Journal of Computer and System Sciences 142 (2024) 103520

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

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

SAT backdoors: Depth beats size [☆]

Jan Dreier [a,*], Sebastian Ordyniak [b], Stefan Szeider [a]

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

A R T I C L E I N F O

Article history:
Received 22 March 2023
Received in revised form 12 January 2024
Accepted 15 January 2024
Available online 19 January 2024

Keywords:
SAT
Satisfiability
Backdoor
Elimination distance
Parameterized algorithms

A B S T R A C T

For several decades, much effort has been put into identifying classes of CNF formulas whose satisfiability can be decided in polynomial time. Classic results are the linear-time tractability of Horn formulas (Aspvall, Plass, and Tarjan, 1979) and Krom (i.e., 2CNF) formulas (Dowling and Gallier, 1984). Backdoors, introduced by Williams, Gomes and Selman (2003), gradually extend such a tractable class to all formulas of bounded distance to the class. Backdoor size provides a natural but rather crude distance measure between a formula and a tractable class. Backdoor depth, introduced by Mählmann, Siebertz, and Vigny (2021), is a more refined distance measure, which admits the utilization of different backdoor variables in parallel. We propose FPT approximation algorithms to compute backdoor depth into the classes Horn and Krom. This leads to a linear-time algorithm for deciding the satisfiability of formulas of bounded backdoor depth into these classes.

1. Introduction

Deciding the satisfiability of a propositional formula in conjunctive normal form (CnfSat) is one of the most important NP-complete problems [6,21]. Despite its theoretical intractability, heuristic algorithms work surprisingly fast on real-world CnfSat instances [10,12]. A common explanation for this discrepancy between theoretical hardness and practical feasibility is the presence of a certain "hidden structure" in realistic CnfSat instances [19]. There are various approaches to capturing the vague notion of a "hidden structure" with a mathematical concept. One widely studied approach is to consider the hidden structure in terms of decomposability. We can associate with each CNF formula a bipartite *incidence graph* with the formulas variables on one side and the formulas clauses on the other side, and edges between a variable and a clause if the former appears in the latter. For instance, CnfSat can be solved in quadratic time for classes of CNF formulas whose incidence graph has bounded treewidth [2,30].

A complementary approach proposed by Williams et al. [34] considers the hidden structure of a CnfSat instance in terms of a small number of key variables, called *backdoor variables*, that, when instantiated, move the instance into a polynomial-time solvable class. More precisely, a *backdoor*[1] of *size* k of a CNF formula F into a polynomial-time solvable class $\mathcal{C}$ is a set B of k variables such that for all partial assignments τ to B, the instantiated formula $F[\tau]$ belongs to $\mathcal{C}$. Here $F[\tau]$ denotes the CNF formula obtained from F by removing those variables set to false by τ and removing those clauses that are already satisfied by τ. In fact, CnfSat can be solved in linear time for any class of CNF formulas that admit backdoors

☆ Parts of this paper appeared in the Proceedings of the 30th Annual European Symposium on Algorithms (ESA 2022) [8].

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

[1] We focus only on strong backdoors, as weak backdoors only apply to satisfiable formulas.

https://doi.org/10.1016/j.jcss.2024.103520

of *bounded size* into the class of *Horn*, *dual Horn* or *Krom* (i.e., 2CNF) formulas. According to Schaefer's Theorem [32], these three classes are the largest nontrivial classes of CNF formulas defined in terms of a property of clauses, for which CNFSAT can be solved in polynomial time.

For a CNF formula F and class of formulas $\mathcal{C}$, the size d of a smallest backdoor of F into $\mathcal{C}$ conceptually measures a certain kind of *distance* that F has to $\mathcal{C}$: If $d = 0$, then F is already in $\mathcal{C}$, and otherwise d measures how many variables need to be assigned to "move" F into $\mathcal{C}$. While the *size* of a smallest backdoor is a fundamental but rather simple distance measure between F and $\mathcal{C}$, Mählmann, Siebertz, and Vigny [22] propose to instead consider the smallest *depth* over all backdoors as distance measure. For a formula F and a class $\mathcal{C}$, we recursively define this depth $\mathrm{depth}_{\mathcal{C}}(F)$ as

$$\mathrm{depth}_{\mathcal{C}}(F) := \begin{cases} 0 & \text{if } F \in \mathcal{C}, \\ 1 + \min\limits_{x \in var(F)} \max\limits_{\epsilon \in \{0,1\}} \mathrm{depth}_{\mathcal{C}}(F[x=\epsilon]) & \text{if } F \notin \mathcal{C} \text{ and } F \text{ is connected}, \\ \max\limits_{F' \in \mathrm{Conn}(F)} \mathrm{depth}_{\mathcal{C}}(F') & \text{otherwise.} \end{cases} \tag{1}$$

Here, $\mathrm{Conn}(F)$ denotes the set of connected components of the incidence graph of F; precise definitions are given in Section 2. We can certify $\mathrm{depth}_{\mathcal{C}}(F) \le k$ with a *component $\mathcal{C}$-backdoor tree* of depth $\le k$ which is a decision tree that reflects the choices made in the above recursive definition.

Backdoor depth is based on the observation that if an instance F decomposes into multiple connected components of $F[x=0]$ and $F[x=1]$, then each component can be treated independently. This way, one is allowed to use in total an unbounded number of backdoor variables. As long as the depth of the component $\mathcal{C}$-backdoor tree is bounded, one can still utilize the backdoor variables to solve the instance efficiently. In the context of graphs, similar ideas are used in the study of tree-depth [24,25] and elimination distance [5,11]. Bounded backdoor size implies bounded backdoor depth, but there are classes of formulas of unbounded backdoor size but bounded backdoor depth.

