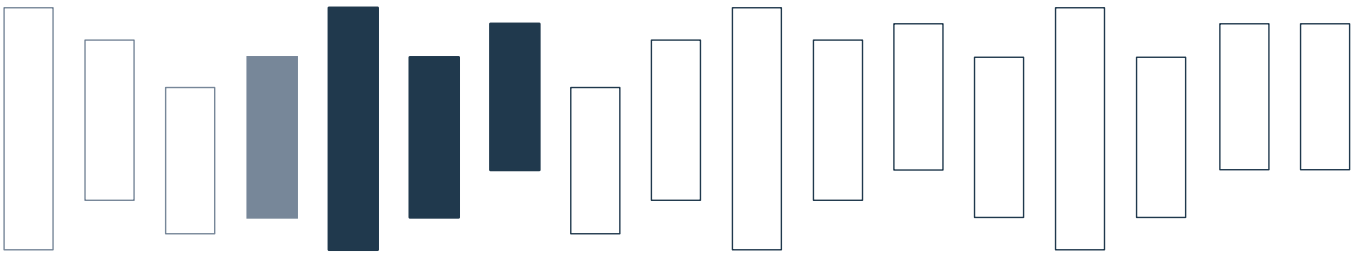
Technical Report AC-TR-21-001

January 2021

Backdoor DNFS

Sebastian Ordyniak, André Schidler, and Stefan Szeider



Backdoor DNFs

Sebastian Ordyniak[1] , **André Schidler**[2] , **Stefan Szeider**[2]
s.ordyniak@leeds.ac.uk, {aschidler,sz}@ac.tuwien.ac.at

Abstract

We introduce backdoor DNFs, as a tool to measure the theoretical hardness of CNF formulas. Like backdoor sets, backdoor DNFs are defined relative to a tractable class of CNF formulas. Each conjunctive term of a backdoor DNF defines a partial assignment which moves the input CNF formula into the base class. Backdoor DNFs are more expressive and potentially smaller than its predecessors backdoor sets and backdoor trees. We establish the fixed-parameter tractability of the backdoor DNF detection problem. Our result holds for the fundamental base classes Horn and 2CNF, and their combination. We complement our theoretical findings by an empirical study. Our experiments show that backdoor DNFs provide a significant improvement over their predecessors.

1 Introduction

Over the last two decades, the progress on practical SAT solving has been "nothing short of spectacular" [Vardi, 2014]. State-of-the-art SAT solvers routinely solve instances with millions of clauses and variables. This is in stark contrast to the theoretical intractability of SAT. The problem is not just NP-complete [Cook, 1971]; the Exponential-Time Hypothesis [Impagliazzo *et al.*, 2001], a standard complexity-theoretic assumption, excludes the existence of an algorithm that solves an n-variable 3SAT instance with $2^{o(n)}$ steps. This apparent discrepancy between theory and practice is often explained by the presence of a "hidden structure" in real-world SAT instances, which is implicitly exploited by the SAT solver. Several approaches have been proposed in the literature to make the vague notion of a hidden structure precise, including modularity [Ansótegui *et al.*, 2014; Newsham *et al.*, 2014; Ganian and Szeider, 2015] and decomposability [Mateescu, 2011; Jamali and Mitchell, 2017; Ganian and Szeider, 2017]. The notion of a *backdoor set*, introduced by Williams *et al.* [2003], provides another way of capturing the existence of a hidden structure in a SAT instance. The idea is to fix a polynomial-time solvable base class $\mathcal{C}$ of CNF formulas (either defined by a polynomial-time subsolver or by a syntactic property such as Horn). We then measure the existence of hidden structure within a SAT instance in terms of the number of variables one needs to instantiate to put the instance into the base class $\mathcal{C}$. The instantiated variables form a backdoor set. One distinguishes between a *weak* backdoor (there exists an instantiation of the backdoor variables that produces a satisfiable instance that belongs to $\mathcal{C}$) and a *strong* backdoor (all instantiations for the backdoor variables result in an instance that belongs to $\mathcal{C}$). This paper shall focus on strong backdoors since weak backdoors exist only for satisfiable formulas.

Suppose we know a size-k backdoor set of a SAT instance F. In that case we can decide its satisfiability by deciding the satisfiability of at most 2^k instances that belong to the tractable base class $\mathcal{C}$, i.e., in time $2^k\|F\|^{O(1)}$. Thus, SAT is *fixed-parameter tractable* (FPT) in the backdoor size if a witnessing backdoor is known. Therefore, it is interesting whether it is also fixed-parameter tractable to find a backdoor set of size k (the *backdoor set detection* problem). The systematic study of the parameterized complexity of backdoor set detection was initiated by Nishimura *et al.* [2004]. They showed that backdoor set detection is FPT for the fundamental base classes Horn and 2CNF. Gaspers and Szeider [2012] survey further results.

As stated above, a backdoor set of size k reduces the given SAT instance to at most 2^k tractable formulas in $\mathcal{C}$. However, 2^k is just a worst-case upper bound, which can be reduced in many cases. Thus, the size of a backdoor set is only a very coarse measure for a backdoor set's quality Samer and Szeider [2008] proposed a more refined measure. They introduced *backdoor trees*, which are decision trees on the backdoor variables, where each leaf corresponds to an instance in $\mathcal{C}$. The number of leaves of a backdoor tree over a backdoor set of size k is a more refined quality measure for a backdoor set. It ranges between the linear best-case lower bound of $k + 1$ and the exponential worst-case upper bound of 2^k. Interestingly, a backdoor tree with the smallest number of leaves is not necessarily based on a backdoor set of the smallest cardinality. Samer and Szeider [2008] showed that the detection of backdoor trees with respect to the fundamental bases classes Horn and 2CNF is fixed-parameter tractable when parameterized by the number of leaves of the backdoor tree. They implicitly assumed that the variables used by a backdoor tree form a subset-minimal backdoor set.

