Technical Report AC-TR-21-001
January 2021

Backdoor DNFS

Sebastian Ordyniak, André Schidler, and Stefan Szeider
Backdoor DNFs

Sebastian Ordyniak¹, André Schidler², Stefan Szeider²
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 $C$ of CNF formulas (either defined by a polynomial-time subproblem 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 $C$. 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 $C$) and a strong backdoor (all instantiations for the backdoor variables result in an instance that belongs to $C$). 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 $C$, 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 $C$. 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 $C$. 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 $C$. 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 expectations 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 $C = (x \lor \neg y) \land (\neg x \lor z)$ and the DNF formula $D = (x \land \neg y) \lor (\neg x \land z)$. For a CNF/DNF formula $F$, $v(F)$ denotes the set of variables occurring negated or un-negated in $F$. By negating a DNF formula we obtain a CNF formula, for instance $\overline{D} = (x \lor y) \land (x \lor \neg z)$. A (partial) truth assignment is a mapping $\tau : X \rightarrow \{0, 1\}$ (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 $\tau$ simply by ‘$x = 1’ or ‘$x = 0’.” An assignment $\tau$ 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 $(x \land \neg y)$ corresponds to $\tau : \{x, y\} \rightarrow \{0, 1\}$ with $\tau(x) = 1$ and $\tau(y) = 0$. $F[\tau]$ denotes the restriction of a CNF formula $F$ to $\tau$ (i.e., $F[\tau]$ is obtained from $F$ by removing all clauses that contain a literal that is true under $\tau$, and by removing from the remaining clauses all literals that are false under $\tau$). A CNF formula $F$ is satisfiable if $F[\tau] = \emptyset$ for some assignment $\tau$, 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 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 $C$: For every $F \in C$ and $x \in v(F)$ also $F[x = 0], F[x = 1] \in C$.

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-\text{-}\text{-}1, i.e., the family of all CNF formulas having at most one negative literal per clause.

Let $F = \{\text{2CNF, Horn, Horn-}1\}$. 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 $C$ such that $C = \bigcup_{F \in F} F$, 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 $C$ can also be extended by adding empty clause detection [Dilkina et al., 2007; Szeider, 2008]. This gives rise to the base class $C^\bot = \{F : F \in C \vee F \text{ contains the empty clause}\}$.

Backdoor Sets Let $C$ be a base class, $F$ a CNF formula, and $B \subseteq v(F)$. Then $B$ is a (strong) $C$-backdoor set ($BS$) if $F[\tau] \in C$ for every truth assignment $\tau : B \rightarrow \{0, 1\}$;
our BSs are usually referred to as strong BSs in the literature. For each base class C we consider the following problem:

**C-Backdoor Set (C-BS).** Instance: A CNF formula F and a non-negative integer k. Parameter: The integer k. Question: Has F a C-backdoor set of cardinality at most k?

Let B be a C-BS of a CNF formula F. B is smallest if F has no C-BS that is smaller than B; B is minimal if F has no C-BS that is a proper subset of B. We say that a set W of variables of F is a C-backdoor branching set for a set B′⊆v(F), if every 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 C-BS (which is also known to be NP-hard for every C ∈ ∪F∈X 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 ⊆ v(F). Then, there is an algorithm that in time O(2|B||F|) computes a C-backdoor branching set W for B such that |W| ≤ 5.

Note, however, that C-BS for C ∈ {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 τt that is defined on all the variables v(P) occurring on the unique path P from the root of T to t such that τt(v) = 0 (τt(v) = 1) if v ∈ v(P) \ {v(t)} and P contains the left child (right child) of the node t on P with v(t) = v.

Let C be a base class, F a CNF formula, and T a DT with v(T) ⊆ v(F). Then T is a C-backdoor tree (BT) of F if Fτt] ∈ C for every leaf v of T. A C-BT T of F with the smallest number of leaves (in the following, let |T| denote the number of leaves), is a smallest C-BT of F. We consider the following parameterized problem:

**C-Backdoor Tree (C-BT)**

**Input:** A CNF formula F and a non-negative integer k.

**Question:** Does F have a C-BT with at most k leaves?

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

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

**3 Backdoor DNFs**

For a truth assignment τ : X → {0, 1} we denote by Dτ 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 C-backdoor DNF (BDNF) for F if (i) for each τ ∈ G, F[τ] ∈ C, and (ii) G_{DNF} = \{ D_\tau : \tau \in G \} is a tautology. We say that G is a smallest C-BDNF for F if |G| is minimal over all C-BDNFs for F. Moreover, we say that G is term-minimal if F[τ] ∉ C for every proper sub-assignment τ′ of an assignment τ ∈ G. We denote by v(G) the set of all variables used by G, i.e., v(G) = ∪_{\tau \in G} v(\tau).

