finding minimal sets holding monotone predicates (see e.g., (Marques-Silva et al., 2013)) cannot be used to compute minimal δ-SRs.

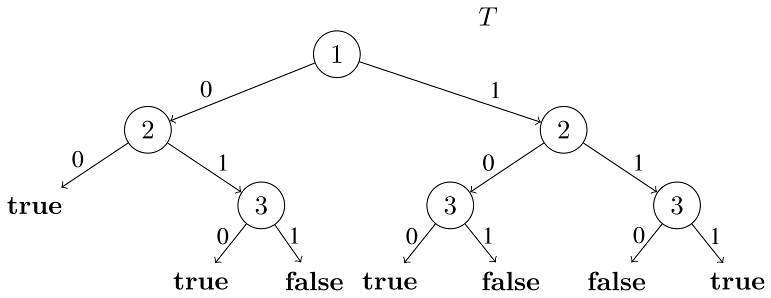
The problems of computing minimum and minimal δ-SR on decision trees were defined and left open by (Izza et al., 2021, 2022). These problems are formally defined as follows.

PROBLEM: :	Compute-Minimum-SR
INPUT :	A decision tree T of dimension n, an instance $\boldsymbol{x}$ of dimension n and $\delta \in (0,1]$
OUTPUT :	A minimum δ-SR for $\boldsymbol{x}$ under T

PROBLEM: :	Compute-Minimal-SR
INPUT :	A decision tree T of dimension n, an instance $\boldsymbol{x}$ of dimension n and $\delta \in (0,1]$
OUTPUT :	A minimal δ-SR for $\boldsymbol{x}$ under T

4 The Complexity of Probabilistic Sufficient Reasons on Decision Trees

In what follows, we show that neither Compute-Minimum-SR nor Compute-Minimal-SR can be solved in polynomial time (unless PTIME = NP). We first consider the problem



$p(\emptyset) = \frac{5}{8}$	$p(\{1,2\}) = \frac{1}{2}$
$p(\{1\}) = \frac{1}{2}$	$p(\{1,3\}) = \frac{1}{2}$
$p(\{2\}) = \frac{1}{2}$	$p(\{2,3\}) = \frac{1}{2}$
$p(\{3\}) = \frac{1}{2}$	$p(\{1,2,3\}) = 1$

Figure 2: The decision tree T and the values $p(X)$ from Example 2.

Compute-Minimum-SR, and in fact prove a stronger result by considering the family of problems δ-Compute-Minimum-SR where $\delta \in (0,1]$ is assumed to be fixed. More precisely, we obtain as a corollary of Theorem 1 that 1-Compute-Minimum-SR cannot be solved efficiently. Moreover, a non-trivial modification of the proof of this theorem shows that this negative result continues to hold for every fixed $\delta \in (0,1]$.

Theorem 2. *Fix $\delta \in (0,1]$. Then assuming that* PTIME $\neq$ NP*, there is no polynomial-time algorithm for δ-*Compute-Minimum-SR.

Let us now look at the problem Compute-Minimal-SR. When $\delta = 1$, this problem can be solved in polynomial time as stated in Theorem 1. However, it is conjectured by Izza et al. (2021) that assuming PTIME $\neq$ NP, this positive behavior does not extend to the general problem Compute-Minimal-SR, in which δ is an input confidence parameter. Our main result confirms that this conjecture is correct.

Theorem 3. *Assuming that* PTIME $\neq$ NP*, there is no polynomial-time algorithm for* Compute-Minimal-SR.

Proof sketch. We first show that the following decision problem, called Check-Sub-SR, is NP-hard. This problem takes as input a tuple $(T, \boldsymbol{x})$, for T a decision tree of dimension n, and $\boldsymbol{x} \in \{0,1,\perp\}^n$ a partial instance, and the goal is to decide whether there is a partial instance $\boldsymbol{y} \subsetneq \boldsymbol{x}$ with $\Pr_{\boldsymbol{z}}[T(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})] \geq \Pr_{\boldsymbol{z}}[T(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{x})]$. We then show that if Compute-Minimal-SR admits a polynomial time algorithm, then Check-Sub-SR is in PTIME, which contradicts the assumption that PTIME $\neq$ NP. The latter reduction requires an involved construction exploiting certain properties of the hard instances for Check-Sub-SR.