This paper proposes a new quality measure for backdoor

sets, which can again be significantly smaller than the number of leaves of a backdoor tree. The new measure is based on a *backdoor DNF* for a CNF formula F, a tautological propositional DNF formula D over the variables of a backdoor set. Each term of D, considered as a partial assignment, moves F into the base class $\mathcal{C}$. We observe that a backdoor tree can be considered a special case of a backdoor DNF when we identify each leaf with the term assignments made on the unique path from the root. We show that the difference between a smallest backdoor tree and a smallest backdoor tree as found by the known algorithm [Samer and Szeider, 2008], as well as between a smallest backdoor tree and a smallest backdoor DNF, can be arbitrarily large (Theorems 2 and 1). As our main theoretical contribution (Theorem 3), we show the following:

The detection of backdoor DNFs and backdoor trees with respect to the fundamental base classes Horn, AntiHorn, and 2CNF is fixed-parameter tractable, parameterized by the number of terms (for backdoor DNFs) or the number of leaves (for backdoor trees).

In this result, we are not limited to backdoor DNFs over a subset-minimal backdoor set. We show that such a limitation prevents us from finding backdoor DNFs/trees with the smallest number of terms/leaves. This strengthens the above mentioned result by Samer and Szeider [2008], who showed this for cardinality-minimal backdoor sets. Consequently, our FPT algorithm needs to be considerably more sophisticated to cover the general case. Although we still start the search with subset-minimal backdoor sets, we have to systematically explore extensions that lead to a smallest backdoor DNF or backdoor tree, respectively.

Our FPT algorithm also works for *heterogeneous base classes* [Gaspers *et al.*, 2017a]. Different terms of a backdoor DNF may lead to instances that belong to different tractable base classes Horn and 2CNF, or AntiHorn and 2CNF. However, we show that similar to the detection of backdoor sets, one cannot combine Horn and AntiHorn, for a fixed-parameter tractable detection of backdoor trees or backdoor DNFs (Theorem 4).

We complement the theoretical results with an empirical evaluation. We compare the size of backdoor trees and backdoor DNFs over a wide range of SAT instances. We utilize SAT encoding for the detection of these structures, as well as an efficient SAT-based algorithm for the extraction of minimal unsatisfiable cores. Our experiments show that in all considered instances, the backdoor DNFs are significantly smaller than backdoor trees. In many cases, the difference is of several orders of magnitude, which exceeds the expectation based on our theoretical results.

2 Preliminaries

We refer to the standard books for a basic overview of parameterized complexity theory [Cygan *et al.*, 2015], and assume that readers are aware of the complexity classes FPT, XP, and W[1].

CNF and DNF formulas We consider propositional formulas in conjunctive normal form (CNF) and disjunctive normal form (DNF) represented by sets of clauses, or sets of terms, respectively; e.g., $F = \{\{x, \neg y\}, \{\neg x, z\}\}$ represents both, the CNF formula $C = (x \vee \neg y) \wedge (\neg x \vee z)$ and the DNF formula $D = (x \wedge \neg y) \vee (\neg x \wedge z)$. For a CNF/DNF formula F, $v(F)$ denotes the set of variables occurring negated or unnegated in F. By *negating* a DNF formula we obtain a CNF formula, for instance $\overline{D} = (\neg x \vee y) \wedge (x \vee \neg z)$. A (partial truth) *assignment* is a mapping $\tau : X \to \{0, 1\}$ (0 representing false, 1 representing true) defined on a set X of variables. We write $v(\tau) = X$. If $v(\tau) = \{x\}$ then we denote τ simply by '$x = 1$' or '$x = 0$'. An assignment τ extends in the obvious way to literals over $v(\tau)$ via $\tau(\neg x) = 1 - \tau(x)$. We identify each term of a DNF formula as a partial assignment, e.g., the term $(x \wedge \neg y)$ corresponds to $\tau : \{x, y\} \to \{0, 1\}$ with $\tau(x) = 1$ and $\tau(y) = 0$. $F[\tau]$ denotes the *restriction* of a CNF formula F to τ (i.e., $F[\tau]$ is obtained from F by removing all clauses that contain a literal that is true under τ, and by removing from the remaining clauses all literals that are false under τ). A CNF formula F is *satisfiable* if $F[\tau] = \emptyset$ for some assignment τ, otherwise it is *unsatisfiable*. A DNF formula is a *tautology* if its negation is unsatisfiable. We also consider *variable deletion* in the following form: If X is a set of variables and F a CNF formula, then $F - X$ denotes the CNF formula obtained from F by removing from all clauses literals of the form x or $\neg x$ for $x \in X$.

Base Classes A *base class* is a class of CNF formulas for which both membership and satisfiability can be decided in polynomial time. Throughout this paper we also assume that *self-reducibility* holds for the considered base classes $\mathcal{C}$: For every $F \in \mathcal{C}$ and $x \in v(F)$ also $F[x = 0], F[x = 1] \in \mathcal{C}$.

In this paper, we consider all base classes that can be obtained as the union of the following fundamental classes of CNF formulas:

- 2CNF, i.e., the family of all CNF formulas having at most two literals per clause,
- HORN, i.e., the family of all CNF formulas having at most one positive literal per clause,
- HORN$_{-1}$, i.e., the family of all CNF formulas having at most one negative literal per clause.

Let $\mathcal{F} = \{\text{2CNF}, \text{HORN}, \text{HORN}_{-1}\}$. The three considered classes are the most important of the six classes considered by Schaefer [1978]: The remaining three classes either don't directly apply to CNF formulas (affine formulas), or are not self-reducible (0-valid and 1-valid formulas).

We consider any *heterogeneous base class* $\mathcal{C}$ such that $\mathcal{C} = \bigcup_{F \in \mathcal{F}} F$, as has been first considered by Gaspers *et al.* [2017a]. Finally, we consider the class of *renamable Horn formulas* (RHORN), which are formulas that can be made Horn by replacing, for a subset X of variables, all occurrences of a literal whose underlying variable belongs to X by its complement [Lewis, 1978; Gaspers and Szeider, 2012]. A base class $\mathcal{C}$ can also be extended by adding *empty clause detection* [Dilkina *et al.*, 2007; Szeider, 2008]. This gives rise to the base class $\mathcal{C}^{\{\}} = \{ F : F \in \mathcal{C} \text{ or } F \text{ contains the empty clause } \}$.