We consider the following parameterized problem:

**C-Backdoor DNF (C-BDNF)**

**Input:** A CNF formula F and a non-negative integer k.

**Question:** Does F have a C-BDNF of size at most k?

If C is a tractable class and one is given a 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|^O(1) by testing satisfiability of the reduced formula F[τ] in time |F|^O(1) for every assignment τ ∈ G.

Because the set \{ τ_l : l ∈ L \} is a C-BDNF for F for every 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 ≥ 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** ([*]). Let G be a C-BDNF of a CNF formula F. Then, v(G) is a C-BS. Similarly, if T is a C-BT for F, then v(T) is a C-BS.

**Observation 2** ([*]). Let G (T) be a C-BDNF (BT) of a CNF formula F. Then |v(G)| ≤ |F| − 1 (|v(T)| ≤ |T| − 1).

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

**Proposition 3** ([*]). Let B be a C-BS for a CNF formula F. Then, a smallest C-BDNF for F containing only variables in B can be computed in time O(2^O(n+1) + 3^B||F|^O(1)).

**4 Finding BDNFs and BTs**

In this section, we will provide a complete classification of the parameterized complexity of C-BT and C-BDNF for every base class C such that C = ∪_{F∈X} F. In particular, we will show that both problems are fixed-parameter tractable if and only if C = HORN ∪ HORN−1 (assuming that FPT ≠ W[1]). We start by giving our fpt-algorithms and then show that both problems are W[1]-hard for the case that C = HORN ∪ HORN−1.
Let \( F_+ \) be the set of all these base classes, i.e., \( F_+ = \{2\text{CNF}, \text{HORN}, \text{HORN}^{-1}, 2\text{CNF} \cup \text{HORN}, 2\text{CNF} \cup \text{HORN}^{-1}\} \).

Note first that using Propositions 2 and 3, both problems are easily seen to be in XP for any base class \( C \). 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 in the following theorem.

**Theorem 2 (⋆).** For every \( C \in F_+ \) and every \( s \geq 1 \), there is a CNF formula \( F_+^C \) such that a smallest \( C \)-BDNF (\( C \)-BT) for \( F_+^C \) is at least \( 2^s - 2(s + 1) \) larger than a smallest \( C \)-BDNF (\( C \)-BT), whose variables form a minimal \( C \)-BS for \( F_+^C \).

**Proof Sketch.** We show the theorem for \( C = \text{HORN} \) and \( C \)-BDNFs. \( F_+^{\text{HORN}} \) 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 \( F_+^{\text{HORN}} \) has only two types of minimal \( \text{HORN} \)-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 \( \text{HORN} \)-BS for \( F_+^{\text{HORN}} \) because of the clauses \( \{a_i, p\} \),

- any \( \text{HORN} \)-BS can miss at most one variable of \( B \) (because of the clause \( \{a_1, \ldots, a_s, q_1, \neg p\} \)), and

- any \( \text{HORN} \)-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 \( \{a_1, \ldots, a_s, q_j, \neg p\} \)).

Therefore, every minimal \( \text{HORN} \)-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 \( \text{HORN} \)-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 \( \text{HORN} \)-BDNF that uses only the variables in \( B \), i.e., that it has size at least \( 2^s \). This is because \( F_+^{\text{HORN}} \) is \( \text{HORN} \) for every partial assignment \( \alpha : B' \to \{0, 1\} \), where \( B' \subseteq B \) (because of the clause \( \{a_i, p\} \)), where \( a_i \in B \setminus B' \). Therefore, every \( \text{HORN} \)-BDNF that uses all variables in \( B \), which implies that its size is at least \( 2^s \).

It only remains to show that \( F_+^{\text{HORN}} \) has a \( \text{HORN} \)-BDNF of size at most \( s + 2 \). To see this consider the following \( \text{HORN} \)-BDNF for \( F_+^{\text{HORN}} \) 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 \( \text{BDNF} \) for \( F_+^{\text{HORN}} \) is at least \( 2^s - 2(s + 2) \geq 2^s - 2(s + 1) \) larger than such a smallest BDNF that only uses variables in a minimal \( C \)-BS for \( F_+ \).
HORN. We denote by \( P(\alpha) \) the set of all variables that occur positively in a clause in \( F[\alpha] \) but are not in \( B \). We claim that every \( C \)-useful set \( U \) for \( B \) has to contain all variables in \( P(\alpha) \) for some assignment \( \alpha \) 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 \( F[\alpha] \notin \text{HORN} \).

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

Algorithm 1 Main method for finding a smallest \( B \)-DNF.

