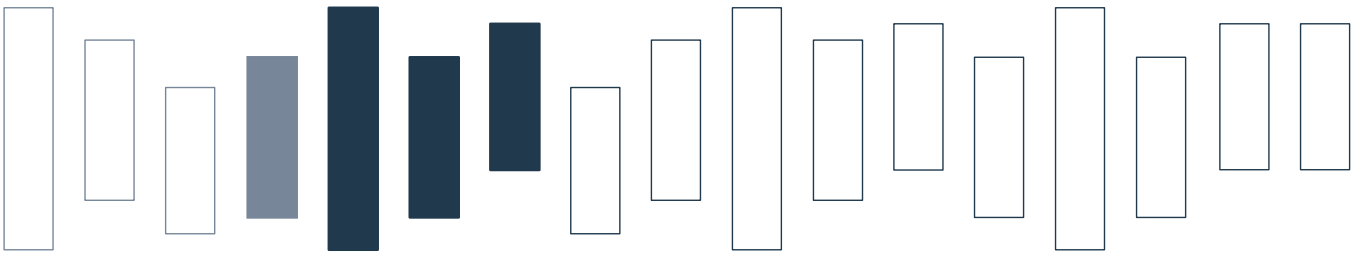
Technical Report AC-TR-21-002

January 2021

Parameterized Complexity of Small Decision Tree Learning

Sebastian Ordyniak and Stefan Szeider



This is the authors' copy of a paper that will appear in the proceedings of AAAI'21, the Thirty-Fifth AAAI Conference on Artificial Intelligence.

www.ac.tuwien.ac.at/tr

Parameterized Complexity of Small Decision Tree Learning

Sebastian Ordyniak[1] **and Stefan Szeider**[2]
[1]University of Leeds, School of Computing, Leeds, UK
[2]Algorithms and Complexity Group, TU Wien, Vienna, Austria

Abstract

We study the NP-hard problem of learning a decision tree (DT) of smallest depth or size from data. We provide the first parameterized complexity analysis of the problem and draw a detailed parameterized complexity map for the natural parameters: size or depth of the DT, maximum domain size of all features, and the maximum Hamming distance between any two examples. Our main result shows that learning DTs of smallest depth or size is fixed-parameter tractable (FPT) parameterized by the combination of all three of these parameters. We contrast this FPT-result by various hardness results that underline the algorithmic significance of the considered parameters.

1 Introduction

Decision Trees have proved to be extremely useful tools for the description, classification, and generalization of data (Larose 2005; Murthy 1998; Quinlan 1986). Because of their simplicity, decision trees are particularly attractive for providing interpretable models of the underlying data, an aspect whose importance has been stronly emphasized over the recent years (Darwiche and Hirth 2020; Doshi-Velez and Kim 2017; Goodman and Flaxman 2017; Lipton 2018; Monroe 2018). In this context, one prefers *small trees* (trees of small size or small depth), as they are easier to interpret and require fewer tests to make a classification. Small trees are also preferred in view of the parsimony principle (Occams Razor) since small trees are expected to generalize better to new data (Bessiere, Hebrard, and O'Sullivan 2009). However, learning small trees is computationally costly: it is NP-hard to decide whether a given data set can be represented by a decision tree of certain size or depth (Hyafil and Rivest 1976). In view of this complexity barrier, several constrained-based and SAT-based methods have been proposed for learning small decision trees (Bessiere, Hebrard, and O'Sullivan 2009; Narodytska et al. 2018; Avellaneda 2020; Schidler and Szeider 2021).

In this paper, we investigate the problem of finding small decision trees (w.r.t. size or depth) under the framework of *Parameterized Complexity* (Downey and Fellows 2013; Gottlob, Scarcello, and Sideri 2002; Niedermeier 2006). This framework allows us to achieve a more fine-grained and qualitative analysis, revealing properties of the input data in terms of *problem parameters* that provide runtime guarantees for decision tree learning algorithms. The key notion of Parameterized Complexity is *fixed-parameter tractability (FPT)* which generalizes the classical polynomial time tractability by allowing the running time to be exponential in a function of the problem parameters while remaining polynomial in the input size (we provide more detailed definitions in Section 2). Fixed-parameter tractability captures the scalability of algorithms to large inputs as long as the problem parameters remain small. Several key problems that arise in AI have been studied in terms of their fixed-parameter tractability, including Planning (Bäckström et al. 2012), SAT and CSP (Bessière et al. 2008; Gaspers et al. 2014), Computational Social Choice (Bredereck et al. 2017), Machine Learning (Ganian et al. 2018), and Argumentation (Dvorák, Ordyniak, and Szeider 2012).

For decision tree learning, we consider parameterizations of the following two fundamental NP-hard problems:

MINIMUM DECISION TREE SIZE (DTS): we are given a set of examples, labeled positive or negative, each over a set of features; each feature f ranges over a linearly ordered range of possible values, and an integer s (for *size*). The task is to decide whether there exists a decision tree with at most s tests or report that no such tree exists. Each test determines whether a certain feature is below a certain threshold or not.

MINIMUM DECISION TREE DEPTH (DTD) is defined similarly, where instead of the bound s on the total number of tests, a bound d (for *depth*) on the number of tests on any root-to-leaf path is provided.

For technical reasons, we define the problems as decision problems; however, all our algorithms are constructive in the sense that if they decide DTS or DTD affirmatively, then they can output a witnessing decision tree; otherwise, they report that a decision tree of the required size or depth does not exist.

For both problems, it is natural to include the *solution size* (i.e., s for DTS and d for DTD, respectively) as a parameter, since our objective is to learn decision trees where these values are small. Another natural parameter is the number of different values a feature ranges over, we refer to this number as the *maximum domain size* ($D_{\max}$). It is reasonable to assume this number to be small, as many features are even Boolean or have a small range (e.g., temperature in a city).