Backdoor Sets Let $\mathcal{C}$ be a base class, F a CNF formula, and $B \subseteq v(F)$. Then B is a *(strong) $\mathcal{C}$-backdoor set (BS) of* F if $F[\tau] \in \mathcal{C}$ for every truth assignment $\tau : B \to \{0, 1\}$;

our BSs are usually referred to as strong BSs in the literature. For each base class $\mathcal{C}$ we consider the following problem:

$\mathcal{C}$-BACKDOOR SET ($\mathcal{C}$-BS). *Instance:* A CNF formula F and a non-negative integer k. *Parameter:* The integer k. *Question:* Has F a $\mathcal{C}$-backdoor set of cardinality at most k?

Let B be a $\mathcal{C}$-BS of a CNF formula F. B is *smallest* if F has no $\mathcal{C}$-BS that is smaller than B; B is *minimal* if F has no $\mathcal{C}$-BS that is a proper subset of B. We say that a set W of variables of F is a *$\mathcal{C}$-backdoor branching set* for a set $B' \subseteq v(F)$, if every $\mathcal{C}$-BS for F that contains B' also contains at least one variable from W. The following proposition lies at the heart of the fpt-algorithms for $\mathcal{C}$-BS (which is also known to be NP-hard for every $\mathcal{C} \in \bigcup_{F \in \mathcal{F}} F$ [Crama *et al.*, 1997]), given by Gaspers *et al.* [2017a] and constitutes a crucial prerequisite for our algorithms for BTs and BDNFs.

Proposition 1 ([Gaspers *et al.*, 2017a])**.** *Let F be a CNF formula and $B \subseteq v(F)$. Then, there is an algorithm that in time $\mathcal{O}(2^{|B|}|F|)$ computes a $\mathcal{C}$-backdoor branching set W for B such that $|W| \leq 5$.*

Note, however, that $\mathcal{C}$-BS for $\mathcal{C} \in \{$RHORN, 2CNF$^{\{\}}$, HORN$^{\{\}}$, HORN$_{-1}{}^{\{\}}\}$ is known to be W[1]-hard [Gaspers and Szeider, 2012].

Backdoor Trees A *binary decision tree (DT)* is a rooted binary tree T. Every inner node of T is assigned a variable, denoted by $v(t)$, and has exactly one left and one right child, which correspond to setting the variable to 0 or 1, respectively. Moreover, every variable occurs at most once on any root-to-leaf path of T. We denote by $v(T)$ the set of all variables assigned to any node of T. Finally, we associate with each node t of T, the truth assignment τ_t that is defined on all the variables $v(P)$ occurring on the unique path P from the root of T to t such that $\tau_t(v) = 0$ ($\tau_t(v) = 1$) if $v \in v(P) \setminus \{v(t)\}$ and P contains the left child (right child) of the node t' on P with $v(t') = v$.

Let $\mathcal{C}$ be a base class, F a CNF formula, and T a DT with $v(T) \subseteq v(F)$. Then T is a *$\mathcal{C}$-backdoor tree (BT) of F* if $F[\tau_v] \in \mathcal{C}$ for every leaf v of T. A $\mathcal{C}$-BT T of F with the smallest number of leaves (in the following, let $|T|$ denote the number of leaves), is a *smallest* $\mathcal{C}$-BT of F. We consider the following parameterized problem:

$\mathcal{C}$-BACKDOOR TREE ($\mathcal{C}$-BT) **Parameter:** k
Input: A CNF formula F and a non-negative integer k.
Question: Does F have a $\mathcal{C}$-BT with at most k leaves?

We will need the following auxiliary proposition showing that computing a smallest $\mathcal{C}$-BT can be done efficiently if the set of allowed variables is small.

Proposition 2 ($\star$)**.** *Let G be a $\mathcal{C}$-BS for a CNF formula F. Then, a smallest $\mathcal{C}$-BT for F using only variables in G can be computed in time $|G|^{2|G|+1}|F|^{\mathcal{O}(1)}$.*

3 Backdoor DNFs

For a truth assignment $\tau : X \to \{0, 1\}$ we denote by D_τ the term that is satisfied by τ, i.e.,

$$D_\tau = \{\, x : x \in X, \tau(x) = 1 \,\} \cup \{\, \neg x : x \in X, \tau(x) = 0 \,\}.$$

Let F be a CNF formula and G a set of partial truth assignments defined on subsets of $v(F)$. We call G a *$\mathcal{C}$-backdoor DNF (BDNF) for F* if (i) for each $\tau \in G$, $F[\tau] \in \mathcal{C}$, and (ii) $G_{\text{DNF}} = \{\, D_\tau : \tau \in G \,\}$ is a tautology. We say that G is a *smallest* $\mathcal{C}$-BDNF for F if $|G|$ is minimal over all $\mathcal{C}$-BDNFs for F. Moreover, we say that G is *term-minimal* if $F[\tau'] \notin \mathcal{C}$ for every proper sub-assignment τ' of an assignment $\tau \in G$. We denote by $v(G)$ the set of all variables used by G, i.e., $v(G) = \bigcup_{\tau \in G} v(\tau)$. We consider the following parameterized problem:

$\mathcal{C}$-BACKDOOR DNF ($\mathcal{C}$-BDNF) **Parameter:** k
Input: A CNF formula F and a non-negative integer k.
Question: Does F have a $\mathcal{C}$-BDNF of size at most k?

If $\mathcal{C}$ is a tractable class and one is given a $\mathcal{C}$-BDNF G for a CNF formula F, then one can decide whether F is satisfiable (and if so compute a satisfying assignment for F) in time $|G|(|F|)^{\mathcal{O}(1)}$ by testing satisfiability of the reduced formula $F[\tau]$ (in time $|F|^{\mathcal{O}(1)}$) for every assignment $\tau \in G$.

Because the set $\{\, \tau_l : l \in L \,\}$ is a $\mathcal{C}$-BDNF for F for every $\mathcal{C}$-BT for F with leaves L, it holds that BTs are a restricted version of BTs (similar to how backdoor sets are a restricted version of BTs). However, BDNFs can be arbitrarily smaller than BTs (which in turn can be arbitrary smaller than BS as shown in [Samer and Szeider, 2008]), which makes them better suited as shortcuts to tractability for Boolean Satisfiability, as shown by the following theorem.

