Journal of Computer and System Sciences 144 (2024) 103547

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

journal homepage: www.elsevier.com/locate/jcss

Backdoor DNFs ☆

Sebastian Ordyniak [b,*], Andre Schidler [a], Stefan Szeider [a]

[a] Algorithms and Complexity Group, TU Wien, Vienna, Austria
[b] Algorithms and Complexity Group, University of Leeds, UK

ARTICLE INFO

Article history:
Received 31 July 2023
Received in revised form 28 February 2024
Accepted 6 May 2024
Available online 21 May 2024

Keywords:
Parameterized complexity
Backdoor sets/trees/DNFs
Boolean satisfiability
Distance to triviality

ABSTRACT

We introduce backdoor DNFs, as a tool to measure the theoretical hardness of CNF formulas. Like backdoor sets and backdoor trees, backdoor DNFs are defined relative to a tractable class of CNF formulas. Each conjunctive term of a backdoor DNF defines a partial assignment that moves the input CNF formula into the base class. Backdoor DNFs are more expressive and potentially smaller than their predecessors backdoor sets and backdoor trees. We establish the fixed-parameter tractability of the backdoor DNF detection problem. Our results hold 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" [66]. State-of-the-art SAT solvers routinely solve instances with millions of clauses and variables [29]. This is in stark contrast to the theoretical intractability of SAT. The problem is not just NP-complete [14]; the Exponential-Time Hypothesis [44], a standard complexity-theoretic assumption, asserts that there is no algorithm that solves every 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 [4,35,55] and decomposability [36,45,51]. The notion of a *backdoor set*, introduced by Williams *et al.* [67], 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.

Conceptually, backdoor sets are closely related to the concept of *distance to triviality*, a general methodology to parameterize problems, proposed by Guo, Hüffner, and Niedermeier in a paper that appeared in the first edition of IWPEC [42] in 2004. They presented this methodology with several case studies of distance from triviality parame-

☆ A preliminary version of this paper appeared in the Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI 2021) [57].

* Corresponding author.
E-mail addresses: s.ordyniak@leeds.ac.uk (S. Ordyniak), aschidler@ac.tuwien.ac.at (A. Schidler), sz@ac.tuwien.ac.at (S. Szeider).

https://doi.org/10.1016/j.jcss.2024.103547

terizations (concerning Clique, Power Dominating Set, Set Cover, and Longest Common Subsequence). Over the last two decades, this methodology has become a rich source of interesting and useful parameters for graph problems. [1–3,5,7–13,17,18,20–22,24,26–28,33,37,43,46,47,49,53,54,58] gives an incomplete list of work that uses the distance to triviality method. Historically, it is a striking coincidence that in the same year when Guo et al. proposed the distance to triviality method, Nishimura et al. presented the first study of backdoor sets under the parameterized complexity framework at the SAT conference [56].

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.* [56]. They showed that backdoor set detection is FPT for the fundamental base classes Horn and 2CNF. Gasper and Szeider [40] 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, and as a distance to triviality measure (distance to $\mathcal{C}$) not fully satisfying. Samer and Szeider [59] 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, as we shall show, a backdoor tree with the smallest number of leaves is not necessarily based on a backdoor set of the smallest cardinality. Samer and Szeider [59] 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 parameter related to backdoor sets, which gives rise to a distance to triviality measure that can be significantly smaller than the distance measured by the number of leaves of a backdoor tree. The new distance 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 [59], 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 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 [59], 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* [38]. 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

2.1. Parameterized complexity

We study the complexity of problems in a two-dimension setting. Each problem instance (I,k) consists of a main part I of bit size n and a parameter k, a non-negative integer. The problem is *fixed-parameter tractable* (or **FPT** for short) if there exists a computable function f and a constant c, such that the problem can be solved in time $f(k)n^c$. The problem is *XP-tractable* if it can be solved in time $n^{f(k)}$. In both cases, if k is a constant, then the stated running times are polynomial. However, in the former case, the order of the polynomial, c, is independent of the parameter k; in the latter case, it depends on k. Therefore, FPT is much preferred, as it provides better scalability in the parameter. By showing that a problem is hard for one of the parameterized complexity classes $\mathrm{W}[1], \mathrm{W}[2], \ldots$ we can get strong theoretical evidence that the problem is

not FPT. This is similar to showing that a problem is NP-hard provides evidence that is not solvable in polynomial time. We refer to text books and survey papers for more background on parameterized complexity [16,23,32,41].

2.2. 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). Moreover, for a clause C of F and an assignment τ not satisfying C, we denote by $C[\tau]$ the clause obtained from C after removing all literals assigned by 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 $x \in X$.

2.3. 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 (Horn$_{-1}$), i.e., the family of all CNF formulas having at most one positive (negative)literal per clause,

Let $\mathcal{D} = \{2\text{CNF}, \text{Horn}, \text{Horn}_{-1}\}$. The three considered classes are the most important of the six classes considered by Schaefer [62]: 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 $\mathcal{C} = \bigcup_{D \in \mathcal{D}'} D$ for $\mathcal{D}' \subseteq \mathcal{D}$, as has been first considered by Gaspers *et al.* [38]. 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 [40,50]. A base class $\mathcal{C}$ can also be extended by adding *empty clause detection* [19,65]. This gives rise to the base class $\mathcal{C}^{\{\}} = \{ F : F \in \mathcal{C} \text{ or } F \text{ contains the empty clause } \}$.

Let $\mathcal{C} = \bigcup_{D \in \mathcal{D}'} D$ for $\mathcal{D}' \subseteq \mathcal{D}$ be a heterogeneous base class and let F be a CNF formula. With a slight abuse of notation, we denote by $F \setminus \mathcal{C}$, the formula obtained from F after removing all clauses c with $\{c\} \in \mathcal{C}$.

2.4. 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 of F* if $F[\tau] \in \mathcal{C}$ for every truth assignment $\tau : B \to \{0, 1\}$; our backdoor sets are usually referred to as strong backdoor sets in the literature. For each base class $\mathcal{C}$ we consider the following problem:

$\mathcal{C}$-Backdoor Set	**Parameter:** k
Input: A CNF formula F and a non-negative integer k.	
Question: Does F has a $\mathcal{C}$-backdoor set of cardinality at most k?	

Let B be a $\mathcal{C}$-backdoor set of a CNF formula F. B is *smallest* if F has no $\mathcal{C}$-backdoor set that is smaller than B; B is *minimal* if F has no $\mathcal{C}$-backdoor set 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 $\mathcal{C}$-backdoor set 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 $\mathcal{C}$-Backdoor Set (which is also known to be NP-hard for every base class $\mathcal{C} = \bigcup_{D \in \mathcal{D}'} D$ [15], where $\mathcal{D}' \subseteq \mathcal{D}$ and $\mathcal{D}' \neq \emptyset$), given by Gaspers *et al.* [38] and constitutes a crucial prerequisite for our algorithms for backdoor trees and backdoor DNFs.

Proposition 1 *([38]). Let F be a CNF formula, $B \subseteq v(F)$, and $\mathcal{C} \in \{2\text{CNF}, \text{Horn}, \text{Horn}_{-1}, 2\text{CNF} \cup \text{Horn}, 2\text{CNF} \cup \text{Horn}_{-1}\}$. 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:*

- $|W| \leq 3$ *if* $\mathcal{C} = 2\text{CNF}$,
- $|W| \leq 2$ *if* $\mathcal{C} \in \{\text{Horn}, \text{Horn}_{-1}\}$
- $|W| \leq 5$ *if* $\mathcal{C} \in \{2\text{CNF} \cup \text{Horn}, 2\text{CNF} \cup \text{Horn}_{-1}\}$.

Note, however, that RHorn-Backdoor Set is W[2]-hard [40] and $2\text{CNF}^{\{\}}$-Backdoor Set, $\text{Horn}^{\{\}}$-Backdoor Set, and $\text{Horn}_{-1}{}^{\{\}}$-Backdoor Set are W[1]-hard [65].

2.5. 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 and we define the *size* of a T be the number of nodes 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 of F* if $F[\tau_v] \in \mathcal{C}$ for every leaf v of T. A $\mathcal{C}$-backdoor tree T of F with the smallest number of leaves (in the following, let $|T|$ denote the number of leaves), is a *smallest* $\mathcal{C}$-backdoor tree of F. We consider the following parameterized problem:

$\mathcal{C}$-Backdoor Tree	**Parameter:** k
Input: A CNF formula F and a non-negative integer k.	
Question: Does F have a $\mathcal{C}$-backdoor tree with at most k leaves?	