To show that Check-Sub-SR is NP-hard, we use a polynomial time reduction from a decision problem over formulas in CNF, called Minimal-Expected-Clauses, and which we also show to be NP-hard. Both the NP-hardness of Minimal-Expected-Clauses and the reduction from Minimal-Expected-Clauses to Check-Sub-SR may be of independent interest.

We now define the problem Minimal-Expected-Clauses. Let φ be a CNF formula over variables $X = \{x_1, \ldots, x_n\}$. Partial assignments of the variables in X, as well as the notions of subsumption and completions over them, are defined in exactly the same way as for partial instances over features. For a partial assignment μ over X, we denote by $E(\varphi, \mu)$ the expected number of clauses of φ satisfied by a random completion of μ. We then consider the following problem for fixed $k \geq 2$ (recall that a k-CNF formula is a CNF formula where each clause has at most k literals):

PROBLEM :	k-Minimal-Expected-Clauses
INPUT :	(φ, σ), for φ a k-CNF formula and σ a partial assignment
OUTPUT :	Yes, if there is a partial assignment $\mu \subsetneq \sigma$ such that $E(\varphi, \mu) \geq E(\varphi, \sigma)$ and No otherwise

We show that k-Minimal-Expected-Clauses is NP-hard even for $k = 2$, via a reduction from the well-known clique problem. Finally, the reduction from k-Minimal-Expected-Clauses to Check-Sub-SR builds an instance $(T, \boldsymbol{x})$ from (φ, σ) in a way that there is a direct correspondence between partial assignments $\mu \subseteq \sigma$ and partial instances $\boldsymbol{y} \subseteq \boldsymbol{x}$, satisfying that

$$\Pr_{\boldsymbol{x}}[T(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})] = \frac{E(\varphi, \mu)}{m},$$

where m is the number of clauses of φ. This implies that (φ, σ) is a Yes-instance for k-Minimal-Expected-Clauses if and only if $(T, \boldsymbol{x})$ is a Yes-instance for Check-Sub-SR. □

5 Tractable Cases

We now study restrictions on decision trees that lead to polynomial time algorithms for Compute-Minimum-SR or Compute-Minimal-SR, hence avoiding the general intractability results shown in the previous section. We identify two such restrictions: *bounded split number* and *monotonicity*.

5.1 Bounded split number

Let T be a decision tree of dimension n. For a set U of nodes of T, we denote by $\mathcal{F}(U)$ the set of features from $\{1, \dots, n\}$ labeling the nodes in U. For each node u of T, we denote by $N_u^{\downarrow}$ the set of nodes appearing in T_u, that is, the subtree of T rooted at u. On the other hand, we denote by $N_u^{\uparrow}$ the set of nodes of T minus the set of nodes $N_u^{\downarrow}$. We define the *split number* of the decision tree T to be

$$\max_{\text{node } u \text{ in } T} \left| \mathcal{F}(N_u^{\downarrow}) \cap \mathcal{F}(N_u^{\uparrow}) \right|.$$

Intuitively, the split number of a decision tree T is a measure of the interaction (number of common features) between the subtrees of the form T_u and their exterior. A small split number allows us to essentially treat each subtree T_u independently (in particular, the left and right subtrees below any node), which in turn leads to efficient algorithms for the problems Compute-Minimum-SR and Compute-Minimal-SR.