Theorem 1. *For every $s \geq 1$, there is a CNF formula F^s such that a smallest* HORN*-BDNF for F^s is at least $s - 2$ smaller than a smallest* HORN*-BT for F^s.*

We will need the following observations for our algorithms, showing that the variables of a BDNF (or BT) always form a BS together with a simple bound on the number of variables used by a BDNF (or BT).

Observation 1 ($\star$)**.** *Let G be a $\mathcal{C}$-BDNF of a CNF formula F. Then, $v(G)$ is a $\mathcal{C}$-BS. Similarly, if T is a $\mathcal{C}$-BT for F, then $v(T)$ is a $\mathcal{C}$-BS.*

Observation 2 ($\star$)**.** *Let G (T) be a $\mathcal{C}$-BDNF (BT) of a CNF formula F. Then $|var(G)| \leq |G| - 1$ ($|var(T)| \leq |T| - 1$).*

Analogously to Proposition 2 for BTs, we will now show that computing a smallest $\mathcal{C}$-BDNF can be done efficiently if the set of allowed variables is small.

Proposition 3 ($\star$)**.** *Let B be a $\mathcal{C}$-BS for a CNF formula F. Then, a smallest $\mathcal{C}$-BDNF for F containing only variables in B can be computed in time $\mathcal{O}(2^{3^{|B|+1}} + 3^{|B|}|F|^{\mathcal{O}(1)})$.*

4 Finding BDNFs and BTs

In this section, we will provide a complete classification of the parameterized complexity of $\mathcal{C}$-BT and $\mathcal{C}$-BDNF for every base class $\mathcal{C}$ such that $\mathcal{C} = \bigcup_{F \in \mathcal{F}} F$. In particular, we will show that both problems are fixed-parameter tractable if and only if $\mathcal{C} \neq$ HORN $\cup$ HORN$_{-1}$ (assuming that FPT $\neq$ W[1]). We start by giving our fpt-algorithms and then show that both problems are W[1]-hard for the case that $\mathcal{C} =$ HORN $\cup$ HORN$_{-1}$.

Let $\mathcal{F}_+$ be the set of all these base classes, i.e., $\mathcal{F}_+ = \{2\text{CNF}, \text{HORN}, \text{HORN}_{-1}, 2\text{CNF} \cup \text{HORN}, 2\text{CNF} \cup \text{HORN}_{-1}\}$. Note first that using Propositions 2 and 3, both problems are easily seen to be in XP for any base class $\mathcal{C}$. This is because there are at most $|v(F)|^k$ sets of variables that can be used by a BDNF (or BT) of size at most k and for each of those sets, we can compute a smallest BDNF (or BT) that uses only those variables in fpt-time. This also illustrates that the main challenge that we have to overcome is to design a fpt-procedure to enumerate all sets of variables that can potentially be used by a smallest BDNF (or BT). Given Observation 1, one might think that any smallest BDNF (or BT) uses only the variables of a smallest BS, which if it were true would already provide us with such an fpt-procedure since Proposition 1 can be easily employed to enumerate all minimal BSs of size at most k in fpt-time. Unfortunately, this is not the case as shown by the following theorem.

Theorem 2 ($\star$)**.** *For every $\mathcal{C} \in \mathcal{F}_+$ and every $s \geq 1$, there is a CNF formula $F_s^{\mathcal{C}}$ such that a smallest $\mathcal{C}$-BDNF ($\mathcal{C}$-BT) for $F_s^{\mathcal{C}}$ is at least $2^s - 2(s+1)$ larger than a smallest $\mathcal{C}$-BDNF ($\mathcal{C}$-BT), whose variables form a minimal $\mathcal{C}$-BS for $F_s^{\mathcal{C}}$.*

Proof Sketch. We show the theorem for $\mathcal{C} = \text{HORN}$ and $\mathcal{C}$-BDNFs. F_s^{HORN} has variables $\{p, a_1, \ldots, a_s\} \cup \{\, q_j : 1 \leq j \leq r \,\}$, where $r = 2^s - s$ and the following clauses:

- a clause $\{a_i, p\}$ for every $1 \leq i \leq s$ and
- the clauses $\{a_1, \ldots, a_s, q_j, \neg p\}$ for every $1 \leq j \leq r$.

We first show that F_s^{HORN} has only two types of minimal HORN-BSs, namely, the set $B = \{a_1, \ldots, a_s\}$ and the sets $B_i = B \setminus \{a_i\} \cup \{p, q_1, \ldots, q_r\}$ for every i with $1 \leq i \leq s$. This is because:

- no proper subset of B is a HORN-BS for F_s^{HORN} because of the clauses $\{a_i, p\}$,
- any HORN-BS can miss at most one variable of B (because of the clause $\{a_1, \ldots, a_s, q_1, \neg p\}$), and
- any HORN-backdoor that misses one variable in B has to contain p (because of the clauses $\{a_i, p\}$) and also every q_j (because of the clauses $\{a_1, \ldots, a_s, q_j, \neg q\}$).

Therefore, every minimal HORN-BS that is not B has size at least $s - 1 + 2^s - s + 1 = 2^s$, which together with Observation 1 implies that any HORN-BDNF that uses only variables in B_i for some i has size at least 2^s.

We now show that the same applies also to every HORN-BDNF that uses only the variables in B, i.e., that it has size at least 2^s. This is because $F[\alpha] \notin \text{HORN}$ for every partial assignment $\alpha : B' \to \{0,1\}$, where $B' \subsetneq B$ (because of the clause $\{a_i, p\}$, where $a_i \in B \setminus B'$). Therefore, every term of a HORN-BDNF has to assign all variables in B, which implies that its size is at least 2^s.