As a first result (Theorem 2), we show that taking solution size and maximum domain size as parameters alone does

Name	$\lvert E\rvert$	$\lvert F\rvert$	$\delta_{\max}$
Append-tis	106	531	14
Australian	690	1164	24
Backache	180	476	31
Car	1728	22	12
Cancer	683	90	18
Colic	368	416	43
Cleve	303	396	23
Haberman	300	93	6

Name	$\lvert E\rvert$	$\lvert F\rvert$	$\delta_{\max}$
Heart-statlog	270	382	23
Hepatitis	155	362	34
HouseVotes	435	17	16
Hungarian	294	331	24
New-thyroid	215	335	10
Promoters	106	335	106
Shuttle	14500	692	18
Spect	250	23	22

Table 1: Boolean classification instances from Narodytska et al. (2018) with their number of examples $|E|$, the number of features $|F|$, and their maximum difference $\delta_{\max}$. The instances originate from the UCI Machine Learning Repository (Dua and Graff 2017).

not yield fixed-parameter tractability), even if all the features are Boolean. The hardness follows by a reduction from the HITTING SET (HS) problem. Hence one needs an additional parameter that restricts the input. Following the principle of *deconstruction of intractability* (Komusiewicz, Niedermeier, and Uhlmann 2011; Fellows, Jansen, and Rosamond 2013), we observe that when the hitting set instance consist of sets of bounded size, then HS becomes fixed-parameter tractable. Bounded set size for HS corresponds to a bound on the number of features two examples with a different classification can disagree in. We call this number the *maximum difference* $\delta_{\max}$ of the classification instance. What makes this parameter even more attractive is that it is small for standard benchmark instances (see Table 1).

Our main positive result (Theorem 8) establishes that, indeed, adding the maximum difference as additional parameter to solution size and maximum domain size renders both problems, DTS and DTD, fixed-parameter tractable. The algorithm consists of two main parts, namely, identifying a small set of features used by an optimal decision tree (*feature selection*) and finding an optimal decision tree that uses only a small set of features. Both of these steps are non-trivial and provide interesting insights for our understanding of decision trees in general. For instance, the feature selection part shows that only a small subset of all features needs to be considered to find an optimal decision tree. Similarly, the second part of our algorithm also shows that the same holds true for the set of thresholds; while the number of thresholds in the input instance can grow with the input size, only a small subset of them needs to be considered to find an optimal decision tree.

The diagram in Figure 1 visualizes the parameterized complexity landscape spanned out by the three parameters under consideration (solution size, maximum domain size, and maximum difference). The diagram shows that, without maximum difference included in the parameterization, neither of the problems is fixed-parameter tractable.

Those parameterizations that do not yield fixed-parameter tractability can be split into two categories: one where the problems DTS and DTD are already NP-hard when the parameters are fixed to a constant (paraNP-hard, in terms of Parameterized Complexity), and those where the problems

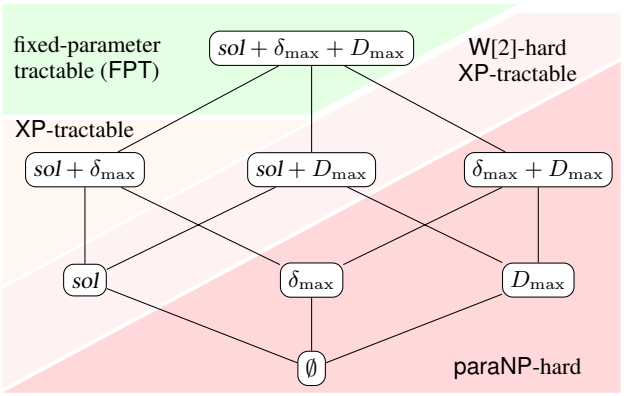


Figure 1: Diagram of the parameterized complexity of DTS and DTD when parameterized by subsets of $\{s, \delta_{\max}, D_{\max}\}$ and $\{d, \delta_{\max}, D_{\max}\}$, where $sol = s$ for DTS and $sol = d$ for DTD. The stated parameterized complexity results apply to all the combinations of parameters that are in the same area indicated by the same color.

are solvable in polynomial-time when the parameters are fixed to a constant (XP-tractable, in terms of Parameterized Complexity).

2 Preliminaries

Classification problems An *example* e is a function $e : \mathit{feat}(e) \to D$ defined on a finite set $\mathit{feat}(e)$ of *features* and a possibly infinite linearly ordered domain $D \subset \mathbb{Z}$. For a set E of examples, we put $\mathit{feat}(E) = \bigcup_{e \in E} \mathit{feat}(e)$. We say that two examples e_1, e_2 *agree* on a feature f if $f \in \mathit{feat}(e_1)$, $f \in \mathit{feat}(e_2)$ and $e_1(f) = e_2(f)$. If $f \in \mathit{feat}(e_1)$, $f \in \mathit{feat}(e_2)$ but $e_1(f) \neq e_2(f)$, we say that the examples *disagree on* f.

A *classification instance (CI)* (or just *instance*) $E = E^+ \uplus E^-$ is the disjoint union of two sets of examples, where for all $e_1, e_2 \in E$ we have $\mathit{feat}(e_1) = \mathit{feat}(e_2)$. The examples in E^+ are called *positive*, the examples in E^- are called *negative*. A set X of examples is *uniform* if $X \subseteq E^+$ or $X \subseteq E^-$; otherwise X is *non-uniform*.

Given a CI E, a subset $F \subseteq \mathit{feat}(E)$ is a *support set* of E if any two examples $e_1 \in E^+$ and $e_2 \in E^-$ disagree in at least one feature of F. Finding a support set of minimal size, denoted by $\mathrm{MSS}(E)$, for a classification instance E is an NP-hard task (Ibaraki, Crama, and Hammer 2011, Theorem 12.2). For two examples e and e' in E, we denote by $\delta(e, e')$ the set of features where e and e' disagree and we denote by $\delta_{\max}(E) = \max_{e^+ \in E^+ \wedge e^- \in E^-} |\delta(e^+, e^-)|$ the *maximum difference* between any non-uniform pair of examples. For a feature $f \in \mathit{feat}(E)$, we denote by $D_E(f)$ the set of domain values for f appearing in any example of E, i.e., $D_E(f) = \{\, e(f) \mid e \in E \,\}$ and we set $D_{\max}$ to be the maximum size of $D_E(f)$ over all features of E, i.e., $D_{\max} = \max_{f \in \mathit{feat}(E)} |D_E(f)|$. If $D_{\max} = 2$ the CI is called *Boolean* or a partially defined Boolean function (Ibaraki, Crama, and Hammer 2011).