Theorem 4. *Let $c \geq 1$ be a fixed integer. Both Compute-Minimum-SR and Compute-Minimal-SR can be solved in polynomial time for decision trees with split number at most c.*

Proof Sketch. It suffices to provide a polynomial time algorithm for Compute-Minimum-SR. (The same algorithm works for Compute-Minimal-SR as a minimum δ-SR is in particular minimal.) In turn, using standard arguments, it is enough to provide a polynomial time algorithm for the following decision problem Check-Minimum-SR: Given a tuple $(T, \boldsymbol{x}, \delta, k)$, where T is a decision tree of dimension n, $\boldsymbol{x} \in \{0, 1, \bot\}^n$ is a partial instance, $\delta \in (0, 1]$, and $k \geq 0$ is an integer, decide whether there is a partial instance $\boldsymbol{y} \subseteq \boldsymbol{x}$ such that $n - |\boldsymbol{y}|_{\bot} \leq k$ and $\Pr_{\boldsymbol{z}}[T(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})] \geq \delta$.

In order to solve Check-Minimum-SR over an instance $(T, \boldsymbol{x}, \delta, k)$, where T has split number at most c, we apply dynamic programming over T in a bottom-up manner. Let $Z \subseteq \{1, \dots, n\}$ be the set of features defined in $\boldsymbol{x}$, that is, features i with $\boldsymbol{x}[i] \neq \bot$. Those are the features we could eventually remove when looking for $\boldsymbol{y}$. For each node u in T, we solve a polynomial number of subproblems over the subtree T_u. We define

$$\mathsf{Int}(u) := \mathcal{F}(N_u^{\downarrow}) \cap \mathcal{F}(N_u^{\uparrow}) \cap Z \qquad\qquad \mathsf{New}(u) := \left(\mathcal{F}(N_u^{\downarrow}) \setminus \mathsf{Int}(u)\right) \cap Z.$$

In other words, $\mathsf{Int}(u)$ are the features appearing both inside and outside T_u, while $\mathsf{New}(u)$ are the features only inside T_u, that is, the new features introduced below u. Both sets are restricted to Z as features not in Z play no role in the process.

Each particular subproblem is indexed by a possible size $s \in \{0, \dots, k\}$ and a possible set $J \subseteq \mathsf{Int}(u)$ and the goal is to compute the quantity:

$$p_{u,s,J} := \max_{\boldsymbol{y} \in \mathcal{C}_{u,s,J}} \Pr_{\boldsymbol{z}}[T_u(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})],$$

where $\mathcal{C}_{u,s,J}$ is the space of partial instances $\boldsymbol{y} \subseteq \boldsymbol{x}$ with $n - |\boldsymbol{y}|_\perp \leq s$ and such that $\boldsymbol{y}[i] = \boldsymbol{x}[i]$ for $i \in J$ and $\boldsymbol{y}[i] = \perp$ for $i \in \mathsf{Int}(u) \setminus J$. In other words, the set J fixes the behavior on $\mathsf{Int}(u)$ (keep features in J, remove features in $\mathsf{Int}(u) \setminus J$) and hence the maximization occurs over choices on the set $\mathsf{New}(u)$ (which features are kept and which features are removed). The key idea is that $p_{u,s,J}$ can be computed inductively using the information already computed for the children u_1 and u_2 of u. Intuitively, this holds since the common features between T_{u_1} and T_{u_2} are at most c, which is a fixed constant, and hence we can efficiently synchronize the information stored for u_1 and u_2. Finally, to solve the instance $(T, \boldsymbol{x}, \delta, k)$ we simply check whether $p_{r,k,\emptyset} \geq \delta$, for the root r of T. □

5.2 Monotonicity

Monotonic classifiers have been studied in the context of XAI as they often present tractable cases for different explanations, as shown by Marques-Silva et al. (2021). The computation of minimal sufficient reasons for monotone models was known to be in PTIME since the work of Goldsmith et al. (2005). We show that this is also the case for computing minimal δ-SRs under a mild assumption on the class of models.