**Input:** CNF formula \( F \), subset \( B \subseteq v(F) \), and integer \( k \).

**Output:** a smallest \( C \)-BDNF for \( F \) using at least the variables in \( B \) having size at most \( k \) if it exists, otherwise \( \text{nill} \).

1. function MINBDNF(\( F, k, B \))
2. \( G_{\text{min}} \leftarrow \) “compute a smallest \( C \)-BDNF for \( F \) using only variables in \( B \) by using Proposition 3”
3. if \(|B| \geq k - 1 \) then
4. if \( G_{\text{min}} = \text{nill} \) or \(|G_{\text{min}}| \leq k \) then
5. return \( G_{\text{min}} \)
6. return \( \text{nill} \)
7. if \( B \) is not a \( C \)-BS for \( F \) then
8. \( A \leftarrow \) “compute a \( C \)-backdoor branching set for \( B \) using Proposition 1”
9. else
10. \( A \leftarrow \) “compute a \( C \)-branching set for \( B \) using Lemma 2”
11. for \( v \in A \) do
12. \( G \leftarrow \text{MINBDNF}(F, k, B \cup \{v\}) \)
13. if \( G \neq \text{nill} \) and \(|G| < |G_{\text{min}}| \) then
14. \( G_{\text{min}} \leftarrow G \)
15. if \( |G_{\text{min}}| \leq k \) then return \( G_{\text{min}} \)
16. return \( \text{nill} \).

We are now ready to show our main tractability result.

**Theorem 3.** Let \( C \in \mathcal{F}^\sim \). Then, the problems \( C \)-BDNF and \( C \)-BT are fixed-parameter tractable.

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

It remains to analyse the runtime of the algorithm. Since every execution of \( \text{minBDNF} \) 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 calls. Moreover, the time required for one call of \( \text{minBDNF} \) is easily seen to be dominated by the time required by Line 2 to compute a smallest \( C \)-BDNF for \( F \) using only variables in \( B \) using Proposition 3, which is at most \( O(2^{|B|+1} + 3^{|B|}|F|^{|G|}) \). Therefore, the total runtime of the algorithm is at most \( O(|A|^{k-1}(2^{|B|+1} + 3^{|B|}|F|^{|G|})) \), which because \( |A| \) is bounded by a function of \( k \) (for all classes \( C \in \mathcal{F}^\sim \), due to Lemma 2) shows that \( C \)-BDNF is in \( \text{FPT}. \)

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

**Theorem 4 (⋆).** Let \( C = \text{HORN} \cup \text{HORN}^{-1} \). Then, the problems \( C \)-BT and \( C \)-BDNF are \( \text{W}[2]\)-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 (aisi), (ii–iii) graph coloring (flat, pret), (iv) logistics car configuration (daimler) [Sinz et al., 2003], (v) parity function learning (parity), (vi) inductive inference (inductive), (vii) planning (blocksworld), (viii) pigeon hole (pigeon), 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 \( \text{HORN}^1 \) and \( \text{RHORN}^1 \), for which already the BS problem is known to be \( \text{W}[1]\)-hard.

\(^1\)https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html
We compute the SAT encodings using Python 3.8.0 and PySAT 1.6.0\textsuperscript{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 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).

### Table 1: Comparison between backdoor DNFs and backdoor trees for several classes and groups of instances.

<table>
<thead>
<tr>
<th>Group</th>
<th>HORN(^{1})</th>
<th>RHORN(^{1})</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Size</td>
<td>Total</td>
</tr>
<tr>
<td>ais</td>
<td>87/1051</td>
<td>2/2</td>
</tr>
<tr>
<td>blocksworld</td>
<td>82/607</td>
<td>2/2</td>
</tr>
<tr>
<td>daimler</td>
<td>140/1877</td>
<td>3/3</td>
</tr>
<tr>
<td>flat</td>
<td>150/545</td>
<td>99/99</td>
</tr>
<tr>
<td>inductive</td>
<td>288/5077</td>
<td>16/16</td>
</tr>
<tr>
<td>parity</td>
<td>201/803</td>
<td>10/10</td>
</tr>
<tr>
<td>pigeon</td>
<td>74/322</td>
<td>5/5</td>
</tr>
<tr>
<td>pret</td>
<td>105/280</td>
<td>8/8</td>
</tr>
<tr>
<td>tw</td>
<td>222/965</td>
<td>9/12</td>
</tr>
<tr>
<td>vc</td>
<td>175/355</td>
<td>38/38</td>
</tr>
</tbody>
</table>

\(^\text{2}\)https://pySAThq.github.io

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 [Dvorak 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 questions for future work.
References