It only remains to show that F_s^{HORN} has a HORN-BDNF of size at most $s + 2$. To see this consider the following HORN-BDNF for F_s^{HORN} of size $s + 2$, which contains the following assignments: (1) the assignment $(p = 0)$, (2) the assignment $(p = 1, a_1 = 0, \ldots, a_s = 0)$, and (3) for every i with $1 \leq i \leq s$ the assignment $(p = 1, a_i = 1)$. Therefore, a smallest $\mathcal{C}$-BDNF for F_s^{HORN} is at least $2^s - (s+2) \geq 2^s - 2(s+1)$ larger than such a smallest BDNF that only uses variables in a minimal $\mathcal{C}$-BS for F_s^{HORN}. □

The theorem also shows that our BTs can be arbitrarily smaller than the BTs detected by Samer and Szeider's algorithm [Samer and Szeider, 2008], which are only allowed to use subset-minimal $\mathcal{C}$-backdoor sets.

It is therefore not sufficient to enumerate all BSs of a CNF formula F to identify a set of variables that is used by a smallest BDNF (or BT). Nevertheless, Observation 1 still allow us to assume that we are given a BS for F and as we will show next this will be sufficient to identify all sets of variables that can lead to a smallest BDNF (or BT). In particular, we will show next that if a smallest a smallest BDNF (or BT) uses additional variables outside of a BS, then the set of those additional variables has a special property (which we will later exploit to extend minimal BSs), which we call useful. Let F be a CNF-formula and B a $\mathcal{C}$-BS. We say that a set U of variables is *$\mathcal{C}$-useful* for B if for every assignment $\beta : U \to \{0,1\}$, there is a partial assignment $\alpha : B' \to \{0,1\}$ for some $B' \subseteq B$ such that $F[\alpha] \notin \mathcal{C}$ but $F[\alpha \cup \beta] \in \mathcal{C}$. The following lemma shows that the set of variables used by a BDNF (or BT) for F that go beyond a BS, needs to be useful.

Lemma 1 ($\star$)**.** *Let G be a smallest term-minimal $\mathcal{C}$-BDNF for F and let B be a $\mathcal{C}$-BS contained in $v(G)$, then the set $U = v(G) \setminus B$ is $\mathcal{C}$-useful. Similarly, if T is a smallest $\mathcal{C}$-BT for F and B is a $\mathcal{C}$-BS contained in $v(T)$, then the set $U = v(T) \setminus B$ is $\mathcal{C}$-useful.*

Proof Sketch. We will show the lemma for BDNFs. If $U = \emptyset$, then there is nothing to show. Hence, assume that $U \neq \emptyset$ and suppose for a contradiction that the statement of the lemma does not hold. Then, there is an assignment $\beta : U \to \{0,1\}$ such that $F[\alpha \cup \beta] \notin \mathcal{C}$ for every assignment $\alpha : B' \to \{0,1\}$ with $B' \subseteq B$ and $F[\alpha] \notin \mathcal{C}$. Let $G[\beta]$ be the set of all assignments in G that are compatible with β, which is non-empty because G_{DNF} is a tautology. If there is no assignment in $G[\beta]$ that assigns at least one variable in U, then $G[\beta]_{\text{DNF}}$ is again a tautology and therefore $G[\beta]$ is a $\mathcal{C}$-BDNF for F, which because $U \neq \emptyset$ is smaller than G contradicting our assumption that G was minimal. Therefore, $G[\beta]$ contains an assignment τ that is defined on at least one variable of U. Let τ' be the restriction of τ to variables in B. Then, $F[\tau'] \in \mathcal{C}$ and therefore $G \setminus \{\tau\} \cup \{\tau'\}$ is a $\mathcal{C}$-BDNF for F, contradicting our assumption that G is term-minimal. □

We will show next how we can efficiently find $\mathcal{C}$-useful sets for a given $\mathcal{C}$-BS B of a CNF formula F. We say that a set A of variables of F is a *$\mathcal{C}$-branching set* for B if $A \cap U \neq \emptyset$ for every $\mathcal{C}$-useful set U for B. As we will see later, all we need to find $\mathcal{C}$-useful sets is to be able to compute "small" $\mathcal{C}$-branching sets efficiently (i.e., fpt parameterized by $|B|$). The following lemma show show to achieve exactly this for all base classes in $\mathcal{F}_+$.

Lemma 2. *Let $\mathcal{C} \in \mathcal{F}_+$ and let B be a $\mathcal{C}$-BS for a CNF formula F. Then, a $\mathcal{C}$-branching set A such that $|A| \leq 5 \cdot 3^{|B|}$ can be computed in time $\mathcal{O}(3^{|B|}|F|)$.*

Proof Sketch. We show the statement of the lemma for the (simple) case that $\mathcal{C} = \text{HORN}$. Let $\alpha : B' \to \{0,1\}$ with $B' \subseteq B$ be a partial assignment of B such that $F[\alpha] \notin$

HORN. We denote by $P(\alpha)$ the set of all variables that occur positively in a clause in $F[\alpha] \setminus$ HORN but are not in B. We claim that every $\mathcal{C}$-useful set U for B has to contain all variables in $P(\alpha)$ for some assignment α as above. This then shows the statement of the lemma because we can obtain a branching set A of size at most $3^{|B|}$ by choosing an arbitrary variable from $P(\alpha)$ for every $\alpha : B' \to \{0,1\}$ with $B' \subseteq B$ and $F[\alpha] \notin$ HORN.

Suppose for a contradiction that this is not the case and let U be a $\mathcal{C}$-useful set for B such that $P(\alpha) \not\subseteq U$ for every assignment $\alpha : B' \to \{0,1\}$ with $F[\alpha] \notin$ HORN. Let $\beta : U \to \{1\}$ the assignment setting all variables in U to 1. Because U is $\mathcal{C}$-useful for B, there is a partial assignment $\alpha : B' \to \{0,1\}$ for B such that $F[\alpha] \notin$ HORN but $F[\alpha \cup \beta] \in$ HORN. Because $P(\alpha) \not\subseteq U$, there is a variable $p \in P(\alpha) \setminus U$ and a clause $C \in F[\alpha] \setminus$ HORN such that all positive literals in C are from $B \cup \{p\}$; this is because B is also a deletion HORN-BS for F and therefore every clause in $F - B$ contains at most one positive literal. Hence, β only assigns negative literals of C to 1 and it follows that $C[\alpha \cup \beta] \notin$ HORN, contradicting our assumption that $F[\alpha \cup \beta] \in$ HORN. □

Algorithm 1 Main method for finding a smallest BDNF.