We need the following auxiliary lemma that can be shown using a simple brute force-algorithm.

Lemma 1. *Let A be a set of variables of size a. Then the number of DTs of size at most s that use only variables in A is at most a^{2s+1} and those can be enumerated in time $\mathcal{O}(a^{2s+1})$.*

Proof. We start by counting the number of trees T with n nodes that can potentially underlie a DT with n nodes. Note that there is one-to-one correspondence between trees T that underlie a DT with n nodes and unlabelled rooted ordered binary trees with n nodes (where ordered refers to an ordering of the at most 2 child nodes). Since it is known that the number of unlabelled rooted ordered binary trees with n nodes is equal to the n-th Catalan number C_n and that those trees can be enumerated in time $\mathcal{O}(C_n)$ [64], we already obtain that we can enumerate all of the at most C_n possible trees T underlying a DT of size n in time $\mathcal{O}(C_n)$. Therefore, there are at most sC_s possible trees of size at most s that can underlie a DT with at most s nodes and those can be enumerated in time $\mathcal{O}(sC_s)$. It now remains to bound the number of possible variable assignments $v(t)$ for these trees. Since we can assume that $a \geq 2$, we obtain that the number of possible variable assignments of a tree T with n nodes is at most a^n. Taking everything together, we obtain that there are at most $sC_s a^s \leq s4^s a^s \leq a^{2s+1}$ many DTs of size at most s using only variables in A and those can be enumerated in time $\mathcal{O}(a^{2s+1})$. □

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

Proposition 2. *Let B be a $\mathcal{C}$-backdoor set for a CNF formula F for some base class $\mathcal{C}$. Then, a smallest $\mathcal{C}$-backdoor tree for F using only variables in B can be computed in time $|B|^{2|B|+1}|F|^{\mathcal{O}(1)}$.*

Proof. Note that every $\mathcal{C}$-backdoor tree for F that uses only variables in B can have size at most $2^{|B|}$. Moreover, given a DT T using only variables from B, we can test in time $(|T|-1)|F|^{\mathcal{O}(1)}$, whether T constitutes a $\mathcal{C}$-backdoor tree for F by testing for each leaf l of T whether $F[\tau_l] \in \mathcal{C}$ (which can be achieved in time $|F|^{\mathcal{O}(1)}$).

Lemma 1 now shows that we can enumerate all DTs using only variables in B in time $\mathcal{O}(|B|^{2|B|+1})$. Therefore, we can find the smallest $\mathcal{C}$-backdoor tree for F by enumerating all DTs using only variables in B and testing for each of them whether they form a $\mathcal{C}$-backdoor tree for F in the stated running time. □

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 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* $\mathcal{C}$-backdoor DNF for F if $|G|$ is minimal over all $\mathcal{C}$-backdoor DNFs 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	**Parameter:** k
Input: A CNF formula F and a non-negative integer k.	
Question: Does F have a $\mathcal{C}$-backdoor DNF of size at most k?	

If $\mathcal{C}$ is a base class and one is given a $\mathcal{C}$-backdoor DNF 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 $\mathcal{C}$-backdoor DNF for F for every $\mathcal{C}$-backdoor tree for F where L is the set of leaves of T, one can consider backdoor trees as restricted version of backdoor DNFs (similar to how backdoor sets are a restricted version of backdoor trees). However, backdoor DNFs can be smaller by an arbitrary number than backdoor trees (which in turn can be smaller by an arbitrary number than backdoor sets as shown in [59]), which makes them better suited as shortcuts to tractability for Boolean Satisfiability, as shown by the following theorem; we conjecture that it is possible to show that there is an exponential difference in the size of backdoor DNFs and backdoor trees are, however, at the moment unable to show this. This theoretical result is also strongly reflected in our experimental evaluation in Section 5.

Theorem 1. *For every $s \geq 1$, there is a CNF formula F^s, whose size is polynomial in s, such that a smallest* Horn*-backdoor DNF for F^s contains at least $s-2$ fewer variables than a smallest* Horn*-backdoor tree for F^s.*

Proof. For every $s \geq 1$, we will construct a CNF formula F^s such that:

- F^s has a Horn-backdoor DNF of size $s+2$ but
- a smallest Horn-backdoor tree for F^s has size $2s$.

This then implies the theorem.

Let $r = 2s$ and F^s be the CNF formula with variables $\{x_1, \ldots, x_s\} \cup \{h_i^j : 0 \leq i \leq s \wedge 1 \leq j \leq r\}$ containing the following clauses:

- the clause $C_P^j = \{x_1, \ldots, x_s, h_0^j\}$ for every j with $1 \leq j \leq r$ and
- the clause $C_i^j = \{\neg x_1, \neg x_2, \ldots, \neg x_s\} \setminus \{\neg x_i\} \cup \{x_i\} \cup \{h_i^j\}$ for every i and j with $1 \leq i \leq s$ and $1 \leq j \leq r$.

Let G^s be the set containing the following assignments (of $\{x_1, \ldots, x_s\}$):

- the assignment $(x_1 = 1, \ldots, x_s = 1)$,
- the assignment $(x_1 = 0, \ldots, x_s = 0)$ and
- the assignment $(x_i = 0, x_{i \mod s+1} = 1)$ for every i with $1 \leq i \leq s$.

We start by showing that G^s is a Horn-backdoor DNF for F^s. Clearly, G^s_{DNF} is a tautology. Moreover, $F[(x_1 = 1, \ldots, x_s = 1)] \in$ Horn and $F[(x_1 = 0, \ldots, x_s = 0)] \in$ Horn, since both assignments set all but one positive literal of every clause in F^s. Now consider the assignment $\tau_i = (x_i = 0, x_{i+1 \mod s} = 1)$ for i with $1 \leq i \leq s$. Then, τ_i clearly satisfies the clauses $C_P^j = \{x_1, \ldots, x_s, h_0^j\}$. Moreover, τ_i also satisfies the clauses C_l^j for every $l \neq i$ since x_i occurs negatively in those clauses. Therefore, it only remain the clauses C_i^j, which are not satisfied by τ_i, but $C_i^j[\tau_i] \in$ Horn since it has only one positive literal, i.e., the literal h_i^j.

We now show that F^s has indeed a Horn-backdoor tree of size at most $2s$. To see this consider the DT T^s having the following leaves:

- one leaf l_0 with $\tau_{l_0} = (x_1 = 0, \ldots, x_s = 0)$,
- one leaf l_0^i with $\tau_{l_0^i} = (x_1 = 0, \ldots, x_{i-1} = 0, x_i = 1)$ for every i with $1 < i \leq s$,
- one leaf l_1 with $\tau_{l_1} = (x_1 = 1, \ldots, x_s = 1)$ and
- one leaf l_1^i with $\tau_{l_1^i} = (x_1 = 1, \ldots, x_{i-1} = 1, x_i = 0)$ for every i with $1 < i \leq s$.

Informally, T^s consists of two paths, one for the all zero and one for the all one assignment, containing nodes for all variables $x_1, \ldots, x_s$ starting at the root, which is labelled with x_1. Then, T^s is easily seen to be a Horn-backdoor tree for F, since we have already shown that $F[\tau_l] \in$ Horn for any leaf of T^s, when we showed that G^s is a Horn-backdoor DNF. Since $|T^s| = 2s$ it only remains to show that there is no smaller Horn-backdoor tree for F.