We denote by $E[\alpha]$ the set of examples in E that agree with the assignment $\alpha : F' \to D$, where $F' \subseteq \mathit{feat}(E)$,

temp	rain	time	day	walk
37	1	10	3	E^-
68	0	60	2	E^+
53	1	60	4	E^-
53	0	15	2	E^-
60	0	60	5	E^+
51	0	40	3	E^+
71	0	35	5	E^+

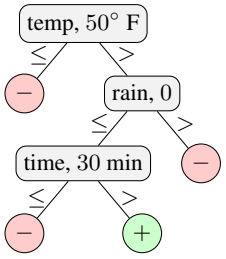


Figure 2: A CI with seven examples over four features (temp, rain, time, day), classifying on whether one walks to work or not, and a DT for it.

i.e., $E[\alpha] = \{\, e \mid e(f) = \alpha(f) \wedge f \in F' \,\}$. For the complexity analysis we set the *input size* $\|E\|$ of a CI E to $|E| \cdot (|\mathit{feat}(E)| + 1) \cdot \log D_{\max}$.

Decision trees A *decision tree* (DT) (or *classification tree*) is a rooted tree T with vertex set $V(T)$ and arc set $A(T)$, where each non-leaf node (called a *test*) $v \in V(T)$ is labeled with a feature $\mathit{feat}(v)$ and an integer *threshold* $\lambda(v)$, each non-leaf node v has exactly two outgoing arcs, a *left arc* and a *right arc*, and each leaf is either a *positive* or a *negative* leaf. We write $\mathit{feat}(T) = \{\, \mathit{feat}(v) \mid v \in V(T) \,\}$.

Consider a CI E and a decision tree T with $\mathit{feat}(T) \subseteq \mathit{feat}(E)$. For each node v of T we define $E_T(v)$ as the set of all examples $e \in E$ such that for each left (right, respectively) arc (u, w) on the unique path from the root of T to v we have $e(\mathit{feat}(u)) \leq \lambda(u)$ ($e(\mathit{feat}(u)) > \lambda(u)$, respectively). T *correctly classifies* an example $e \in E$ if e is a positive (negative) example and $e \in E_T(v)$ for a positive (negative) leaf. We say that T *classifies* E (or simply that T is a DT for E) if T correctly classifies every example $e \in E$. An example for a CI E and a possible DT for E are illustrated in Figure 2. The *size* of T, denoted by $|T|$, is the number of non-leaf nodes (tests) in T and the *depth* of T, denoted by $\mathit{dep}(T)$, is the maximum number of non-leaf nodes (tests) in any root-to-leaf path of T. DTS (and DTD) are now the problems of deciding whether for a given CI E and an integer s (d), there is a DT for E of size at most s (depth at most d).

Observation 1. *Let T be a DT for a CI E, then $\mathit{feat}(T)$ is a support set of E.*

Proof. Suppose for a contradiction that this is not the case and there is an example $e^+ \in E^+$ and an example $e^- \in E^-$ such that e^+ and e^- agree on all features in $\mathit{feat}(T)$. Therefore, e^+ and e^- are contained in the same leaf node of T, contradicting our assumption that T is a DT. □

2.1 Parameterized Complexity

We give some basic definitions of Parameterized Complexity and refer for a more in-depth treatment to other sources (Downey and Fellows 2013). Parameterized complexity considers problems in a two-dimensional setting, where a problem instance is a pair (I, k), where I is the main part and k is the parameter (k can be composed of several parameters, e.g., $k = k_1 + k_2$). A parameterized problem is *fixed-parameter tractable (FPT-tractable)* if there exists a computable function g such that instances (I, k) can be solved in time $g(k)\|I\|^{O(1)}$. A parameterized problem is XP-*tractable* if instances (I, k) can be solved in time $\|I\|^{g(k)}$. There are different shades of parameterized hardness. A problem is paraNP-hard if fixing the parameter to a constant gives an NP-hard problem. Many parameterized problems that are XP-tractable but not fixed-parameter tractable are contained in the complexity classes of the Weft hierarchy, W[1] $\subseteq$ W[2] $\subseteq \ldots$. Hardness for any of these classes provides a strong theoretical evidence that a problem is not fixed-parameter tractable. For DTS and DTD we consider mainly the three parameters solution size (s for DTS and d for DTD), maximum domain size $D_{\max}$, and maximum difference $\delta_{\max}$.

3 Hardness Results

In this section we show our hardness results for DTS and DTD, i.e., we will show that even restricted to Boolean instances both problems are W[2]-hard parameterized by s or d and they are paraNP-hard parameterized by $\delta_{\max}(E)$. Underlying our hardness results is the following (polynomial-time) reduction from the well-known HITTING SET problem, which given a family $\mathcal{F}$ of sets over some universe U and an integer k asks whether $\mathcal{F}$ has a *hitting set* of size at most k, i.e., a subset H of U of size at most k such that $F \cap H \neq \emptyset$ for every $F \in \mathcal{F}$. The *maximum arity* Δ of a HITTING SET instance is the size of a largest set in $\mathcal{F}$. For an instance $\mathcal{I} = (\mathcal{F}, U, k)$ of HITTING SET, we let $E(\mathcal{I})$ be the CI that has a (Boolean) feature for every element in U, one positive example p with $p(u) = 0$ for every $u \in U$ and one negative example n_F for every $F \in \mathcal{F}$ such that $n_F(u) = 1$ for every $u \in F$ and $n_F(u) = 0$ otherwise.

Theorem 2. DTS *parameterized by s and* DTD *parameterized by h are* W[2]*-hard, even for Boolean instances.*

Proof. We use the polynomial-time reduction from HITTING SET defined by $E(\mathcal{I})$ with $\mathcal{I} = (\mathcal{F}, U, k)$ together with the fact that HITTING SET is W[2]-hard parameterized by the size k of the hitting set (Downey and Fellows 2013). Note that a set H of features of $E(\mathcal{I})$ is a support set if and only if H is a hitting set for $\mathcal{I}$. Therefore, $\mathcal{I}$ has a hitting set of size at most k if and only if $E(\mathcal{I})$ has a support set of size k. Because $E(\mathcal{I})$ contains only one positive example, every support set S gives rise to a DT whose depth and size are equal to $|S|$. Together with Observation 1, this implies that $E(\mathcal{I})$ has a DT of depth/size at most k if and only if $E(\mathcal{I})$ has a support set of size at most k, which—as we showed above—is the case if and only if $\mathcal{I}$ has a hitting set of size at most k. □

The same reduction also shows that DTS and DTD are paraNP-hard parameterized by $\delta_{\max}(E)$, because $\delta_{\max}(E(\mathcal{I})) = \Delta(\mathcal{I})$ and HITTING SET is NP-complete even for $\Delta = 2$.