Let us define the ordering $\preceq$ for instances in $\{0,1\}^n$ as follows:

$$\boldsymbol{x} \preceq \boldsymbol{z} \quad \text{iff} \quad \boldsymbol{x}[i] \leq \boldsymbol{z}[i], \text{ for all } i \in \{1, \dots, n\}.$$

We can now define monotonicity as follows. A model $\mathcal{M}$ of dimension n is said to be *monotone* if for every pair of instances $\boldsymbol{x}, \boldsymbol{z} \in \{0,1\}^n$, it holds that $\boldsymbol{x} \preceq \boldsymbol{z} \implies \mathcal{M}(\boldsymbol{x}) \leq \mathcal{M}(\boldsymbol{z})$.

We now prove that the problem of computing minimal probabilistic sufficient reasons can be solved in polynomial time for any class $\mathfrak{C}$ of monotone Boolean models for which the problem of counting positive completions can be solved efficiently. Formally, the latter problem is defined as follows: given a model $\mathcal{M} \in \mathfrak{C}$ of dimension n and a partial instance $\boldsymbol{y} \in \{0, 1, \perp\}^n$, compute $|\{\boldsymbol{x} \in \textsc{Comp}(\boldsymbol{y}) \mid \mathcal{M}(\boldsymbol{x}) = 1\}|$. We call this problem $\mathfrak{C}$-Count-Positive-Completions

Theorem 5. *Let $\mathfrak{C}$ be a class of monotone Boolean models such that the problem $\mathfrak{C}$-Count-Positive-Completions can be solved in polynomial time. Then Compute-Minimal-SR can be solved in polynomial time over $\mathfrak{C}$.*

Proof sketch. Consider a partial instance $\boldsymbol{y}$ of dimension n and $i \in \{1, \dots, n\}$. Suppose $\boldsymbol{y}[i] \neq \perp$. We write $\boldsymbol{y} \setminus \{i\}$ for the partial instance $\boldsymbol{y}'$ that is exactly equal to $\boldsymbol{y}$ save for the fact that $\boldsymbol{y}'[i] = \perp$. We make use of the following lemma, which is a probabilistic counterpart to Observation 1.

Lemma 1. *Let $\mathfrak{C}$ be a class of monotone models, $\mathcal{M} \in \mathfrak{C}$ a model of dimension n, and $\boldsymbol{x} \in \{0,1\}^n$ an instance. Consider any $\delta \in (0, 1]$. Then if $\boldsymbol{y} \subseteq \boldsymbol{x}$ is a δ-SR for $\boldsymbol{x}$ under $\mathcal{M}$ which is not minimal, then there is a partial instance $\boldsymbol{y} \setminus \{i\}$, for some $i \in \{1, \dots, n\}$, that is also a δ-SR for $\boldsymbol{x}$ under $\mathcal{M}$.*

With this lemma we can prove Theorem 5. In fact, it can be shown then that a slight variant of Algorithm 1 computes in polynomial time a minimal δ-SR for $\boldsymbol{x}$ under $\mathcal{M}$. □

As a corollary, the computation of minimal probabilistic sufficient reasons can be carried out in polynomial time not only over monotone decision trees, but also over monotone free BDDs.

Corollary 1. *The problem of computing minimal δ-SRs can be solved in polynomial time over the class of monotone free BDDs.*

6 SAT to the Rescue!

Despite the theoretical intractability results presented earlier on, many NP-complete problems can be solved over practical instances with the aid of *SAT solvers*. By definition, any NP-complete problem can be *encoded* as a CNF satisfiability (SAT) problem, which is then solved by a highly optimized program, a *SAT solver*. In particular, if a satisfying assignment is found for the CNF instance, one can translate such an assignment back to a solution for the original problem. In fact, this paradigm has been successfully used in the literature for other explainability problems (Ignatiev et al., 2021, 2022; Izza et al., 2020b). The effectiveness of this approach is highly dependent on the particular encoding being used (Biere et al., 2009), where the aspect that arguably impacts performance the most is the size of the encoding, measured as the number of clauses of the resulting CNF formula.