Towards showing this, let T be any minimal Horn-backdoor tree for F. Then, w.l.o.g., we can assume that $v(T) \subseteq \{x_1, \ldots, x_s\}$ since if $h_i^j \in v(T)$, then we also need that $h_i^\ell \in v(T)$ for any ℓ with $1 \leq \ell \leq r$, which implies that $|T| > r = 2s$. Let l_0 be the leaf of T reached from the root by always choosing the left child and let l_1 be the leaf of T reached from the root

by always choosing the right child. Then, τ_{l_0} is the all zero assignment for some subset $X_0 \subseteq \{x_1, \ldots, x_s\}$ and similarly τ_{l_0} is the all one assignment for some subset $X_1 \subseteq \{x_1, \ldots, x_s\}$. We claim that $X_0 = X_1 = \{x_1, \ldots, x_s\}$, which implies that T has size at least $2s$. Suppose not, then X_0 misses a variable x_{i_0} or X_1 misses a variable x_{i_1}. In the former case, $F[\tau_{l_0}] \notin$ Horn because it contains the clauses C_P^j each having at least two positive literals (x_{i_0} and h_0^j). Similarly, in the latter case, $F[\tau_{l_1}] \notin$ Horn because it contains the clauses $C_{i_1}^j[\tau_{l_1}]$ each having exactly two positive literals (x_{i_1} and $h_{i_1}^j$). □

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

Observation 1. *Let G be a $\mathcal{C}$-backdoor DNF of a CNF formula F. Then, $v(G)$ is a $\mathcal{C}$-backdoor set. Similarly, if T is a $\mathcal{C}$-backdoor tree for F, then $v(T)$ is a $\mathcal{C}$-backdoor set.*

Proof. The observation for backdoor trees was already shown in [59]. Assume for a contradiction that this is not the case, i.e., there is an assignment $\tau : v(G) \to \{0,1\}$ such that $F[\tau] \notin \mathcal{C}$. Because G_{DNF} is a tautology, there is an assignment $\tau' \in G$ that is consistent with τ. Therefore, $F[\tau'] \notin \mathcal{C}$ because neither is $F[\tau]$, contradicting our assumption that G is a $\mathcal{C}$-backdoor DNF of F. □

Observation 2. *For each $\mathcal{C}$-backdoor DNF or $\mathcal{C}$-backdoor tree G of a CNF formula F we have $|v(G)| \leq |G| - 1$.*

Proof. The observation for backdoor trees was already shown in [59]. For backdoor DNFs it follows because G_{DNF} is a tautological DNF. □

Analogously to Proposition 2 for backdoor trees, the next result asserts that computing a smallest $\mathcal{C}$-backdoor DNF can be done efficiently if the set of allowed variables is small.

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

Proof. We first compute the set $\mathcal{A}$ of all partial assignments $\alpha : B' \to \{0,1\}$ with $B' \subseteq B$ such that $F[\alpha] \in \mathcal{C}$ in time $3^{|B|}|F|^{\mathcal{O}(1)}$. Then, clearly any $\mathcal{C}$-backdoor DNF for F is a subset of $\mathcal{A}$. Therefore, we can obtain a smallest $\mathcal{C}$-backdoor DNF for F by enumerating all of the at most $2^{3^{|B|}}$ subsets of $\mathcal{A}$ and checking for each of them whether it constitutes a $\mathcal{C}$-backdoor DNF for F. We then return the smallest such subset of $\mathcal{A}$. Note that testing whether a given subset A of $\mathcal{A}$ is a $\mathcal{C}$-backdoor DNF can be achieved by testing whether the formula A_{DNF} is a tautology in time $\mathcal{O}(2^{|B|})$. Therefore, the total run-time of our algorithm is as stated. □

4. Finding backdoor DNFs and backdoor trees

In this section, we will provide a complete classification of the parameterized complexity of $\mathcal{C}$-Backdoor Tree and $\mathcal{C}$-Backdoor DNF for every base class $\mathcal{C}$ such that $\mathcal{C} = \bigcup_{D \in \mathcal{D}'} D$, where $\mathcal{D}' \subseteq \mathcal{D}$ and $\mathcal{D}' \neq \emptyset$. 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[2]). We start by giving our FPT-algorithms and then show that both problems are W[2]-hard for the case that $\mathcal{C} =$ Horn $\cup$ Horn$_{-1}$.

Let $\mathcal{D}_+$ be the set of all these base classes, i.e., $\mathcal{D}_+ = \{2\text{CNF}, \text{Horn}, \text{Horn}_{-1}, 2\text{CNF} \cup \text{Horn}, 2\text{CNF} \cup \text{Horn}_{-1}\}$. 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 backdoor DNF (or backdoor tree) of size at most k and for each of those sets, we can compute a smallest backdoor DNF (or backdoor tree) that uses only those variables in FPT-time. This also illustrates that the main challenge that we have to overcome is to design an FPT-procedure to enumerate all sets of variables that can potentially be used by a smallest backdoor DNF (or backdoor tree). Given Observation 1, one might think that any smallest backdoor DNF (or backdoor tree) uses only the variables of a smallest backdoor set, 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 backdoor sets of size at most k in FPT-time. Unfortunately, this is not the case as asserted by the following theorem.

Theorem 2. *For every $\mathcal{C} \in \mathcal{D}_+$ and every $s \geq 1$, there is a CNF formula $F_s^{\mathcal{C}}$, whose size is exponential in s, such that $F_s^{\mathcal{C}}$ has a $\mathcal{C}$-backdoor DNF ($\mathcal{C}$-backdoor tree) of size $s+2$, but any $\mathcal{C}$-backdoor DNF ($\mathcal{C}$-backdoor tree), whose variables form a minimal $\mathcal{C}$-backdoor set for $F_s^{\mathcal{C}}$, has size at least 2^s.*

Proof. We start by showing the lemma for the case that $\mathcal{C} =$ Horn. 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-backdoor sets, 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-backdoor set for F_s^{Horn} because of the clauses $\{a_i, p\}$,
- any Horn-backdoor set 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 $\{a_1, \ldots, a_s, q_j, \neg p\}$).

Therefore, every minimal Horn-backdoor set 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-backdoor tree and Horn-backdoor DNF 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-backdoor tree and every Horn-backdoor DNF that uses only the variables in B, i.e., that it has size at least 2^s. This is because $F_s^{\text{Horn}}[\alpha] \notin \text{Horn}$ 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 leaf of a Horn-backdoor tree and similarly every term of a Horn-backdoor DNF 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-backdoor tree and a Horn-backdoor DNF of size at most $s + 2$.

Towards showing this, let T be the DT with variable p at its root. Because $F[(p = 0)] \in \text{Horn}$, the left child of p in T is a leaf. Moreover, because $F[(p = 1)]$ only consists of the clauses $\{a_1, \ldots, a_s, q_i, \neg p\}$, we obtain that T can be completed to a Horn-backdoor tree for F_s^{Horn} by adding a path of lengths s to the right child of p in T, whose variables are $a_1, \ldots, a_s$ such that the node of variable a_i is the left child of the node of variable a_{i-1} and the right child of the node of variable a_i is a leaf (since setting a_i to 1 satisfies all remaining clauses in $F[(p = 1)]$. Clearly, T has exactly $s + 2$ leaves, as required. Finally, we obtain a Horn-backdoor DNF for F_s^{Horn} of the same size $s + 2$, by taking 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 $(p = 1, a_i = 1)$.

The formulas for the remaining base classes in $\mathcal{D}_+ \setminus \{\text{2CNF}\}$ are based on a similar construction. Namely:

- $F_s^{\text{Horn}_{-1}}$ is the formula obtained from F_s^{Horn} after complementing the literals of every clause.
- $F_s^{\text{2CNF} \cup \text{Horn}}$ is the formula obtained from F_s^{Horn} after adding the negated literals of (the same) $2^s + 3$ fresh variables to every clause. Note that $F_s^{\text{2CNF} \cup \text{Horn}}$ and F^{Horn} still have the same Horn-backdoor DNFs (Horn-backdoor trees). Moreover, because no clause can become 2CNF by assigning at most 2^s variables, it follows that any 2CNF $\cup$ Horn-backdoor DNF (2CNF $\cup$ Horn-backdoor tree) of size at most 2^s for $F^{\text{2CNF} \cup \text{Horn}}$ is also a Horn-backdoor DNF (Horn-backdoor tree) for $F^{\text{2CNF} \cup \text{Horn}}$.
- $F_s^{\text{2CNF} \cup \text{Horn}_{-1}}$ is the formula obtained from $F_s^{\text{2CNF} \cup \text{Horn}}$ after complementing the literals of every clause.

In the case that $\mathcal{C} = \text{2CNF}$, the formula F_s^{2CNF} is slightly different. That is, F_s^{2CNF} has variables $\{p_1, p_2\} \cup \{a_i : 1 \leq i \leq s\} \cup \{q_j, r_j : 1 \leq j \leq r\}$ for $r = 2^s - s$ and the following clauses:

- a clause $\{a_i, p_1, p_2\}$ for every i with $1 \leq i \leq s$ and
- the clauses $\{a_1, \ldots, a_s, q_j, r_j\}$ for every j with $1 \leq j \leq r$.