Theorem 3. DTS *and* DTD *are* paraNP*-hard parameterized by $\delta_{\max}(E)$.*

The reduction also shows paraNP-hardness for other parameters such as $\min_{\#}(E) = \min\{|E^+|, |E^-|\}$ and the number of branching nodes of the resulting DT; a node is a *branching node* if none of its children is a leaf. This is because $\min_{\#}(E(\mathcal{I})) = 1$ and therefore it suffices to consider only DTs for $E(\mathcal{I})$ with 0 branching nodes.

4 Algorithms

In this section, we will provide our algorithmic results for DTS and DTD. Namely, we will show that both problems are fixed-parameter tractable parameterized by solution size (s or d for DTS respectively DTD), $\delta_{\max}$, and $D_{\max}$. To complete the complexity map given in Figure 1, we will also show that both problems can be solved in polynomial-time when searching for a DT of constant size or depth. To simplify the presentation, we will provide the result only for DTS and defer the proof for DTD to the supplementary material. The algorithm for our main result, i.e., the FPT-result, consists of two main steps. In the first step, which we call *feature selection* and which is provided in Subsection 4.2, we show that one can efficiently enumerate all sets of features such that every optimal DT only uses the features from one of the sets. We then show that given such a set of features, which is small since it can have size at most s, it is possible to find an optimal DT that uses only those features. We start with the second step of the algorithm.

4.1 An Algorithm for Instances with Few Features

This subsection is devoted to a proof of the following theorem, showing that one can efficiently compute a decision tree T of minimum size that uses exactly the features of a given support set S for a CI E. Note that the theorem actually shows that finding such an optimal decision tree is FPT parameterized by *sol* only. This is strictly not required to show our main result (Theorem 8), however, we find the result surprising, interesting in its own right, and believe that it provides a first step towards giving an FPT-result solely for $\mathit{sol} + \delta_{\max}(E)$.

Theorem 4. *Let E be a CI, $S \subseteq \mathit{feat}(E)$ be a support set for E, and let s be an integer. Then, there is an algorithm that runs in time $2^{\mathcal{O}(s^2)}\|E\|^{1+o(1)} \log \|E\|$ and computes a DT of minimum size among all DTs T with $\mathit{feat}(T) = S$ and $\mathit{size}(T) \leq s$ if such a DT exists;otherwise* `nil` *is returned.*

First note that since $\mathit{feat}(T) = S$, we have that $k = |S| \leq s$ and therefore the size of S, i.e., the number of features that we need to consider for our DT T, is bounded in terms of our parameter s. This implies that we can enumerate both the tree underlying the DT T but also the possible assignments of the nodes of T to features (because there are at most k features) by brute-force; the details of this procedure can be found in Lemma 5. Unfortunately, the same is not true for obtaining the thresholds employed by the DT, since the number of possible thresholds is potentially as large as the input size. However, perhaps suprisingly Lemma 6 shows that it is possible to find the thresholds efficiently. We start by introducing the necessary notions and definitions.

We say that a tree T is a *pseudo DT* if T is a rooted ordered binary tree, where every non-leaf node of T has a left and a right child, i.e., T has the same structure as a decision tree. Furthermore, we say that a function α is a *feature assignment* for T, if α assigns a feature in $\mathit{feat}(E)$ to every non-leaf node v of T. Similarly, we say that a function γ is a *threshold assignment* for T (and α), if γ assigns a threshold in $D_E(\alpha(v))$ to every non-leaf node of T. Finally, we say that a pair (T, α), where T is a pseudo DT and α is a feature assignment for T can be *extended* to a DT for E, if there is a threshold assignment γ for T and α such that T together with $\mathit{feat} = \alpha$ and $\lambda = \gamma$ is a DT for E.

Our first lemma shows that we can efficiently enumerate all pseudo DTs and feature assignments if the size of the DT and the set of features that the DT is allowed to use is small. The lemma can essentially be shown by analysing the number of pseudo DTs and feature assignments.

Lemma 5. *Let E be a CI, $S \subseteq \mathit{feat}(E)$ be a support set for E with $k = |S|$, and let s be an integer with $s \geq k$. Then, there is an algorithm that enumerates all pairs (T, α) such that T is a pseudo DT of size at most s and α is a feature assignment for T with $\alpha(T) = S$ in time $\mathcal{O}(s^s)$.*

Our next lemma shows that we can efficiently compute a threshold assignment for a given pseudo DT and feature assignment.

Lemma 6. *Let E be a CI, let T be a pseudo DT of depth d and let α be a feature assignment for T. Then, there is an algorithm that runs in time $\mathcal{O}(2^{d^2/2}\|E\|^{1+o(1)} \log \|E\|)$ and decides whether the pair (T, α) can be extended to a DT for E.*

Proof Sketch. Let r be the root of T with $f = \alpha(r)$ having left child c_l and right child c_r. Then, (T, α) can be extended to a DT for E if and only if there is a threshold $t \in D_E(f)$ for f such that: (1) (T_{c_l}, α) can be extended to a DT for $E[(f \leq t)]$, where T_{c_l} is the subtree of T rooted at c_l and $E[(f \leq t)]$ is the set of all examples e in E with $f(e) \leq t$ and (2) (T_{c_r}, α) can be extended to a DT for $E[(f > t)]$, where $E[(f > t)]$ is the set of all examples $e \in E$ with $f(e) > t$. Therefore, the problem can be reduced to finding the threshold t for the feature f of the root r of T. Unfortunately, the number of possible thresholds, i.e., the size of $D_E(f)$ can be large and it is therefore not possible to try every possible threshold. However, as it turns out this is not necessary since the influence of the threshold on the existence of an extension is in a certain sense monotone. Namely, it holds that if (T_{c_l}, α) can be extended to a DT for $E[(f \leq t)]$, then it can also be extended to a DT for $E[(f \leq t')]$ for every $t' \leq t$. This is because $E[(f \leq t')] \subseteq E[(f \leq t)]$ and therefore every DT for $E[(f \leq t)]$ is also a DT for $E[(f \leq t')]$. An analogous property holds for the right child c_r of r, i.e., (T_{c_r}, α) can be extended to a DT for $E[(f > t)]$, then it can also be extended to a DT for $E[(f > t')]$ for every $t' \geq t$. Therefore, it suffices to find the maximum threshold $t \in D_E(f)$ such that (T_{c_l}, α) can be extended to a DT for $E[(f \leq t)]$ and then to check (only for this maximum threshold) that also (T_{c_r}, α)

can be extended to a DT for $E[(f > t)]$. The big advantage of this is that we can now use binary search to find the largest threshold. This idea leads to the recursive algorithm illustrated in Algorithm 1.