Input: CNF formula F, subset $B \subseteq v(F)$, and integer k
Output: a smallest $\mathcal{C}$-BDNF for F using at least the variables in B having size at most k if it exists, otherwise `nil`

```
1:  function MINBDNF(F, k, B)
2:      G_min ← "compute a smallest C-BDNF for F using only
                 variables in B using Proposition 3"
3:      if |B| ≥ k − 1 then
4:          if G_min = nil or |G_min| ≤ k then
5:              return G_min
6:          return nil
7:      if B is not a C-BS for F then
8:          A ← "compute a C-backdoor branching set for B
                 using Proposition 1"
9:      else
10:         A ← "compute a C-branching set for B using Lemma 2
11:     for v ∈ A do
12:         G ←MINBDNF(F, k, B ∪ {v})
13:         if G ≠ nil and |G| < |G_min| then
14:             G_min ← G
15:     if |G_min| ≤ k then return G_min
16:     return nil
```

We are now now ready to show our main tractability result.

Theorem 3. *Let $\mathcal{C} \in \mathcal{F}_+$. Then, the problems* $\mathcal{C}$-BDNF *and* $\mathcal{C}$-BT *are fixed-parameter tractable.*

Proof. We present the algorithm for $\mathcal{C}$-BDNF, which is illustrated in Algorithm 1. Given a CNF formula F, a subset $B \subseteq v(F)$, and an integer k, the main function **minBDNF** behind the algorithm computes a smallest $\mathcal{C}$-BDNF for F that uses at least the variables in B and has size at most k; if no such $\mathcal{C}$-BDNF exists, the algorithm returns `nil`. To solve $\mathcal{C}$-BDNF, the function **minBDNF** needs to be called with B being the emptyset. Towards showing the correctness of the algorithm consider the case that F has a $\mathcal{C}$-BDNF of size at most k and let G be a smallest such $\mathcal{C}$-BDNF. Because of Observation 2, $|v(G)| \leq k-1$. Moreover, because of Observation 1, $v(G)$ contains a minimal $\mathcal{C}$-BS say S of size at most $k-1$. We first show that the algorithm is called for $B = S$. This is because as long as the set B is not a strong $\mathcal{C}$-BS, the algorithm branches on the variables inside a $\mathcal{C}$ backdoor branching set A, which by definition must also contains a variable from $S \setminus B$. If $v(G) = S$, then the call of **minBDNF** for $B = S$ already finds a $\mathcal{C}$-BDNF of size $|G|$ in Line 2, which will eventually be returned. Otherwise, we obtain from Lemma 1 that $v(G) \setminus S$ is $\mathcal{C}$-useful for S, and it remains to show that the algorithm is eventually called for $B = v(G)$. To see this consider the calls following the call where $B = S$. Since B is already a $\mathcal{C}$-BS, the algorithm now branches on all variables of a $\mathcal{C}$-branching set A for B, which by definition must also contain a variable of $v(G) \setminus B$. Finally, it is easy to see that any solution returned by the algorithm is a $\mathcal{C}$-BDNF of size at most k.

It remains to analyse the runtime of the algorithm. Since every execution of **minBDNF** leads to at most $|A|$ recursive calls, each recursive call adds at least one variable to B and the algorithm stops whenever $|B| \geq k-1$, we obtain that the algorithm makes at most $|A|^{k-1}$ recursive class. Moreover, the time required for one call of **minBDNF** is easily seen to be dominated by the time required by Line 2 to compute a smallest $\mathcal{C}$-BDNF for F using only variables in B using Proposition 3, which is at most $\mathcal{O}(2^{3^{|B|+1}} + 3^{|B|}|F|^{\mathcal{O}(1)})$. Therefore, the total runtime of the algorithm is at most $\mathcal{O}(|A|^{k-1}(2^{3^{|B|+1}} + 3^{|B|}|F|^{\mathcal{O}(1)})$, which because $|A|$ is bounded by a function of k (for all classes $\mathcal{C} \in \mathcal{F}_+$ due to Lemma 2) shows that $\mathcal{C}$-BDNF is in FPT. □

The following theorem now shows that the problems are W[2]-hard for the only remaining case that $\mathcal{C} =$ HORN $\cup$ HORN$_{-1}$. The proof is based on a reduction by Gaspers *et al.* [2017a].

Theorem 4 ($\star$)**.** *Let $\mathcal{C} =$ HORN $\cup$ HORN$_{-1}$. Then, the problems* $\mathcal{C}$-BT *and* $\mathcal{C}$-BDNF *are* W[2]*-hard.*

5 Experiments

We complement our theoretical results by experiments. We compute BDNFs and BTs on a large number of CNF formulas, stemming from various applications like logistics, planning, and combinatorics. The instances form ten groups: (i) all interval series (*ais*)[1], (ii–iii) graph coloring (*flat, pret*)[1], (iv) logistics car configuration (*daimler*) [Sinz *et al.*, 2003], (v) parity function learning (*parity*)[1], (vi) inductive inference (*inductive*)[1], (vii) planning (*blocksworld*)[1], (viii) pigeon hole (*pigeon*)[1], and (ix–x) vertex cover and treewidth for named graphs (*vc and tw*). Since our algorithms are based on SAT encodings, we can avoid the restriction to base classes that allow for fixed-parameter tractability. In particular, this allows us to use the base classes HORN$^{\{\}}$ and RHORN$^{\{\}}$, for which already the BS problem is known to be W[1]-hard.