As before $B = \{a_1, \ldots, a_s\}$ is a minimal 2CNF-backdoor set for F. Moreover, every 2CNF-backdoor set for F that misses at least one a_i has to contain either p_j or r_j for every j with $1 \leq j \leq r$ (because of the clause $\{a_1, \ldots, a_s, q_j, r_j\}$). Therefore, every minimal 2CNF-backdoor set that is not B has size at least $s + r = s + 2^s - s = 2^s$, which together with Observation 1 implies that any 2CNF-backdoor tree and 2CNF-backdoor DNF 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-backdoor tree and every Horn-backdoor DNF that uses only the variables in B, i.e., that it has size at least 2^s. This is because $F[\alpha] \notin \text{2CNF}$ for every partial assignment $\alpha : B' \to \{0, 1\}$, where $B' \subsetneq B$ (because of the clause $\{a_i, p_1, p_2\}$, where $a_i \in B \setminus B'$). Therefore, every leaf of a 2CNF-backdoor tree and similarly every term of a 2CNF-backdoor DNF 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^{2CNF} has a 2CNF-backdoor tree and a 2CNF-backdoor DNF of size at most $2s$. Towards showing this, let T be the DT with variable p_1 at its root. Then, $F[(p_1 = 0)] \setminus \text{2CNF}$ and $F[(p_1 = 1)]$ only contain the clauses $\{a_1, \ldots, a_s, q_j, r_j\}$. Let P be a path of length s whose variables are $a_1, \ldots, a_s$ such that the node of variable a_i is the left child of the node of variable a_{i-1} and the right child of the node of variable a_i is a leaf (since setting a_i to 1 satisfies all clauses $\{a_1, \ldots, a_s, q_j, r_j\}$). Therefore, similar to the case for $\mathcal{C} = \text{Horn}$, we can complete T to a 2CNF-backdoor tree by P to the left and the right child of the root with variable p_1. Clearly, T has exactly $2(s + 1)$ leaves. Finally, we obtain a 2CNF-backdoor DNF for F_s^{2CNF} of the same size $2(s+1)$, by taking the following assignments: (1) the assignment $(p_1 = 0, a_1 = 0, \ldots, a_s = 0)$, (2) the assignment $(p_1 = 0, a_i = 1)$ for every i with $1 \leq i \leq s$, (3) the assignment $(p_1 = 1, a_1 = 0, \ldots, a_s = 0)$, (4) the assignment $(p_1 = 1, a_i = 1)$ for every i with $1 \leq i \leq s$. □

The theorem also shows that our backdoor trees can be arbitrarily smaller than the backdoor trees detected by Samer and Szeider's algorithm [59], which are only allowed to use subset-minimal $\mathcal{C}$-backdoor sets.

It is therefore not sufficient to enumerate all backdoor sets of a CNF formula F to identify a set of variables that is used by a smallest backdoor DNF (or backdoor tree). Nevertheless, Observation 1 still allows us to assume that we are given a backdoor set for F and as we will show next that this will be sufficient to identify all sets of variables that can lead to a smallest backdoor DNF (or backdoor tree). In particular, we will show next that if a smallest backdoor DNF (or backdoor tree) uses additional variables outside of a backdoor set, then the set of those additional variables has a special property (which we will later exploit to extend minimal backdoor sets), which we call useful. Let F be a CNF-formula and B a $\mathcal{C}$-backdoor set. We say that a set U of variables is *$\mathcal{C}$-useful* for B if $U = \emptyset$ or it holds that 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 $F[\alpha \cup \beta] \in \mathcal{C}$; note that if U is $\mathcal{C}$-useful for B then $U \setminus B$ is also $\mathcal{C}$-useful for B and therefore U can be assumed to be disjoint from B. The following lemma shows that the set of variables used by a minimal backdoor DNF (or minimal backdoor tree) for F that go beyond a backdoor set, needs to be useful.

Lemma 2. *Let G be a smallest term-minimal $\mathcal{C}$-backdoor DNF for F and let B be a $\mathcal{C}$-backdoor set contained in $v(G)$, then the set $U = v(G) \setminus B$ is $\mathcal{C}$-useful. If T is a smallest $\mathcal{C}$-backdoor tree for F and B is a $\mathcal{C}$-backdoor set contained in $v(T)$, then the set $U = v(T) \setminus B$ is $\mathcal{C}$-useful.*

Proof. We start by showing the lemma for backdoor DNFs. 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 consistent 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 $\mathcal{C}$-backdoor DNF 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 $\mathcal{C}$-backdoor DNF for F, contradicting our assumption that G is term-minimal.

It remains to show the lemma for the case of backdoor trees. 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 $F[\alpha] \notin \mathcal{C}$. 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.

- For every inner node t of T with $v = v(t) \in U$, let c be the left child of t if $\beta(v) = 1$ and let c be the right child of t otherwise. Remove the subtree rooted at c from T and let T'' be the tree obtained from T after applying this step exhaustively. Note that every node t of T'' with $v(t) \in U$ has exactly one child node.
- Let P be a path of maximum length in T'' consisting only of nodes t with $v(t) \in U$. We obtain the tree $T''|P$ by contracting the path P in T, i.e., we remove all nodes on P from T'' and if P does not contain the root of T'' we additionally 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 $v(t) \in U$.

Note that every non-leaf node t in T' has exactly two children and moreover $v(t) \in B$. We now show that T' is also $\mathcal{C}$-backdoor tree for F, which, because $U \neq \emptyset$, contradicts our assumption that T had minimum size. Suppose for a contradiction that this is not the case, i.e., there is a leaf l' of T' such that $F[\tau_{l'}] \notin \mathcal{C}$ and let l be the unique leaf in T that corresponds to l'. Then $F[\tau_{l'} \cup \beta] \notin \mathcal{C}$ and since τ_l is a sub-assignment of $\tau_{l'} \cup \beta$ also $F[\tau_l] \notin \mathcal{C}$ contradicting our assumption that T is a $\mathcal{C}$-backdoor tree for F. □

We will show next how we can efficiently find $\mathcal{C}$-useful sets for B for a given $\mathcal{C}$-backdoor set 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 for B is to be able to compute "small" $\mathcal{C}$-branching sets efficiently (i.e., FPT parameterized by $|B|$). The following lemma shows this for all base classes in $\mathcal{D}_+$.

Lemma 3. *Let $\mathcal{C} \in \mathcal{D}_+$ and let B be a $\mathcal{C}$-backdoor set for a CNF formula F. Then, a $\mathcal{C}$-branching set A such that:*

- $|A| \leq 2 \cdot 3^{|B|}$ *(if $\mathcal{C} = 2\text{CNF}$),*
- $|A| \leq 3^{|B|}$ *(if $\mathcal{C} \in \{\text{Horn}, \text{Horn}_{-1}\}$), and*
- $|A| \leq 5 \cdot 3^{|B|}$ *(if $\mathcal{C} \in \{2\text{CNF} \cup \text{Horn}, 2\text{CNF} \cup \text{Horn}_{-1}\}$)*

can be computed in time $\mathcal{O}(3^{|B|}|F|)$.

Proof. We start by showing the statement of the lemma for the case that $\mathcal{C} = 2\text{CNF}$. Let U be a $\mathcal{C}$-useful set for B, let $\beta : U \to \{0,1\}$ be an arbitrary assignment for U. Because U is $\mathcal{C}$-useful, there is an assignment $\alpha : B' \to \{0,1\}$ with $B' \subseteq B$ such

that $F[\alpha] \notin$ 2CNF but $F[\alpha \cup \beta] \in$ 2CNF. Therefore, U has to contain at least one variable from every clause in $F[\alpha] \setminus$ 2CNF. Therefore, every $\mathcal{C}$-useful set U for B has to contain at least one variable from every clause C in $F[\alpha] \setminus$ 2CNF for some assignment $\alpha : B' \to \{0,1\}$ with $F[\alpha] \notin$ 2CNF. Since B is also a deletion 2CNF-backdoor set, it holds that every such clause C contains at most two variables that are not in B. Therefore, we obtain the $\mathcal{C}$-branching set A by adding the at most two variables not in B of an arbitrary clause in $F[\alpha] \setminus$ 2CNF for every assignment $\alpha : B' \to \{0,1\}$ with $F[\alpha] \notin$ 2CNF.