In this section we present an encoding that uses some standard automated reasoning tools (e.g., *sequential enconding*s (Sinz, 2005)) combined with ad-hoc *bit-blasting* (Biere et al., 2009), where the arithmetic operations required for probabilistic reasoning are implemented as Boolean circuits and then encoded as CNF by Tseitin transformations. The appendix describes experimentation both over synthetic datasets and standard datasets (MNIST) and reports empirical results. A recent report by Izza et al. (2022) also uses automated reasoning for this problem, although through an SMT solver.

Let us consider the following decision version of the Compute-Minimum-SR problem.

PROBLEM: :	Decide-Minimum-SR
INPUT :	A decision tree T of dimension n, an instance $\boldsymbol{x}$ of dimension n, an integer $k \leq n$, and $\delta \in (0, 1]$
OUTPUT :	Yes if there is a δ-SR $\boldsymbol{y}$ for $\boldsymbol{x}$ under T with $|\boldsymbol{y}|_\perp \leq k$, and No otherwise.

This problem is NP-complete. Membership is already proven by Izza et al. (2021). Hardness follows directly from Theorem 2, as if one were able to solve Decide-Minimum-SR in polynomial time, then a simple binary search over k would allow solving Compute-Minimum-SR in polynomial time.

6.1 Deterministic Encoding

Let us first propose an encoding for the particular case of $\delta = 1$. First, create Boolean variables f_i for $i \in \{1, \dots, n\}$, with f_i representing that $\boldsymbol{y}[i] \neq \perp$, where $\boldsymbol{y}$ is the desired δ-SR. Then, for every node u of the tree T, create a variable r_u representing that node u is *reachable* by a completion of $\boldsymbol{y}$, meaning that there exists a completion of $\boldsymbol{y}$ that goes through node u when evaluated over T. We then want to enforce that:

1. $r_{\text{ROOT}} = 1$, where ROOT is the root of T. This means that the root is always reachable.
2. The desired δ-SR $\boldsymbol{y}$ satisfies $n - |\boldsymbol{y}|_\perp \leq k$, meaning that $\sum_{i=1}^{n} f_i \leq k$.
3. If $T(\boldsymbol{x}) = 1$, and F is the set of **false** leaves of T, then $r_\ell = 0$ for every $\ell \in F$. This means that if we want to explain a positive instance, the completions of the explanation $\boldsymbol{y}$ must all be positive (recall we assume $\delta = 1$), and thus no **false** leaf should be reachable. Conversely, If $T(\boldsymbol{x}) = 0$, and G is the set of **true** leaves of T, then $r_\ell = 0$ for every $\ell \in G$.
4. The semantics of reachability is *consistent*: if a node u is reachable, and its labeled with feature i, then if $f_i = 0$, meaning that feature i is undefined in $\boldsymbol{y}$, both children of u should be reachable too. In case $f_i = 1$, only the child along the edge corresponding to $\boldsymbol{x}[i]$ should be reachable.

Let us analyze the size of this encoding. Condition 1 requires a single clause to be enforced. Condition 2 can be enforced with $O(nk)$ variables and clauses using the linear encoding of Sinz (2005). Condition 3 requires at most one clause per leaf of T and thus at most $O(|T|)$ clauses. Condition 4 can be implemented with a constant number of clauses per node. We thus incur in a total of $O(nk + |T|)$ variables and clauses, which is pretty much optimal considering $\Omega(n + |T|)$ is a lower bound on the representation of the input.

Note as well that from a satisfying assignment we can trivially recover the desired explanation $\boldsymbol{y}$ as

$$\boldsymbol{y}[i] = \begin{cases} \boldsymbol{x}[i] & \text{if } f_i = 1, \\ \perp & \text{otherwise.} \end{cases}$$

An efficient encoding supporting values of δ different than 1 is significantly more challenging and involved, and thus we only provide an outline here. An exhaustive description, together with the implementation, is provided in the supplementary material.