[1]https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html

	HORN$^{\{\}}$				RHORN$^{\{\}}$			
Group	Size	Total	|BDNF|/|BT|	σ^2	Size	Total	|BDNF|/|BT|	σ^2
ais	87/1051	2/2	$8.5 \cdot 10^{-3}$	$1.3 \cdot 10^{-4}$	61/581	1/1	$1.7 \cdot 10^{-2}$	$0.0 \cdot 10^{0}$
blocksworld	82/607	2/2	$2.6 \cdot 10^{-1}$	$3.5 \cdot 10^{-2}$	82/607	2/2	$2.4 \cdot 10^{-1}$	$2.7 \cdot 10^{-2}$
daimler	1407/1887	3/3	$3.2 \cdot 10^{-1}$	$4.3 \cdot 10^{-3}$	1667/3977	18/18	$4.1 \cdot 10^{-1}$	$2.2 \cdot 10^{-1}$
flat	150/545	99/99	$1.5 \cdot 10^{-3}$	$5.6 \cdot 10^{-6}$	150/545	97/99	$6.4 \cdot 10^{-4}$	$9.4 \cdot 10^{-8}$
inductive	288/5077	16/16	$5.4 \cdot 10^{-1}$	$1.0 \cdot 10^{-1}$	655/9649	41/41	$1.1 \cdot 10^{0}$	$5.6 \cdot 10^{-1}$
parity	201/803	10/10	$9.5 \cdot 10^{-1}$	$3.6 \cdot 10^{-1}$	70/277	5/5	$1.1 \cdot 10^{0}$	$5.0 \cdot 10^{-2}$
pigeon	74/322	5/5	$3.0 \cdot 10^{-3}$	$2.7 \cdot 10^{-5}$	49/169	2/2	$1.2 \cdot 10^{-2}$	$1.4 \cdot 10^{-4}$
pret	105/280	8/8	$3.6 \cdot 10^{-5}$	$1.3 \cdot 10^{-9}$	160	4/4	$3.2 \cdot 10^{-5}$	$4.5 \cdot 10^{-12}$
tw	222/965	9/12	$5.6 \cdot 10^{-1}$	$2.7 \cdot 10^{-1}$	125/433	5/6	$6.1 \cdot 10^{-1}$	$4.4 \cdot 10^{-2}$
vc	175/355	38/38	$5.3 \cdot 10^{-1}$	$1.5 \cdot 10^{-1}$	175/355	38/38	$5.5 \cdot 10^{-1}$	$1.5 \cdot 10^{-1}$

Table 1: Comparison between backdoor DNFs and backdoor trees for several classes and groups of instances. |BDNF|/|BT| is the average ratio between the number of terms of the computed BDNF and the number of leaves of the computed BT, σ^2 is the variance. *Size* shows the average number of variables/clauses; *Total* shows the number of instances for which a BDNF could be computed.

We compute the SAT encodings using Python 3.8.0 and PySAT 1.6.0[2]. As the SAT solver, we use Cadical as provided by PySAT, which works slightly better with our encodings than the other solvers provided by PySAT. We run the experiments on servers with two Intel Xeon E5540 CPUs, each running at 2.53 GHz per core, use Ubuntu 18.04. Each run is limited to six hours and 12 GB RAM.

The algorithm for **BDNFs** is based on incremental SAT solving. It finds one potential term of a BDNF in each solver call. Once a term is found, it is added to the encoding and so excluded in future calls. We use a cardinality constraint on the size of the term to obtain only subset-minimal terms. When all the found terms together form a tautological DNF, the algorithm terminates. Termination is checked using a second incremental SAT solver instance, which checks, in increments of 1000 added terms, whether the DNF's negation is an unsatisfiable CNF. Finally, we minimize the DNF by computing a minimal unsatisfiable core [Belov *et al.*, 2014] for its negation. The found DNF is then inclusion-minimal but not necessarily of smallest cardinality. We compute **BTs** using a recursive algorithm. The algorithm computes one branch of the tree at a time using a SAT solver call. The algorithm then calls itself for each sub-branch.

Results In total, we select 2197 instances from the sources mentioned above that were small enough for the encodings. For each instance, we compute a deletion BS and discard instances based on the BS's size: we choose 192 instances where a HORN-backdoor is smaller than 100 and 222 instances where a RHORN-backdoor is smaller than 50.

Given our theoretical results, we expect BDNFs to be smaller than BTs. Indeed, in Table 1 we see this comparison in terms of the ratio of the BDNF size to BT size. The lower the ratio, the smaller the BDNF in comparison to the respective BT.

We found the lowest ratios for the graph coloring instances in *pret* and *flat*. For RHORN the DNFs for the groups *inductive* and *parity* are comparatively large. *Parity* is a group where it is easy to obtain empty clauses. Therefore, the DNFs (4 partial assignments) and trees (2 partial assignments) are very small compared to the BS size (21–26). *Inductive* are instances that are almost in RHORN and have a deletion BS of size 1. The respective DNFs and trees are also very small. For the vertex cover and treewidth encodings, the DNFs are about half as large as the trees for all classes.

Interestingly, the set of variables used by about 90 % of the BDNFs are not equal (but only contain) a minimal BS. This is also strongly supported by our theoretical analysis showing that BTs and BDNFs can be arbitrarily smaller if they are not restricted to use only variables from a minimal BS (Theorem 2).

6 Conclusion

We have introduced backdoor DNFs as a versatile tool for representing the hidden structure in a SAT instance. Our main theoretical results show that for fundamental base classes for which the detection of strong backdoor sets is FPT, also the detection of backdoor DNFs is FPT. This finding is significant, as backdoor DNFs can be far more succinct than backdoor sets or backdoor trees. Our experiments show that SAT instances drawn from a wide range of application domains indeed contain backdoor DNFs that are by several orders of magnitude smaller than their backdoor tree counterparts.

In the past, parameterized complexity of backdoor set detection, and the use of backdoor sets for tractable problem solving, has been explored in a wide range of problems beyond SAT: CSP [Gaspers *et al.*, 2017c; Ganian *et al.*, 2017; Gaspers *et al.*, 2017b], ASP [Fichte and Szeider, 2015a; Fichte and Szeider, 2015b], Temporal Logic [Meier *et al.*, 2019], QBF [Samer and Szeider, 2009] Abstract Argumentation [Dvorák *et al.*, 2012], and Planning [Kronegger *et al.*, 2019]. We think that many of these results can be lifted to backdoor DNFs. This provides several challenging research question for future work.

[2]https://pysathq.github.io

References

[Ansótegui *et al.*, 2014] Carlos Ansótegui, Maria Luisa Bonet, Jesús Giráldez-Cru, and Jordi Levy. The fractal dimension of SAT formulas. In *Proc. IJCAR '14*, *LNCS 8562*, pp. 107–121. Springer, 2014.