We now show the statement of the lemma for the case that $\mathcal{C} \in \{\text{Horn}, \text{Horn}_{-1}\}$. In particular, we give the proof for the case that $\mathcal{C} = \text{Horn}$; the case for $\mathcal{C} = \text{Horn}_{-1}$ is analogously.

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-backdoor set 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.

We now show the statement of the lemma for the case that $\mathcal{C} = 2\text{CNF} \cup \text{Horn}$, the proof for the case that $\mathcal{C} = 2\text{CNF} \cup \text{Horn}_{-1}$ is analogously.

We first show that because B is a $\mathcal{C}$-backdoor set for F, it holds that every clause of F contains at most $|B| + 2$ positive literals.

Claim 1. *Every clause of F contains at most 2 positive literals, whose variables are not in B.*

Proof. Assume this is not the case and let C be a clause of F containing at least three positive literals, whose variables are not in B. Let $\alpha : B \to \{0,1\}$ be any assignment that does not satisfy C. Then, $C[\alpha] \in F[\alpha]$ and $C[\alpha]$ contains at least three positive literals, which implies that $F[\alpha] \notin \mathcal{C}$, contradicting our assumption that B is a $\mathcal{C}$-backdoor set for F. □

Let $\alpha : B' \to \{0,1\}$ be a partial assignment of B with $B' \subseteq B$ such that $F[\alpha] \notin \mathcal{C}$. Then, one of the following conditions holds:

(1) $F[\alpha]$ contains a clause C that is not in $\mathcal{C}$ or
(2) $F[\alpha]$ contains two clauses C and C' such that $C \in 2\text{CNF} \setminus \text{Horn}$ and $C' \in \text{Horn} \setminus 2\text{CNF}$.

Let P_α be the set of variables such that either:

- If $F[\alpha]$ satisfies (1), let C be an arbitrary clause in $F[\alpha] \setminus \mathcal{C}$. Moreover, let W be the set of at most two variables corresponding to positive literals of C that are not in B; the fact that there are at most two such variables follows from Claim 1. Then, P_α is any set of exactly three variables of C containing W.
- If $F[\alpha]$ satisfies (2), let C and C' be two arbitrary clauses in $F[\alpha]$ with $C \in 2\text{CNF} \setminus \text{Horn}$ and $C' \in \text{Horn} \setminus 2\text{CNF}$. Let W be the set of at most two variables corresponding to positive literals of C' that are not in B; the fact that there are at most two such variables follows from Claim 1 and let W' be the set of any three variables of C containing W. Then, P_α is equal to $v(C) \cup W'$.

Note that P_α can easily be computed in polynomial-time for any α and $|P_\alpha| \leq 5$. We claim that the set A obtained as the union of all sets P_α for every $\alpha : B' \to \{0,1\}$ with $B' \subseteq B$ and $F[\alpha] \notin \mathcal{C}$ is a $\mathcal{C}$-branching set for B, which concludes the proof of the statement of the lemma.

Towards showing this claim, let U be a $\mathcal{C}$-useful set for B and let $\beta_1 : U \to \{1\}$ be the assignment for $U \subseteq v(F)$ setting all variables to 1. Because U is $\mathcal{C}$-useful, there is an assignment $\alpha : B' \to \{0,1\}$ with $B' \subseteq B$ such that $F[\alpha] \notin \mathcal{C}$ but $F[\alpha \cup \beta_1] \in \mathcal{C}$. We distinguish two cases:

- $F[\alpha]$ satisfies (1), then there is a clause C in $F[\alpha] \setminus \mathcal{C}$ such that P_α contains 3 variables of C including all variables occurring positively in C and are not in B. Because $F[\alpha \cup \beta_1] \in \mathcal{C}$, we obtain that either:
  - C is satisfied by β_1. This implies that U contains at least one variable corresponding to a positive literal in C; this is because β_1 sets every variable in U to 1. Therefore, U contains at least one variable from P_α.
  - C is not satisfied by β_1. In this case U cannot contain any variables that correspond to positive literals in C. Because $C[\beta_1] \in \mathcal{C}$, we obtain that either:
    - $C[\beta_1] \in 2\text{CNF}$, which implies that $|C| \leq 2 + |U|$ and U contains all but at most two variables of C and therefore at least one variable from P_α, or

Algorithm 1 Main method for finding a smallest backdoor DNF.

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

function MINBDNF(F, k, B)
  $G_{\min} \leftarrow$ "compute a smallest $\mathcal{C}$-backdoor DNF for F using only variables in B using Proposition 3"
  if $|B| \geq k-1$ **then**
    if $G_{\min} =$ `nil` or $|G_{\min}| \leq k$ **then**
      return $G_{\min}$
    return `nil`
  if B is not a $\mathcal{C}$-BS for F **then**
    $A \leftarrow$ "compute a $\mathcal{C}$-backdoor branching set for B using Proposition 1"
  else
    $A \leftarrow$ "compute a $\mathcal{C}$-branching set for B using Lemma 3
  for $v \in A$ **do**
    $G \leftarrow$MINBDNF(F, k, $B \cup \{v\}$)
    if $G \neq$ `nil` and $|G| < |G_{\min}|$ **then**
      $G_{\min} \leftarrow G$
  if $|G_{\min}| \leq k$ **then return** $G_{\min}$
  return `nil`

$*$ $C[\beta_1] \in$ HORN, which is not possible because $C \notin$ HORN and U contains no variables corresponding to positive literals of C.

Therefore, if $F[\alpha]$ satisfies (1), we obtain that U contains at least one variable from P_α.

- Otherwise, there are clauses C and C' in $F[\alpha]$ with $C \in$ 2CNF $\setminus$ HORN and $C' \in$ HORN $\setminus$ 2CNF such that P_α contains all variables of C and exactly 3 variables of C' including all variables that occur positively in C' and are not in B. Because $F[\alpha \cup \beta_1] \in \mathcal{C}$, we obtain that either:
  - U contains at least one variable of C and therefore from P_α or
  - $F[\alpha \cup \beta_1] \in$ 2CNF and therefore either:
    $*$ C' is satisfied by β_1 and therefore U contains at least one variable corresponding to a positive literal from C' and therefore at least one variable from P_α or
    $*$ C' is not satisfied by β_1 but $C'[\beta_1] \in$ 2CNF. Therefore, $|C'| \leq 2 + |U|$ and U contains all but at most two variables of C and therefore at least one variable from P_α.

  Therefore, if F satisfies (2), we obtain that U contains at least one variable from P_α.

This shows that U contains at least one variable from P_α and therefore A is a $\mathcal{C}$-branching set for B. □

Algorithm 2 Main method for finding a smallest $\mathcal{C}$-backdoor tree.

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

function MINBT(F, k, B)
  $T_{\min} \leftarrow$ "compute a smallest $\mathcal{C}$-backdoor tree for F using only variables in B using Proposition 2"
  if $|B| \geq k$ **then**
    if $T_{\min} =$ `nil` or $|T_{\min}| \leq k$ **then**
      return $T_{\min}$
    return `nil`
  if B is not a $\mathcal{C}$-BS for F **then**
    $A \leftarrow$ "compute a $\mathcal{C}$-backdoor branching set for B using Proposition 1"
  else
    $A \leftarrow$ "compute a $\mathcal{C}$-branching set for B using Lemmas 3"
  for $v \in A$ **do**
    $T \leftarrow$MINBT(F, k, $B \cup \{v\}$)
    if $T \neq$ `nil` and $|T| < |T_{\min}|$ **then**
      $T_{\min} \leftarrow T$
  if $T_{\min} =$ `nil` or $|T_{\min}| \leq k$ **then return** $T_{\min}$
  return `nil`

We are now ready to show our main tractability result.

Theorem 3. *Let $\mathcal{C} \in \mathcal{D}_+$. Then, the problems $\mathcal{C}$-*Backdoor DNF *and $\mathcal{C}$-*Backdoor Tree *are fixed-parameter tractable.*

Proof. We start by presenting the algorithm for $\mathcal{C}$-Backdoor DNF, 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 $\mathcal{C}$-backdoor DNF for F that uses at least the variables in B and has size at most k; if no such $\mathcal{C}$-backdoor DNF exists, the algorithm returns `nil`. To solve $\mathcal{C}$-Backdoor DNF, the function **minBDNF** needs to be called with B being the emptyset.