6.2 Probabilistic Encoding

In order to encode that $\Pr_{\boldsymbol{z}}[T(\boldsymbol{z}) = T(\boldsymbol{x}) \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})] \geq \delta$, we will directly encode that $|\{\boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}) \mid T(\boldsymbol{z}) = T(\boldsymbol{x})\}| \geq \lceil \delta 2^{|\boldsymbol{y}|_\perp} \rceil$, where we assume for simplicity that the ceiling can be take safely, in order to have a value we can represent by an integer. This assumption is not crucial. As before, we will have f_i variables representing whether $\boldsymbol{y}[i] \neq \perp$ or not, and enforce that $\sum_{i=1}^{n} f_i \leq k$. Now define variables c_j for $j \in \{0, \dots, n\}$, such that c_j is true exactly when $\sum_{i=1}^{n} f_i = j$. This can again be done efficiently via a linear encoding. Let t_i for $i \in \{0, \dots, n\}$ a series of variables that represent the bits of an integer t. The integer t will correspond to the value of $\lceil \delta 2^{|\boldsymbol{y}|_\perp} \rceil$. Note that as $|\boldsymbol{y}|_\perp$ can only take $n+1$ different values, there are only $n+1$ different values that t can take. Moreover, the value of the c_j variables completely determines the value of t, and thus we can manually encode how the c_j variables determine the bits t_i. We can now assume that we have access to t through its bits t_i. Our goal now is to build a binary representation of

$$\alpha = |\{\boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}) \mid T(\boldsymbol{z}) = T(\boldsymbol{x})\}|,$$

so that we can then implement the condition $\alpha \geq t$ with a Boolean circuit on their bits.

In order to represent α, we will decompose the number of instances $\boldsymbol{z}$ according to the leaves of T as follows. If L is the set of leaves of T whose label matches $T(\boldsymbol{x})$, then

$$|\{\boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}) \mid T(\boldsymbol{z}) = T(\boldsymbol{x})\}| = \sum_{\ell \in L} |\{\boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}) \mid T(\boldsymbol{z}) \rightsquigarrow \ell)\}|,$$

where notation $T(\boldsymbol{z}) \rightsquigarrow \ell$ means that the evaluation of $\boldsymbol{z}$ under T ends in the leaf ℓ. For a given leaf ℓ, we can compute its *weight* $w_\ell := |\{\boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}) \mid T(\boldsymbol{z}) \rightsquigarrow \ell)\}|$ as described next. Let F_ℓ be the set of labels appearing in the unique path from the root of T down to ℓ. Now let u_ℓ be the number of undefined features of $\boldsymbol{y}$ along said unique path, and thus u_ℓ can be defined as follows: $u_\ell := |F_\ell \setminus \{i \mid f_i = 1\}|$. The sets F_ℓ depend only on T and thus can be precomputed. Therefore we can use a linear encoding again to define the values u_ℓ. It is simple to observe now that $w_\ell = 2^{u_\ell}$. which means that by representing the values u_ℓ we can trivially represent the values w_ℓ in binary (they will simply consist of a 1 in their $(u_\ell + 1)$-th position right-to-left), and then implement a Boolean addition circuit to compute $\alpha = \sum_\ell w_\ell$. This concludes the outline of the encoding. Its number of variables clauses can be shown to be at most $O(n^2|T| + nk)$.

7 Conclusions and Future Work

The results proven in this paper suggest that minimal (or minimum) explanations might be hard to obtain in practice even for decision trees, especially in problems where the feature space has large dimension. A way to circumvent the limitations proven in our work is to relax the guarantee of minimality, or introduce some probability of error, as done in the work of Blanc et al. (2021). A promising direction of future research is to better understand the kind of guarantees and settings in which is still possible to obtain tractability.