Algorithm 1 Algorithm to compute the threshold assignment for a pseudo DT and a feature assignment.

Input: CI E, pseudo DT T, feature assignment α for T
Output: threshold assignment λ for T such that T together with $\mathit{feat} = \alpha$ and λ is a DT for E; if no such function exists, `nil`

function FINDTH(E, T, α)
  $r \leftarrow$ "root of T"
  if r is a leaf **then**
    if E is not uniform **then**
      return `nil`
    return $()$ ▷ "()" is the empty assignment
  $f \leftarrow \alpha(r)$; $c_l, c_r \leftarrow$ "left child and right child of r"
  $t \leftarrow$ BINARYSEARCH(E, T, α, c_l, f)
  $\lambda_r \leftarrow$ FINDTH($E[f > t], T_{c_r}, \alpha$)
  if $\lambda_r =$`nil` **then**
    return `nil`
  $\lambda_l \leftarrow$ FINDTH($E[f \leq t], T_{c_l}, \alpha$)
  return $(f = t) \cup \lambda_l \cup \lambda_r$

Algorithm 2 Sub-routine BinarySearch used by Algorithm 1.

Input: CI E, pseudo DT T, feature assignment α for T, feature f of the root of T, left child c_l of the root of T
Output: the largest threshold t in $D_E(f)$ for f such that (T_{c_l}, α) can be extended to a DT for $E[f \leq t]$; if no such threshold exists output the smallest value in $D_E(f)$ minus 1

function BINARYSEARCH(E, T, α, f, c_l)
  $D \leftarrow$ "array containing all elements in $D_E(f)$ in ascending order"
  $L \leftarrow 0$; $R \leftarrow |D_E(f)| - 1$; $b \leftarrow 0$
  while $L \leq R$ **do**
    $m \leftarrow \lfloor (L + R)/2 \rfloor$
    if FINDTH($E[f \leq D[m]], T_{c_l}, \alpha$) $\neq$ `nil` **then**
      $L \leftarrow m + 1$; $b \leftarrow 1$
    else
      $R \leftarrow m - 1$; $b \leftarrow 0$
  if $b = 1$ **then**
    return $D[m]$
  return $D[m - 1]$ ▷ assuming that $D[-1] = D[0] - 1$

The run-time of Algorithm 1 can be obtained as the number of recursive calls to **findTH** times the time required for one recursive call. The former can be bounded by $\mathcal{O}((\log \|E\|)^d)$ given that **findTH** calls itself at most $\log \|E\| + 2$ times for a decision tree of smaller depth. Together with the fact that one recursive call needs time at most $\mathcal{O}(\|E\| \log \|E\|)$, we obtain the stated run-time. □

Proof Sketch of Theorem 4. We first use Lemma 5 to enumerate all pairs (T, α) such that T is a pseudo DT of size at most s and α is a feature assignment for T with $\alpha(T) = S$ in time $\mathcal{O}(s^s)$. Then, for every such pair (T, α), we employ Lemma 6, to check whether the pair can be extended to a DT for E. Finally, we return the DT T with $\mathit{feat} = \alpha$ and $\lambda = \gamma$, where T is the smallest DT such that (T, α) can be extended to a DT for E using the threshold assignment γ or we return `nil` if no such pair (T, α) exists. The total run-time of the algorithm is the time required by the algorithm of Lemma 5 ($\mathcal{O}(s^s)$) times the time required by the algorithm of Lemma 6 ($\mathcal{O}(2^{d^2/2}(\|E\|)^{1+o(1)} \log \|E\|)$, where $d \leq s$). Since this expression is dominated by the second term, the total run-time is $\mathcal{O}(2^{s^2/2}(\|E\|)^{1+o(1)} \log \|E\|)$. □

4.2 Feature Selection

Because of Theorem 4, we know that the problems can be solved efficiently once we identified the right set of features. Since they are at most $\mathcal{O}(|\mathit{feat}(E)|^s)$ possible sets of features for a DT of size at most s, we can already employ Theorem 4 to solve DTS in polynomial-time if s is considered to be constant.

Theorem 7. DTS *and* DTD *are* XP*-tractable parameterized by either s or s.*

We show next that we can perform the feature selection step significantly faster if $D_{\max} + \delta_{\max}$ are small, which together with Theorem 4 will establish our FPT-result for DTS.

Theorem 8. DTS *and* DTD *are fixed-parameter tractable parameterized by $D_{\max} + \delta_{\max}(E) + s$ respectively $D_{\max} + \delta_{\max}(E) + d$.*

We start by showing that, given a CI E and an integer k, it is possible to enumerate all (inclusion-wise) minimal support sets of size at most k for E. The result is based on a well-known result for HITTING SET.

Corollary 9. *Let E be a CI and let k be an integer. Then there is an algorithm that in time $\mathcal{O}(\delta_{\max}(E)^k |E|)$ enumerates all (of the at most $\delta_{\max}(E)^k$) minimal support sets of size at most k for E.*

Because of Observation 1, it holds that the set of features of any DT forms a support set. One might be tempted to think that any DT of minimum size or depth uses only features contained in some inclusion-wise minimal support set and that one could use this to find the set of variables used by an optimal decision tree by enumerating over all minimal support sets of a certain size using Corollary 9. Unfortunately, the following lemma shows that this is not the case.