The algorithm starts by computing a smallest $\mathcal{C}$-backdoor DNF for F that uses only the variables in B using Proposition 3; if no such $\mathcal{C}$-backdoor DNF exists because B is not a $\mathcal{C}$-backdoor set, $G_{\min}$ is set to `nil`. The algorithm then checks whether the size of B is already maximal, i.e., if $|B| \geq k-1$ then no more variables can be added to B since every $\mathcal{C}$-backdoor DNF of size at most k uses at most $k-1$ variables (Observation 2). If so, the best $\mathcal{C}$-backdoor DNF already found for B is returned provided it has size at most k. Otherwise, the algorithm checks in Line 7, whether B is already a $\mathcal{C}$-backdoor set. If not the algorithm computes a $\mathcal{C}$-backdoor branching set for B using Proposition 1 to ensure that a $\mathcal{C}$-backdoor set is eventually obtained. Otherwise, the algorithm computes a $\mathcal{C}$-branching set in Line 10 using Lemma 3. This ensures that in this case only variables belonging to a $\mathcal{C}$-useful set for B are added. After computing the respective branching set and storing it in the variable A, the algorithm now branches on the variables in A and for every such variable $v \in A$, the algorithm calls itself recursively for F, k, and B updated by adding the variable v. If any of these recursive calls returns a smaller $\mathcal{C}$-backdoor DNF than the current best, the algorithm stores the new best backdoor DNF in the variable $G_{\min}$. Finally, the algorithm returns the best $\mathcal{C}$-backdoor DNF if it has size at most k and otherwise it returns `nil`.

Towards showing the correctness of the algorithm consider the case that F has a $\mathcal{C}$-backdoor DNF of size at most k and let G be a smallest such $\mathcal{C}$-backdoor DNF. Because of Observation 2, $|v(G)| \leq k-1$. Moreover, because of Observation 1, $v(G)$ contains a minimal $\mathcal{C}$-backdoor set 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 $\mathcal{C}$-backdoor set, 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 $\mathcal{C}$-backdoor DNF of size $|G|$ in Line 2, which will eventually be returned. Otherwise, we obtain from Lemma 2 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 $\mathcal{C}$-backdoor set, 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 $\mathcal{C}$-backdoor DNF of size at most k. This is because $G_{\min}$ is only updated in Line 2 and only returned if its size is 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 calls. 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 $\mathcal{C}$-backdoor DNF 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 $\mathcal{C} \in \mathcal{D}_+$ due to Lemma 3) shows that $\mathcal{C}$-Backdoor DNF is in FPT.

The algorithm for $\mathcal{C}$-Backdoor Tree is illustrated in Algorithm 2. The algorithm works very similar to the algorithm for $\mathcal{C}$-Backdoor DNF with the only difference being that Line 2 uses Proposition 2 (instead of Proposition 3) to compute a smallest $\mathcal{C}$-backdoor tree for F that uses exactly the variables in B. □

The following theorem, whose proof is based on a reduction by Gaspers *et al.* [38], shows that the problems are W[2]-hard for the only class $\mathcal{C} = \text{Horn} \cup \text{Horn}_{-1}$.

Theorem 4. *Let $\mathcal{C} = \text{Horn} \cup \text{Horn}_{-1}$. Then, the problems $\mathcal{C}$-*Backdoor Tree *and $\mathcal{C}$-*Backdoor DNF *are* W[2]*-hard.*

Proof. We give a parameterized reduction from the W[2]-complete Hitting Set problem, which given a family $\mathcal{W}$ of subsets of a set U and an integer k, asks whether $\mathcal{W}$ has a *hitting set* $H \subseteq U$ with $|H| \leq k$, i.e., $H \cap W \neq \emptyset$ for every $W \in \mathcal{W}$. Given an instance $(\mathcal{W}, U, k)$ for Hitting Set, we construct a formula F as follows. The variables of F are $U \cup \{d_W^1, \ldots, d_W^{|U \setminus W|} : W \in \mathcal{W}\}$. For each set $W \in \mathcal{W}$, there is one clause $C_W = W \cup \{d_W^1, \ldots, d_W^{|U \setminus W|}\}$. There is also one clause $C_U = \{\neg u : u \in U\}$. This completes the description of the reduction.

We claim that $\mathcal{W}$ has a hitting set of size at most k if and only if the formula F has a $\mathcal{C}$-backdoor tree/backdoor DNF of size at most $k+1$. Suppose $X = \{x_1, \ldots, x_l\} \subseteq U$, $|X| \leq k$, is a hitting set for $\mathcal{W}$.

Let $G = \{(x_1 = 1, \ldots, x_\ell = 1), (x_1 = 0), \ldots, (x_\ell = 0)\}$. Moreover, let T be the DT that consists of a path on nodes with variables $x_1, \ldots, x_\ell$ such that the node with variable x_i is the right child of the node with variable x_{i-1} and the left child of every node is a leaf. We claim that T is a $\mathcal{C}$-backdoor tree and that G is a $\mathcal{C}$-backdoor DNF, which completes the proof for the forward direction since $|T| = |G| = k+1$. Towards showing that T is a $\mathcal{C}$-backdoor tree for F consider a leaf l of T. Then, either $\tau_l = (x_i = 0)$ for some i with $1 \leq i \leq \ell$ or $\tau_l = (x_1 = 1, \ldots, x_\ell = 1)$. In the former case, τ_l satisfies the clause C_U and therefore $F[\tau_l] \in \text{Horn}_{-1}$ since every other clause contains no negative literals. In the latter case, τ_l satisfies every

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 backdoor DNF and the number of leaves of the computed backdoor tree, σ^2 is the variance. *Size* shows the average number of variables/clauses; *Total* shows the number of instances for which a backdoor DNF could be computed.

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

clause C_W for $W \in \mathcal{W}$ because X is a hitting set for $\mathcal{W}$. Therefore, $F[\tau_l] \in$ HORN since the only remaining clause C_U has no positive literals. This shows that T is indeed a $\mathcal{C}$-backdoor tree for F and moreover it also shows that $F[\tau] \in \mathcal{C}$ for every $\tau \in G$. Moreover, since G_{DNF} is a tautology, we also obtain that G is a $\mathcal{C}$-backdoor DNF for F.

For the other direction, first suppose that T is a $\mathcal{C}$-backdoor tree of size at most $k+1$. Let l be the leaf of T that corresponds to setting all variables to 1 and let X be the set of variables that occur in T on the path from the root of T to l. Then, τ_l does not satisfy C_U, which implies that $F[\tau_l] \notin$ HORN and therefore τ_l has to satisfy every clause C_W for every $W \in \mathcal{W}$. Therefore, the set X' obtained from X by replacing each $d^i_W \in X$ for any i with $1 \leq i \leq |U \setminus W|$ by some variable from W is a hitting set of $\mathcal{W}$, which because $|X'| = |X| \leq |T| - 1$ completes the argument for backdoor trees.

Now suppose that G is a $\mathcal{C}$-backdoor DNF of size at most $k+1$. Let τ be any assignment in G that is consistent with the assignment of $v(G)$ that sets all variables to 1; τ must exists because G_{DNF} is a tautology. Then, τ does not satisfy C_U, which implies that $F[\tau_l] \notin$ HORN and therefore τ has to satisfy every clause C_W for every $W \in \mathcal{W}$. Therefore, the set X obtained from $v(\tau)$ by replacing each $d^i_W \in X$ for any i with $1 \leq i \leq |U \setminus W|$ by some variable from W is a hitting set of $\mathcal{W}$, which because $|X'| = |v(\tau)| \leq |G| - 1$ completes the argument for backdoor trees. Note that $|v(\tau)| \leq |G| - 1$ is a special case of $|v(G)| \leq |G| - 1$, which holds because of Observation 2. □

5. Experiments

We complement our theoretical results by experiments.[1] We compute backdoor DNFs and backdoor trees on a large number of CNF formulas, stemming from various applications like logistics, planning, and combinatorics. We collect instances from *SATLIB*[3], past SAT Competitions,[2] car configuration [63], as well as vertex cover and treewidth instances[3] from named graphs. The instances form ten groups: (i) all interval series (*ais*),[4] (ii–iii) graph colouring (*flat, pret*)[3], (iv) logistics car configuration (*daimler*) [63], (v) parity function learning (*parity*)[3], (vi) inductive inference (*inductive*)[3], (vii) planning (*blocksworld*)[3], (viii) pigeon hole (*pigeon*)[3], and (ix–x) vertex cover and treewidth for named graphs (*vc and tw*). To avoid the restriction to base classes that support fixed-parameter tractability, we base our experiments on SAT encodings. This allows us to use the base classes HORN$^{\{\}}$ and RHORN$^{\{\}}$, for which already the backdoor set problem is known to be W[1]-hard.