[Belov *et al.*, 2014] Anton Belov, Marijn Heule, and João Marques-Silva. MUS extraction using clausal proofs. In *Proc. SAT '14*, *LNCS 8561*, pp. 48–57. Springer, 2014.

[Cook, 1971] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proc. STOC '71*, pp. 151–158, Shaker Heights, Ohio, 1971.

[Crama *et al.*, 1997] Y. Crama, O. Ekin, and P. L. Hammer. Variable and term removal from Boolean formulae. *Discr. Appl. Math.*, 75(3):217–230, 1997.

[Cygan *et al.*, 2015] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[Dilkina *et al.*, 2007] Bistra N. Dilkina, Carla P. Gomes, and Ashish Sabharwal. Tradeoffs in the complexity of backdoor detection. In *Proc. CP '07*, *LNCS 4741*, pp. 256–270. Springer, 2007.

[Dvorák *et al.*, 2012] Wolfgang Dvorák, Sebastian Ordyniak, and Stefan Szeider. Augmenting tractable fragments of abstract argumentation. *Artif. Intell.*, 186:157–173, 2012.

[Fichte and Szeider, 2015a] Johannes Klaus Fichte and Stefan Szeider. Backdoors to normality for disjunctive logic programs. *ACM Trans. Comput. Log.*, 17(1), 2015.

[Fichte and Szeider, 2015b] Johannes Klaus Fichte and Stefan Szeider. Backdoors to tractable answer set programming. *Artif. Intell.*, 220:64–103, March 2015.

[Ganian and Szeider, 2015] Robert Ganian and Stefan Szeider. Community structure inspired algorithms for SAT and #SAT. In *Proc. SAT '15*, *LNCS 9340*, pp. 223–237. Springer, 2015.

[Ganian and Szeider, 2017] Robert Ganian and Stefan Szeider. New width parameters for model counting. In *SAT '17*, *LNCS 10491*, pp. 38–52. Springer, 2017.

[Ganian *et al.*, 2017] Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Discovering archipelagos of tractability for constraint satisfaction and counting. *ACM Trans. on Alg.*, 13(2):29:1–29:32, 2017.

[Gaspers and Szeider, 2012] Serge Gaspers and Stefan Szeider. Backdoors to satisfaction. In *The Multivariate Algorithmic Revolution and Beyond*, *LNCS 7370*, pp. 287–317. Springer, 2012.

[Gaspers *et al.*, 2017a] Serge Gaspers, Neeldhara Misra, Sebastian Ordyniak, Stefan Szeider, and Stanislav Zivný. Backdoors into heterogeneous classes of SAT and CSP. *J. Comput. Syst. Sci.*, 85:38–56, 2017.

[Gaspers *et al.*, 2017b] Serge Gaspers, Neeldhara Misra, Sebastian Ordyniak, Stefan Szeider, and Stanislav Zivny. Backdoors into heterogeneous classes of SAT and CSP. *J. of Comput. and Syst. Sci.*, 85:38–56, 2017.

[Gaspers *et al.*, 2017c] Serge Gaspers, Sebastian Ordyniak, and Stefan Szeider. Backdoor sets for CSP. In *The Constraint Satisfaction Problem*, *Dagstuhl Follow-Ups 7*, pp. 137–157. Dagstuhl, 2017.

[Impagliazzo *et al.*, 2001] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. of Comput. and Syst. Sci.*, 63(4):512–530, 2001.

[Jamali and Mitchell, 2017] Sima Jamali and David Mitchell. Improving SAT solver performance with structure-based preferential bumping. In *Proc. GCAI '17*, *EPiC 50*, pp. 175–187. EasyChair, 2017.

[Kronegger *et al.*, 2019] Martin Kronegger, Sebastian Ordyniak, and Andreas Pfandler. Backdoors to planning. *Artif. Intell.*, 269:49–75, 2019.

[Lewis, 1978] Harry R. Lewis. Renaming a set of clauses as a Horn set. *J. of the ACM*, 25(1):134–135, January 1978.

[Mateescu, 2011] Robert Mateescu. Treewidth in industrial SAT benchmarks. MSR-TR-2011-22, Microsoft, 2011.

[Meier *et al.*, 2019] Arne Meier, Sebastian Ordyniak, M. S. Ramanujan, and Irena Schindler. Backdoors for linear temporal logic. *Algorithmica*, 81(2):476–496, 2019.

[Newsham *et al.*, 2014] Zack Newsham, Vijay Ganesh, Sebastian Fischmeister, Gilles Audemard, and Laurent Simon. Impact of community structure on SAT solver performance. In *Proc. SAT '14*, *LNCS 8561*, pp. 252–268. Springer, 2014.

[Nishimura *et al.*, 2004] Naomi Nishimura, Prabhakar Ragde, and Stefan Szeider. Detecting backdoor sets with respect to Horn and binary clauses. In *Proc. SAT '04*, pp. 96–103, 2004.

[Samer and Szeider, 2008] Marko Samer and Stefan Szeider. Backdoor trees. In *Proc. AAAI '08*, pp. 363–368. AAAI Press, 2008.

[Samer and Szeider, 2009] Marko Samer and Stefan Szeider. Backdoor sets of quantified Boolean formulas. *J. Autom. Reason.*, 42(1):77–97, 2009.

[Schaefer, 1978] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proc. STOC '78*, pp. 216–226. ACM, 1978.

[Sinz *et al.*, 2003] Carsten Sinz, Andreas Kaiser, and Wolfgang Küchlin. Formal methods for the validation of automotive product configuration data. *Artif. Intell. Eng. Des. Anal. Manuf.*, 17(1):75–97, 2003.

[Szeider, 2008] Stefan Szeider. Matched formulas and backdoor sets. *J. on Satisf. Boolean Model. Computat.*, 6:1–12, 2008.

[Vardi, 2014] Moshe Y. Vardi. Boolean satisfiability: theory and engineering. *Comm. ACM*, 57(3):5, March 2014.

[Williams *et al.*, 2003] Ryan Williams, Carla Gomes, and Bart Selman. Backdoors to typical case complexity. In *Proc. IJCAI '03*, pp. 1173–1178. M. Kaufmann, 2003.