Lemma 10. *There is a CI E_D such that $\mathit{feat}(T)$ is not a minimal support set for any decision T of minimum depth. Similarily, there is a CI E_S such that $\mathit{feat}(T)$ is not a minimal support set for any decision T of minimum size.*

Proof Sketch. Figure 3 shows a simple example for E_D on five Boolean variables a_1, a_2, b_1, b_2, and x. Note that the set $S = \{a_1, a_2, b_1, b_2\}$ is the only minimal support set for E_D. It is now straightforward to show that every DT for E_D that uses only the features S has depth at least 4; such a DT is illustrated in Figure 3. However, E_D has a DT of depth at most 3 that additionally uses the variable x. The supplementary material contains the construction for E_S. □

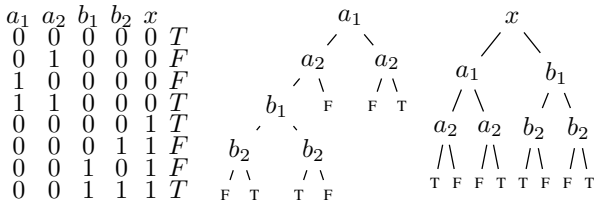


Figure 3: The CI E_D used in Lemma 10 to show that the set of features used by a DT of minimum depth is not necessarily a minimal support set. E_D has only one minimal support set, i.e., the set $M = \{a_1, a_2, b_1, b_2\}$. The DT in the centre shows a DT of minimum depth for E_D that uses only the features in M and the DT on the right shows the DT of minimum depth for E_D.

Therefore, to solve DTS it is not sufficient to enumerate minimal support sets. Nevertheless, because of Observation 1, we know that every DT T contains some minimal support set, say S. Moreover, as we will show next, if T has minimum size , then the set of features that T uses in addition to the features in S (i.e., features in $\mathit{feat}(T) \setminus S$) has a special property, which can be exploited to enumerate all sets of features that can be used by any DT of minimum size or depth. Namely, we say that a set R of features is *useful* for a given support set S of E if for every assignment $\beta : R \to D$, there is an assignment $\alpha : S \to D$ with $E[\alpha] \neq \emptyset$ but $E[\alpha \cup \beta] = \emptyset$. Informally, R is not useful for S if at least one assignment of R leaves all equivalence classes of S intact, where an equivalence class contains all examples that have the same value for all features in S.

Lemma 11. *Let T be a DT of minimum size for E and let S be a support set contained in $\mathit{feat}(T)$ (which exists due to Observation 1). Then, the set $R = \mathit{feat}(T) \setminus S$ is useful.*

Proof. If $R = \emptyset$, then there is nothing to show. Hence, assume that $R \neq \emptyset$ and suppose for a contradiction that the statement of the lemma does not hold. Then, there is an assignment $\beta : R \to D$ such that $E[\alpha \cup \beta] \neq \emptyset$ for every assignment $\alpha : S \to D$ with $E[\alpha] \neq \emptyset$. We will show a contradiction to our assumption that T has minimum size. To achieve this, consider the tree T' obtained from T as follows. (1) For every node t in T with $f = \mathit{feat}(t) \in R$ and $\lambda = \lambda(t)$, let t' be the left child of t if $\beta(f) > \lambda$ and let t' be the right child of t otherwise. Remove the subtree rooted at t' from T and let T'' be the tree obtained from T after applying this step exhaustively. Note that every node t of T'' with $\mathit{feat}(t) \in R$ has exactly one child node. (2) Let P be a path of maximum length in T'' consisting only of nodes t with $\mathit{feat}(t) \in R$. We obtain the tree $T''|P$ by contracting the path P in T, i.e., we remove all nodes on P from T'' and add an edge between the unique parent and the unique child of the two endpoints of P in T''. Then T' is the tree obtained from T'' after exhaustively contracting all maximal paths P in T'' that only consists of nodes t with $\mathit{feat}(t) \in R$.

Note that every non-leaf node t in T' has exactly two children and moreover $\mathit{feat}(t) \in S$. We now show that T' is also a DT for E, which, because $R \neq \emptyset$, contradicts our assumption that T had minimum size. Suppose for a contradiction that T' is not a DT. Then there is a leaf l of T' such that $E_{T'}(l)$ is not uniform, i.e., it contains at least one example from e^+ from E^+ and at least one example e^- from E^-. Let $\alpha^+ : S \to D$ be the unique assignment agreeing with e^+ on S, i.e., $\alpha^+(f) = e^+(f)$ for every $f \in S$, and similarly let $\alpha^- : S \to D$ be the unique assignment agreeing with e^- on S. Then, $E[\alpha^+]$ and $E[\alpha^-]$ are non-empty, $E[\alpha^+], E[\alpha^-] \subseteq E_{T'}(l)$, and moreover because S is a support set the former only contains positive examples while the latter only contains negative examples. But then $E[\alpha^+ \cup \beta]$ and $E[\alpha^- \cup \beta]$ are also non-empty and are contained in $E_T(l')$ for the unique leaf l' in T corresponding to l in T'. Therefore, $E_T(l')$ is not uniform, contradicting our assumption that T is a DT for E. □

We say that a set R_0 is a *branching set* for S if every useful set R for S contains at least one feature in R_0. To exploit Lemma 11, we will show next that we can efficiently compute a small branching set for S. Consider an arbitrary assignment $\beta : R \to D$ for some useful set R for S. Then, because R is useful, there must be an assignment $\alpha : S \to D$ with $E[\alpha] \neq \emptyset$ such that $E[\alpha \cup \beta] = \emptyset$. In other words, for every example e in $E[\alpha]$, the set R must contain at least one feature f for which e disagrees with β. Therefore, every useful set has to contain at least one feature from the set $\delta(e, \beta)$, i.e., the set of features where e disagrees with β.

Lemma 12. *Let $E(S)$ be an arbitrary set of examples that contains exactly one example from $E[\alpha]$ for every assignment $\alpha : S \to D$ with $E[\alpha] \neq \emptyset$. Then, every useful set R for S has to contain at least one feature from the set $\bigcup_{e \in E(S)} \delta(e, \beta)$ for every $\beta : R \to D$.*

Unfortunately, the set $\delta(e, \beta)$ for some $e \in E(S)$ can be arbitrary large in general (i.e., cannot be bounded by our parameters). However, the following lemma shows that there is a "global assignment" $\gamma : \mathit{feat}(E) \to D$ such that every example in E disagrees with γ in at most $2\delta_{\max}(E)$ features.

Lemma 13. *Let E be a CI. Then, there is an assignment $\gamma : \mathit{feat}(E) \to D$ such that every example e in E disagrees with γ in at most $2\delta_{\max}(E)$ features. Moreover, γ can be computed in polynomial-time.*

Proof Sketch. Setting $\gamma(f) = e(f)$ for an arbitary example $e \in E$ is sufficient because any example in E can differ from e in at most $2\delta_{\max}(E)$ features. □

Therefore, after setting β in Lemma 12 equal to the global assignment γ from Lemma 13, we obtain that the set $\{\, \delta(e, \beta) \mid e \in E(S) \,\}$ contains at most $2\delta_{\max}(E)$ features for all (of the at most $D^{|S|}_{\max}$) examples in $E(S)$.