We compute the SAT encodings using Python 3.8.0 and PySAT 1.6.0.[5] 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

[1] Results and source code are available under https://doi.org/10.5281/zenodo.11259842 and https://github.com/ASchidler/backdoor_cube.
[2] http://www.satcompetition.org/.
[3] Generated with the encoding by Samer and Veith [61].
[4] https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html.
[5] https://pysathq.github.io.

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 **backdoor DNFs** is based on incremental SAT solving. It finds one potential term of a backdoor DNF 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 [6] for its negation. The found DNF is then inclusion-minimal but not necessarily of smallest cardinality. We compute **backdoor trees** using a recursive algorithm. The algorithm computes one branch of the tree at a time using a SAT solver call. Each branch has the first node's variable, the branch's root, set to true, and all subsequent variables set to false. The partial assignment represented by the current branch is extended. We use these partial assignments to reduce the number of clauses for each subsequent recursive call.

Backdoor trees and backdoor DNFs are either **restricted**, where they may only use variables from a given backdoor set, or **unrestricted**, where any variable can be used. For this purpose, we compute minimal deletion backdoor sets. While we do not use empty clause detection for the backdoor sets, we select among the backdoor sets those that maximize the number of empty clauses obtained by deleting the backdoor set's variables.

5.1. 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 backdoor set and discard instances based on the backdoor set'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 backdoor DNFs to be smaller than backdoor trees. Indeed, in Table 1 we see this comparison in terms of the ratio of the backdoor DNF size to backdoor tree size. The lower the ratio, the smaller the backdoor DNF in comparison to the respective backdoor tree.

We found the lowest ratios for the graph colouring 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 backdoor set size (21–26). *Inductive* are instances that are almost in RHORN and have a deletion backdoor set 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. Restricted backdoor DNFs have the advantage that the search space for the DNF is limited. Although this should speed up the search, our results suggest that the restriction impedes the search. For two thirds of the instances, we could not compute a restricted DNF inside the time limit, independent of the base class. Further investigation showed that 90% of the backdoor DNFs do not represent a minimal backdoor set. Restricting the backdoor DNF search to a minimum backdoor set will therefore generally not help the search. This is also supported by our result that restricted DNFs are often larger than unrestricted DNFs.

6. Conclusion

We have introduced backdoor DNFs as a versatile tool for representing the hidden structure in a SAT instance. The size of a smallest $\mathcal{C}$-backdoor DNF provides a distance to triviality measure (distance to $\mathcal{C}$) that can be much smaller than the distances provided by smallest $\mathcal{C}$-backdoor trees or smallest strong $\mathcal{C}$-backdoor sets. Our main theoretical results show that for fundamental base classes $\mathcal{C}$ for which the detection of strong $\mathcal{C}$-backdoor sets is FPT, also the detection of $\mathcal{C}$-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 [34,38,39], ASP [30,31], Temporal Logic [52], QBF [60] Abstract Argumentation [25], and Planning [48]. We think that many of these results can be lifted to backdoor DNFs. This provides several challenging research questions for future work.

CRediT authorship contribution statement

Sebastian Ordyniak: Conceptualization, Funding acquisition, Investigation, Methodology, Writing – original draft, Writing – review & editing. **Andre Schidler:** Conceptualization, Data curation, Investigation, Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **Stefan Szeider:** Conceptualization, Funding acquisition, Investigation, Methodology, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The paper contains links to all data and software used.

Acknowledgments

Schidler and Szeider acknowledge the support by the FWF (P36420, W1255) and WWTF (ICT19-065). Ordyniak acknowledges the support by the EPSRC (EP/V00252X/1).

References

[1] A. Agrawal, L. Kanesh, F. Panolan, M. Ramanujan, S. Saurabh, An FPT algorithm for elimination distance to bounded degree graphs, in: 38th International Symposium on Theoretical Aspects of Computer Science (STACS 2021), Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.
[2] A. Agrawal, L. Kanesh, F. Panolan, M. Ramanujan, S. Saurabh, A fixed-parameter tractable algorithm for elimination distance to bounded degree graphs, SIAM J. Discrete Math. 36 (2) (2022) 911–921.
[3] A. Agrawal, M. Ramanujan, On the parameterized complexity of clique elimination distance, in: 15th International Symposium on Parameterized and Exact Computation (IPEC 2020), Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.
[4] C. Ansótegui, M.L. Bonet, J. Giráldez-Cru, J. Levy, The fractal dimension of SAT formulas, in: Proc. IJCAR '14, in: LNCS, vol. 8562, Springer, 2014, pp. 107–121.
[5] E. Arrighi, H. Fernau, M. de Oliveira Oliveira, P. Wolf, Width notions for ordering-related problems, in: 40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2020, December 14-18, 2020, BITS Pilani, K K Birla Goa Campus, Goa, India (Virtual Conference), in: LIPIcs, vol. 182, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, pp. 9:1–9:18.
[6] A. Belov, M. Heule, J. Marques-Silva, MUS extraction using clausal proofs, in: Proc. SAT '14, in: LNCS, vol. 8561, Springer, 2014, pp. 48–57.
[7] M. Bentert, T. Fluschnik, A. Nichterlein, R. Niedermeier, Parameterized aspects of triangle enumeration, J. Comput. Syst. Sci. 103 (2019) 61–77.
[8] I. Bliznets, D. Sagunov, On happy colorings, cuts, and structural parameterizations, in: Graph-Theoretic Concepts in Computer Science: 45th International Workshop, WG 2019, Vall de Núria, Spain, June 19–21, 2019, Revised Papers 45, Springer, 2019, pp. 148–161.
[9] J. Bulian, A. Dawar, Graph isomorphism parameterized by elimination distance to bounded degree, Algorithmica 75 (2) (2016) 363–382.
[10] J. Bulian, A. Dawar, Fixed-parameter tractable distances to sparse graph classes, Algorithmica 79 (1) (2017) 139–158.
[11] L. Bulteau, F. Hüffner, C. Komusiewicz, R. Niedermeier, Multivariate algorithmics for NP-hard string problems, Bull. Eur. Assoc. Theor. Comput. Sci. 114 (2014).
[12] J. Choudhari, I.V. Reddy, On structural parameterizations of happy coloring, empire coloring and boxicity, in: WALCOM: Algorithms and Computation: 12th International Conference, WALCOM 2018, Dhaka, Bangladesh, March 3-5, 2018, Proceedings 12, Springer, 2018, pp. 228–239.
[13] P. Choudhary, V. Raman, Improved kernels for tracking path problem, preprint, arXiv:2001.03161, 2020.
[14] S.A. Cook, The complexity of theorem-proving procedures, in: Proc. STOC '71, Shaker Heights, Ohio, 1971, pp. 151–158.
[15] Y. Crama, O. Ekin, P.L. Hammer, Variable and term removal from Boolean formulae, Discrete Appl. Math. 75 (3) (1997) 217–230.
[16] M. Cygan, F.V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, S. Saurabh, Parameterized Algorithms, Springer, 2015.
[17] P. Damaschke, Dividing splittable goods evenly and with limited fragmentation, Algorithmica 82 (5) (2020) 1298–1328.
[18] B. Das, M.K. Enduri, M. Kiyomi, N. Misra, Y. Otachi, I.V. Reddy, S. Yoshimura, On structural parameterizations of firefighting, Theor. Comput. Sci. 782 (2019) 79–90.
[19] B.N. Dilkina, C.P. Gomes, A. Sabharwal, Tradeoffs in the complexity of backdoor detection, in: Proc. CP '07, in: LNCS, vol. 4741, Springer, 2007, pp. 256–270.
[20] Ö.Y. Diner, A.C. Giannopoulou, G. Stamoulis, D.M. Thilikos, Block elimination distance, Graphs Comb. 38 (5) (2022) 133.
[21] J. Dreier, S. Ordyniak, S. Szeider, CSP beyond tractable constraint languages, Constraints 28 (2023) 450–471.
[22] J. Dreier, S. Ordyniak, S. Szeider, SAT backdoors: depth beats size, J. Comput. Syst. Sci. 142 (2024).
[23] R.G. Downey, M.R. Fellows, Fundamentals of Parameterized Complexity, Texts in Computer Science, Springer, 2013.
[24] P. Dvořák, E. Eiben, R. Ganian, D. Knop, S. Ordyniak, Solving integer linear programs with a small number of global variables and constraints, preprint, arXiv:1706.06084, 2017.
[25] W. Dvorák, S. Ordyniak, S. Szeider, Augmenting tractable fragments of abstract argumentation, Artif. Intell. 186 (2012) 157–173.
[26] M.R. Fellows, B.M. Jansen, F. Rosamond, Towards fully multivariate algorithmics: parameter ecology and the deconstruction of computational complexity, Eur. J. Comb. 34 (3) (2013) 541–566.
[27] H. Fernau, F. Foucaud, K. Mann, U. Padariya, K.R. Rao, Parameterizing path partitions, in: International Conference on Algorithms and Complexity, Springer, 2023, pp. 187–201.
[28] H. Fernau, K. Mann, Hitting the romans, preprint, arXiv:2302.11417, 2023.
[29] J.K. Fichte, D.L. Berre, M. Hecher, S. Szeider, The silent (r)evolution of SAT, Commun. ACM 66 (6) (June 2023) 64–72.
[30] J.K. Fichte, S. Szeider, Backdoors to normality for disjunctive logic programs, ACM Trans. Comput. Log. 17 (1) (2015).
[31] J.K. Fichte, S. Szeider, Backdoors to tractable answer set programming, Artif. Intell. 220 (Mar. 2015) 64–103.
[32] J. Flum, M. Grohe, Parameterized complexity and subexponential time, Bull. Eur. Assoc. Theor. Comput. Sci. 84 (2004) 71–100.
[33] T. Fluschnik, H. Molter, R. Niedermeier, M. Renken, P. Zschoche, Temporal graph classes: a view through temporal separators, Theor. Comput. Sci. 806 (2020) 197–218.
[34] R. Ganian, M.S. Ramanujan, S. Szeider, Discovering archipelagos of tractability for constraint satisfaction and counting, ACM Trans. Algorithms 13 (2) (2017) 29:1–29:32.
[35] R. Ganian, S. Szeider, Community structure inspired algorithms for SAT and #SAT, in: Proc. SAT '15, in: LNCS, vol. 9340, Springer, 2015, pp. 223–237.
[36] R. Ganian, S. Szeider, New width parameters for model counting, in: SAT '17, in: LNCS, vol. 10491, Springer, 2017, pp. 38–52.
[37] M. Garlet Millani, H. Molter, R. Niedermeier, M. Sorge, Efficient algorithms for measuring the funnel-likeness of dags, J. Comb. Optim. 39 (2020) 216–245.
[38] S. Gaspers, N. Misra, S. Ordyniak, S. Szeider, S. Zivny, Backdoors into heterogeneous classes of SAT and CSP, J. Comput. Syst. Sci. 85 (2017) 38–56.
[39] S. Gaspers, S. Ordyniak, S. Szeider, Backdoor sets for CSP, in: The Constraint Satisfaction Problem: Complexity and Approximability, in: Dagstuhl Follow-Ups, vol. 7, Schloss Dagstuhl, 2017, pp. 137–157.
[40] S. Gaspers, S. Szeider, Backdoors to satisfaction, in: The Multivariate Algorithmic Revolution and Beyond, in: LNCS, vol. 7370, Springer, 2012, pp. 287–317.
[41] G. Gottlob, S. Szeider, Fixed-parameter algorithms for artificial intelligence, constraint satisfaction, and database problems, Comput. J. 51 (3) (2008) 303–325. Survey paper.