Our work leaves open some interesting technical questions: (1) What is the *parameterized* complexity of computing minimum δ-SRs over decision trees, assuming that the parameter is the size of the explanation one is looking for? It is not hard to see that W[2] hardness follows from our proofs (i.e., they are parameterized reductions all the way to Set Cover), but membership in any class is fully open. (2) Does Theorem 2 continue to hold for monotone decision trees? That is, is it the case that computing minimum δ-SRs over monotone decision trees is hard for every fixed $\delta \in (0, 1]$? (3) Is it the case that the hardness of computation for minimal sufficient reasons holds for a *fixed* $\delta \in (0, 1]$? If so, does it hold for every fixed such a δ, or only for some? (4) Is it possible to extend the positive behavior of decision trees with bounded split number to free BDDs?

Acknowledgments

Barceló is funded by Fondecyt grant 1200967. Arenas and Barceló have been funded by ANID - Millennium Science Initiative Program - Code ICN17002. Barceló has also been funded by National Center for Artificial Intelligence CENIA FB210017, Basal ANID. Romero is funded by Fondecyt grant 11200956, the Data Observatory Foundation, and the National Center for Artificial Intelligence CENIA FB210017, Basal ANID. Subercaseaux is supported by NSF awards CCF-2015445, CCF1955785, and CCF2006953.

References

A. B. Arrieta, N. Díaz-Rodríguez, J. D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, June 2020. doi: 10.1016/j.inffus.2019.12.012. URL https://doi.org/10.1016/j.inffus.2019.12.012.

G. Audemard, S. Bellart, L. Bounia, F. Koriche, J. Lagniez, and P. Marquis. On the computational intelligibility of boolean classifiers. In M. Bienvenu, G. Lakemeyer, and E. Erdem, editors, *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021*, pages 74–86, 2021. doi: 10.24963/kr.2021/8. URL https://doi.org/10.24963/kr.2021/8.

P. Barceló, M. Monet, J. Pérez, and B. Subercaseaux. Model interpretability through the lens of computational complexity. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15487–15498. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/b1adda14824f50ef24ff1c05bb66faf3-Paper.pdf.

A. Biere, A. Biere, M. Heule, H. van Maaren, and T. Walsh. *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*. IOS Press, NLD, 2009. ISBN 1586039296.

A. Biere, K. Fazekas, M. Fleury, and M. Heisinger. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020. In T. Balyo, N. Froleyks, M. Heule, M. Iser, M. Järvisalo, and M. Suda, editors, *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, volume B-2020-1 of *Department of Computer Science Report Series B*, pages 51–53. University of Helsinki, 2020.

G. Blanc, J. Lange, and L.-Y. Tan. Provably efficient, succinct, and precise explanations. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=9UjRw5bqURS.

A. Choi, W. Shi, A. Shih, and A. Darwiche. Compiling neural networks into tractable boolean circuits. *intelligence*, 2017.

L. Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. A. Specter, and L. Kagal. Explaining explanations: An overview of interpretability of machine learning. In F. Bonchi, F. J. Provost, T. Eliassi-Rad, W. Wang, C. Cattuto, and R. Ghani, editors, *DSAA*, pages 80–89, 2018.

J. Goldsmith, M. Hagen, and M. Mundhenk. Complexity of DNF and isomorphism of monotone formulas. In *Mathematical Foundations of Computer Science 2005*, pages 410–421. Springer Berlin Heidelberg, 2005. doi: 10.1007/11549345_36. URL https://doi.org/10.1007/11549345_36.

X. Huang, Y. Izza, A. Ignatiev, and J. Marques-Silva. On efficiently explaining graph-based classifiers. In M. Bienvenu, G. Lakemeyer, and E. Erdem, editors, *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021*, pages 356–367, 2021. doi: 10.24963/kr.2021/34. URL https://doi.org/10.24963/kr.2021/34.

A. Ignatiev, N. Narodytska, N. Asher, and J. Marques-Silva. From contrastive to abductive explanations and back again. In *AIxIA 2020 – Advances in Artificial Intelligence*, pages 335–355. Springer International Publishing, 2021. doi: 10.1007/978-3-030-77091-4_21. URL https://doi.org/10.1007/978-3-030-77091-4_21.