Lemma 14. *There is a polynomial-time algorithm that given a support set S computes a branching set R_0 for S of size at most $D^{|S|}_{\max} 2\delta_{\max}(E)$.*

Proof Sketch of Theorem 8. We start by presenting the algorithm for DTS, which is illustrated in Algorithm 3 and Algorithm 4. The algorithm for DTD is similar and can be found in the supplementary material.

Algorithm 3 Main method for finding a DT of minimum size.

Input: CI E and integer s
Output: DT for E of minimum size (among all DTs of size at most s) if such a DT exists, otherwise `nil`

1: **function MINDT**(E, d)
2: "compute γ using Lemma 13"
3: $\mathcal{S} \leftarrow$ "set of all minimal support sets for E of size at most s"
4: $B \leftarrow$ `nil`
5: **for** $S \in \mathcal{S}$ **do**
6: $T \leftarrow$ MINDTS(E, s, S)
7: **if** ($T \neq$ `nil`) and ($B =$ `nil` or $|B| > |T|$) **then**
8: $B \leftarrow T$
9: **if** $B \neq$ `nil` and $|B| \leq s$ **then**
10: **return** B
11: **return** `nil`

Algorithm 4 Method for finding a DT of minimum size using at least the features in a given support set S.

Input: CI E, integer s, support set S for E with $|S| \leq s$
Output: DT of minimum size among all DTs T for E of size at most s such that $S \subseteq \mathit{feat}(T)$; if no such DT exists, `nil`

1: **function MINDTS**(E, s, S)
2: $B \leftarrow$ "compute a DT of minimum size for E using exactly the features in S using Theorem 4"
3: $R_0 \leftarrow$ "compute the branching set R_0 for S using Lemma 14"
4: **for** $f \in R_0$ **do**
5: $T \leftarrow$ MINDTS(E, s, $S \cup \{f\}$)
6: **if** $T \neq$ `nil` and $|T| < |B|$ **then**
7: $B \leftarrow T$
8: **if** $|B| \leq s$ **then**
9: **return** B
10: **return** `nil`

Given a CI E and an integer s, the algorithm returns a DT of minimum size among all DTs of size at most s if such a DT exists and otherwise the algorithm returns `nil`. The algorithm starts by computing the global assignment γ in polynomial-time using Lemma 13. The algorithm then computes the set $\mathcal{S}$ of all minimal support sets for E of size at most s, which because of Corollary 9 results in a set $\mathcal{S}$ of size at most $(\delta_{\max}(E))^s$. In Line 5 the algorithm then interates over all sets S in $\mathcal{S}$ and calls the function **minDTS** given in Algorithm 4 for E, s, and S, which returns a DT of minimum size among all DTs T for E of size at most s such that $S \subseteq \mathit{feat}(T)$. It then updates the currently best decision tree B if necessary with the DT found by the function **minDTS**. Moreover, if the best DT found after going through all sets in $\mathcal{S}$ has size at most s, it is returned (in Line 10), otherwise the algorithm returns `nil`. Finally, the function **minDTS** given in Algorithm 4 does the following. It first computes a DT T of minimum size that uses exactly the features in S using Theorem 4. It then tries to improve upon T with the help of useful sets. That is, it uses Lemma 14 to compute the branching set R_0. It then interates over all (of the at most $D_{\max}^{|S|} 2\delta_{\max}(E)$) features $v \in R_0$ (using the for-loop in Line 4), and calls itself recursively on the feature set $S \cup \{v\}$. If this call finds a smaller DT, then the current best DT B is updated. Finally, after the for-loop the algorithm either returns B if its size is less then s or `nil` otherwise.

Towards showing the correctness of Algorithm 3, consider the case that E has a DT of size at most s and let T be a such a DT of minimum size. Because of Observation 1, $\mathit{feat}(T)$ is a support set for E and therefore $\mathit{feat}(T)$ contains a minimal support set S of size at most s. Because the for-loop in Line 5 of Algorithm 3 interates over all minimal support sets of size at most s for E, it follows that Algorithm 4 is called with parameters E, s, and S. If $\mathit{feat}(T) = S$, then B is set to a DT for E of size $|T|$ in Line 2 of Algorithm 4 and the algorithm will output a DT of size at most $|T|$ for E. If, on the other hand, $\mathit{feat}(T) \setminus S \neq \emptyset$, then because T has minimum size and S is a support set for E with $S \subseteq \mathit{feat}(T)$, we obtain from Lemma 11 that the set $R = \mathit{feat}(T) \setminus S$ is useful for S. Therefore, because of Lemma 14, R has to contain a variable v from the set R_0 computed in Line 3. It follows that Algorithm 4 is called with parameters E, s, and $S \cup \{v\}$. From now onwards the argument repeats and since $R_0 \neq \emptyset$ the process stops after at most $s - |S|$ recursive calls after which a DT for E of size at most $|T|$ will be computed in Line 2 of Algorithm 4. Finally, it is easy to see that if Algorithm 3 outputs a DT T, then it is a valid solution. This is because, T must have been computed in Line 2 of Algorithm 4, which implies that T is a DT for E. Moreover, T has size at most s, because of Line 9 in Algorithm 3. The run-time of the algorithm can be obtained by noting that the algorithm can be seen as a branching algorithm with recursion depth at most s and at most $D_{\max}^{s} 2\delta_{\max}(E)$ branches per call, using at most $2^{\mathcal{O}(s^2)} \|E\|^{1+o(1)} \log \|E\|$ time per call. □

5 Conclusion

We have carried out the first analysis of the parameterized complexity of learning DTs of small depth or size, providing an almost comprehensive map of this problem's parameterized complexity w.r.t. natural parameters. While our algorithmic results open up the opportunity for the development of practically more efficient exact (or heuristic) algorithms, our lower-bound results provide a formal and rigorous justification for the use of heuristics or approximation algorithms. Moreover, the developed techniques provide novel and useful insights into the problem of learning DTs.