[42] J. Guo, F. Hüffner, R. Niedermeier, A structural view on parameterizing problems: distance from triviality, in: R. Downey, M. Fellows, F. Dehne (Eds.), 1st International Workshop on Parameterized and Exact Computation (IWPEC 2004), in: Lecture Notes in Computer Science, vol. 3162, Springer, 2004, pp. 162–173.
[43] D. Hermelin, Y. Itzhaki, H. Molter, R. Niedermeier, Temporal unit interval independent sets, in: 1st Symposium on Algorithmic Foundations of Dynamic Networks (SAND 2022), Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.
[44] R. Impagliazzo, R. Paturi, F. Zane, Which problems have strongly exponential complexity?, J. Comput. Syst. Sci. 63 (4) (2001) 512–530.
[45] S. Jamali, D. Mitchell, Improving SAT solver performance with structure-based preferential bumping, in: Proc. GCAI '17, in: EPiC Series in Computing, vol. 50, EasyChair, 2017, pp. 175–187.
[46] B.M. Jansen, J.J. de Kroon, Fpt algorithms to compute the elimination distance to bipartite graphs and more, in: International Workshop on Graph-Theoretic Concepts in Computer Science, Springer, 2021, pp. 80–93.
[47] A.S. Kare, I. Vinod Reddy, Parameterized algorithms for graph burning problem, in: International Workshop on Combinatorial Algorithms, Springer, 2019, pp. 304–314.
[48] M. Kronegger, S. Ordyniak, A. Pfandler, Backdoors to planning, Artif. Intell. 269 (2019) 49–75.
[49] A. Lassota, A. Łukasiewicz, A. Polak, Tight vector bin packing with few small items via fast exact matching in multigraphs, preprint, arXiv:2203.10077, 2022.
[50] H.R. Lewis, Renaming a set of clauses as a Horn set, J. ACM 25 (1) (Jan. 1978) 134–135.
[51] R. Mateescu, Treewidth in industrial SAT benchmarks, Technical Report MSR-TR-2011-22, Microsoft, February 2011.
[52] A. Meier, S. Ordyniak, M.S. Ramanujan, I. Schindler, Backdoors for linear temporal logic, Algorithmica 81 (2) (2019) 476–496.
[53] G.B. Mertzios, A. Nichterlein, R. Niedermeier, Fine-grained algorithm design for matching, Technical Report, 2016.
[54] G.B. Mertzios, A. Nichterlein, R. Niedermeier, The power of linear-time data reduction for maximum matching, Algorithmica 82 (12) (2020) 3521–3565.
[55] Z. Newsham, V. Ganesh, S. Fischmeister, G. Audemard, L. Simon, Impact of community structure on SAT solver performance, in: Proc. SAT '14, in: LNCS, vol. 8561, Springer, 2014, pp. 252–268.
[56] N. Nishimura, P. Ragde, S. Szeider, Detecting backdoor sets with respect to Horn and binary clauses, in: Proc. SAT '04, 2004, pp. 96–103.
[57] S. Ordyniak, A. Schidler, S. Szeider, Backdoor dnfs, in: Z. Zhou (Ed.), Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021, ijcai.org, 2021, pp. 1403–1409.
[58] I.V. Reddy, Parameterized algorithms for conflict-free colorings of graphs, Theor. Comput. Sci. 745 (2018) 53–62.
[59] M. Samer, S. Szeider, Backdoor trees, in: Proc. AAAI '08, AAAI Press, 2008, pp. 363–368.
[60] M. Samer, S. Szeider, Backdoor sets of quantified Boolean formulas, J. Autom. Reason. 42 (1) (2009) 77–97.
[61] M. Samer, H. Veith, Encoding treewidth into SAT, in: Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings, in: LNCS, vol. 5584, Springer, 2009, pp. 45–50.
[62] T.J. Schaefer, The complexity of satisfiability problems, in: Proc. STOC '78, ACM, 1978, pp. 216–226.
[63] C. Sinz, A. Kaiser, W. Küchlin, Formal methods for the validation of automotive product configuration data, Artif. Intell. Eng. Des. Anal. Manuf. 17 (1) (2003) 75–97.
[64] R. Stanley, E.W. Weisstein, Catalan Number, from Mathworld–a Wolfram Web Resource, 2015.
[65] S. Szeider, Matched formulas and backdoor sets, J. Satisf. Boolean Model. Comput. 6 (2008) 1–12.
[66] M.Y. Vardi, Boolean satisfiability: theory and engineering, Commun. ACM 57 (3) (Mar. 2014) 5.
[67] R. Williams, C. Gomes, B. Selman, Backdoors to typical case complexity, in: Proc. IJCAI '03, Morgan Kaufmann, 2003, pp. 1173–1178.