A. Ignatiev, Y. Izza, P. J. Stuckey, and J. Marques-Silva. Using maxsat for efficient explanations of tree ensembles. In *AAAI*, 2022.

Y. Izza, A. Ignatiev, and J. Marques-Silva. On explaining decision trees. *CoRR*, abs/2010.11034, 2020a.

Y. Izza, A. Ignatiev, and J. Marques-Silva. On explaining decision trees, 2020b. URL https://arxiv.org/abs/2010.11034.

Y. Izza, A. Ignatiev, N. Narodytska, M. C. Cooper, and J. Marques-Silva. Efficient explanations with relevant sets. *ArXiv*, abs/2106.00546, 2021.

Y. Izza, A. Ignatiev, N. Narodytska, M. C. Cooper, and J. Marques-Silva. Provably precise, succinct and efficient explanations for decision trees, 2022. URL https://arxiv.org/abs/2205.09569.

Z. C. Lipton. The mythos of model interpretability. *Queue*, 16(3):31–57, 2018.

S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *NeurIPS*, pages 4765–4774, 2017.

J. Marques-Silva and A. Ignatiev. Delivering trustworthy ai through formal xai. In *AAAI*, 2022.

J. Marques-Silva, M. Janota, and A. Belov. Minimal sets over monotone predicates in boolean formulae. In N. Sharygina and H. Veith, editors, *Computer Aided Verification*, pages 592–607, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-39799-8.

J. Marques-Silva, T. Gerspacher, M. C. Cooper, A. Ignatiev, and N. Narodytska. Explaining naive bayes and other linear classifiers with polynomial time and delay. In *NeurIPS*, 2020.

J. Marques-Silva, T. Gerspacher, M. C. Cooper, A. Ignatiev, and N. Narodytska. Explanations for monotonic classifiers. In *ICML*, pages 7469–7479, 2021.

J. Marques-Silva, T. Gerspacher, M. C. Cooper, A. Ignatiev, and N. Narodytska. Explanations for monotonic classifiers. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 7469–7479. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/marques-silva21a.html.

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

M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI*, pages 1527–1535, 2018.

A. Shih, A. Choi, and A. Darwiche. A symbolic approach to explaining bayesian network classifiers. *arXiv preprint arXiv:1805.03364*, 2018.

C. Sinz. Towards an optimal CNF encoding of boolean cardinality constraints. In *Principles and Practice of Constraint Programming - CP 2005*, pages 827–831. Springer Berlin Heidelberg, 2005. doi: 10.1007/11564751_73. URL https://doi.org/10.1007/11564751_73.

S. Wäldchen, J. MacDonald, S. Hauch, and G. Kutyniok. The computational complexity of understanding binary classifier decisions. *J. Artif. Intell. Res.*, 70:351–387, 2021.

E. Wang, P. Khosravi, and G. V. den Broeck. Probabilistic sufficient explanations. In Z. Zhou, editor, *IJCAI*, pages 3082–3088, 2021.

I. Wegener. *Branching Programs and Binary Decision Diagrams*. Society for Industrial and Applied Mathematics, Jan. 2000. doi: 10.1137/1.9780898719789. URL https://doi.org/10.1137/1.9780898719789.

I. Wegener. Bdds–design, analysis, complexity, and applications. *Discret. Appl. Math.*, 138(1-2):229–251, 2004.

T. Yan and A. D. Procaccia. If you like shapley then you'll love the core. In *AAAI*, pages 5751–5759, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] The paper provides several negative results concerning the studied problems
   (c) Did you discuss any potential negative societal impacts of your work? [No] We do not foresee negative societal consequences of our work.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] Absolutely.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] All of them.
   (b) Did you include complete proofs of all theoretical results? [Yes] But due to the space constraints, the complete proofs are included in the appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]