For future work, we leave the only remaining open question from our analysis: whether our FPT-result still holds without the parameter $D_{\max}$. We conjecture that this is indeed the case, since the second part of our algorithm (i.e., learning the DT and thresholds for a small set of features) is already FPT parameterized by solution size and $\delta_{\max}$ alone.

We hope that our work stimulates further investigations on the parameterized complexity of DT learning, for instance by considering other structural parameters that have previously been employed for various problems in AI, including decompositional parameters (Gottlob, Pichler, and Wei 2006) and backdoor sets size (Gaspers and Szeider 2012).

Acknowledgements

Stefan Szeider acknowledges support by the Austrian Science Fund (FWF, project P32441) and the Vienna Science and Technology Fund (WWTF, project ICT19-065). Sebastian Ordyniak acknowledges support from the Engineering and Physical Sciences Research Council (EPSRC, project EP/V00252X/1).

References

Avellaneda, F. 2020. Efficient Inference of Optimal Decision Trees. In *Proceedings of the The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI-20*. AAAI Press, Palo Alto, California USA.

Bäckström, C.; Chen, Y.; Jonsson, P.; Ordyniak, S.; and Szeider, S. 2012. The Complexity of Planning Revisited - A Parameterized Analysis. In Hoffmann, J.; and Selman, B., eds., *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*. AAAI Press.

Bessière, C.; Hebrard, E.; Hnich, B.; Kiziltan, Z.; Quimper, C.-G.; and Walsh, T. 2008. The Parameterized Complexity of Global Constraints. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, 235–240. AAAI Press.

Bessiere, C.; Hebrard, E.; and O'Sullivan, B. 2009. Minimising Decision Tree Size as Combinatorial Optimisation. In Gent, I. P., ed., *Principles and Practice of Constraint Programming - CP 2009*, 173–187. Berlin, Heidelberg: Springer Berlin Heidelberg.

Bredereck, R.; Chen, J.; Niedermeier, R.; and Walsh, T. 2017. Parliamentary Voting Procedures: Agenda Control, Manipulation, and Uncertainty. *J. Artif. Intell. Res.* 59: 133–173. doi:10.1613/jair.5407. URL https://doi.org/10.1613/jair.5407.

Darwiche, A.; and Hirth, A. 2020. On The Reasons Behind Decisions. In *ECAI 2020*. To appear.

Doshi-Velez, F.; and Kim, B. 2017. A Roadmap for a Rigorous Science of Interpretability. *CoRR* abs/1702.08608. URL http://arxiv.org/abs/1702.08608.

Downey, R. G.; and Fellows, M. R. 2013. *Fundamentals of parameterized complexity*. Texts in Computer Science. Springer Verlag.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml.

Dvorák, W.; Ordyniak, S.; and Szeider, S. 2012. Augmenting tractable fragments of abstract argumentation. *Artificial Intelligence* 186: 157–173.

Fellows, M. R.; Jansen, B. M.; and Rosamond, F. 2013. Towards fully multivariate algorithmics: Parameter ecology and the deconstruction of computational complexity. *European J. Combin.* 34(3): 541–566.

Ganian, R.; Kanj, I.; Ordyniak, S.; and Szeider, S. 2018. Parameterized Algorithms for the Matrix Completion Problem. In *Proceeding of ICML, the Thirty-fifth International Conference on Machine Learning, Stockholm, July 10–15, 2018*, 1642–1651. JMLR.org. ISSN: 1938-7228.

Gaspers, S.; Misra, N.; Ordyniak, S.; Szeider, S.; and Zivny, S. 2014. Backdoors into Heterogeneous Classes of SAT and CSP. In Brodley, C. E.; and Stone, P., eds., *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, 2652–2658. AAAI Press.

Gaspers, S.; and Szeider, S. 2012. Backdoors to Satisfaction. In Bodlaender, H. L.; Downey, R.; Fomin, F. V.; and Marx, D., eds., *The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, volume 7370 of *Lecture Notes in Computer Science*, 287–317. Springer Verlag.

Goodman, B.; and Flaxman, S. R. 2017. European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation". *AI Magazine* 38(3): 50–57.

Gottlob, G.; Pichler, R.; and Wei, F. 2006. Bounded Treewidth as a Key to Tractability of Knowledge Representation and Reasoning. In *21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference*. AAAI Press.

Gottlob, G.; Scarcello, F.; and Sideri, M. 2002. Fixed-parameter complexity in AI and nonmonotonic reasoning. *Artificial Intelligence* 138(1-2): 55–86.

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

Ibaraki, T.; Crama, Y.; and Hammer, P. L. 2011. *Partially defined Boolean functions*, 511563. Encyclopedia of Mathematics and its Applications. Cambridge University Press.

Komusiewicz, C.; Niedermeier, R.; and Uhlmann, J. 2011. Deconstructing intractability - A multivariate complexity analysis of interval constrained coloring. *J. Discrete Algorithms* 9(1): 137–151.

Larose, D. T. 2005. *Discovering knowledge in data*. Wiley-Interscience [John Wiley & Sons], Hoboken, NJ. ISBN 0-471-66657-2. An introduction to data mining.

Lipton, Z. C. 2018. The mythos of model interpretability. *Communications of the ACM* 61(10): 36–43.

Monroe, D. 2018. AI, explain yourself. *AI Communications* 61(11): 11–13. doi:10.1145/3276742. URL https://doi.org/10.1145/3276742.

Murthy, S. K. 1998. Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey. *Data Min. Knowl. Discov.* 2(4): 345–389. doi:10.1023/A:1009744630224. URL https://doi.org/10.1023/A:1009744630224.

Narodytska, N.; Ignatiev, A.; Pereira, F.; and Marques-Silva, J. 2018. Learning Optimal Decision Trees with SAT. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, 1362–1368. International Joint Conferences on Artificial Intelligence Orga-

nization. doi:10.24963/ijcai.2018/189. URL https://doi.org/10.24963/ijcai.2018/189.

Niedermeier, R. 2006. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and its Applications. Oxford: Oxford University Press.

Quinlan, J. R. 1986. Induction of Decision Trees. *Machine Learning* 1(1): 81–106. doi:10.1023/A:1022643204877. URL https://doi.org/10.1023/A:1022643204877.

Schidler, A.; and Szeider, S. 2021. SAT-based Decision Tree Learning for Large Data Sets. In *Proceedings of AAAI'21, the Thirty-Fifth AAAI Conference on Artificial Intelligence*. AAAI Press.