The challenging algorithmic problem $\mathcal{C}$-BACKDOOR DEPTH is to find for a fixed base class $\mathcal{C}$ and a given formula F, a component $\mathcal{C}$-backdoor tree of F of depth $\le k$. Mählmann et al. [22] gave an FPT-approximation algorithm for this problem, with k as the parameter) where $\mathcal{C}$ is the trivial class NULL for formulas without variables. A component NULL-backdoor tree must instantiate all variables of F. In an accompanying paper [9] we have extended this result to obtain fixed-parameter tractability of the constraint satisfaction problem (CSP) parameterized by the backdoor depth into CSP defined by any finite, tractable, semi-conservative constraint language.

1.1. New results

In this paper, we give the first positive algorithmic results for backdoor depth into nontrivial classes. A minimization problem admits a *standard fixed-parameter tractable approximation (FPT-approximation)* [23] if for an instance of size n and parameter k there is an *FPT-algorithm*, i.e., an algorithm running in time $f(k)n^{\mathcal{O}(1)}$, that either outputs a solution of size at most $g(k)$ or outputs that the instance has no solution of size at most k, for some computable functions f and g; $g(k)$ is also referred to as the performance ratio of the algorithm.

Main Result 1 *(Theorem 19).* $\mathcal{C}$-BACKDOOR DEPTH admits an FPT-approximation algorithm if $\mathcal{C}$ is any of the Schaefer classes Horn, dual Horn, or Krom.

Since our FPT algorithms have linear running time for fixed backdoor depth k, we obtain the following corollary:

Main Result 2 *(Corollary 20).* CNFSAT can be solved in linear time for formulas of bounded backdoor depth into the Schaefer classes Horn, dual Horn, and Krom.

Backdoor depth is a powerful parameter that is able to capture and exploit structure in CNFSAT instances that is not captured by any other known method. In Fig. 1, we give a brief comparison between backdoor depth and other well-known parameters. Definitions of these parameters and separation proofs are given in Section 8.

1.2. Approach and techniques

A common approach to construct backdoors is to compute in parallel both an upper bound and a lower bound. The upper bounds are obtained by constructing the backdoor itself, and lower bounds are usually obtained in the form of so-called *obstructions*. These are parts of an instance that are proven to be "far away" from the base class. In the context of backdoor depth, this role is fulfilled by so-called *obstruction trees*, introduced in the pioneering work of Mählmann et al. [22]. In this paper, we also use obstruction trees, but construct them with using quite different techniques. The results of [22] are limited to the trivial base class NULL, where the obstructions are rather simple because they can contain only boundedly many variables. Our central technical contribution is overcoming this limitation by introducing *separator obstructions*.

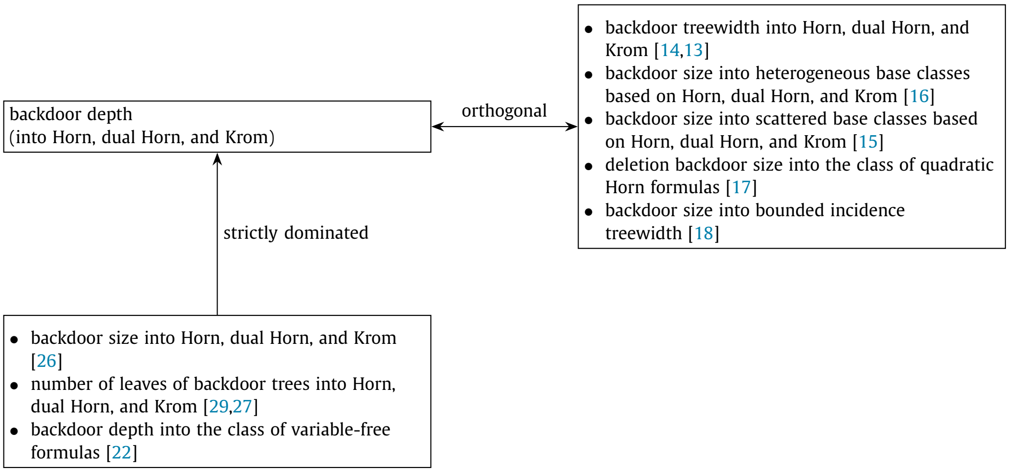

Fig. 1. We list some well-known parameters which render CnfSat fixed-parameter tractable (the list is not complete but covers some of the most essential parameters). For all these parameters, there exist CNF formulas with constant backdoor depth (into *Horn*, *dual Horn*, and *Krom*) but where the other parameter is arbitrarily large; if there also exist formulas where the converse is true, we call the respective parameter "orthogonal", otherwise we call it "strictly dominated." We give definitions and separation proofs in Section 8.

Separator obstructions allow us to algorithmically work with obstruction trees containing an unbounded number of variables, an apparent requirement for dealing with nontrivial base classes different form Null. In the context of backdoor depth, it is crucial that an existing obstruction is disjoint from all potential future obstructions, so they can later be joined safely into a new obstruction of increased depth. Mählmann et al. [22] ensure this by placing the whole current obstruction tree into the backdoor—an approach that only works for the most trivial base class because only there the obstructions have a bounded number of variables. As one considers more and more general base classes, one needs to construct more and more complex obstructions to prove lower bounds. For example, as instances of the base class no longer have bounded diameter (of the incidence graph of the formula) or bounded clause length, neither have the obstructions one needs to consider. Such obstructions become increasingly hard to separate. Our separator obstructions can separate obstruction trees containing an unbounded number of variables from all potential future obstruction trees. We obtain backdoors of bounded depth by combining the strengths of separator obstructions and obstruction trees. We further introduce a *game-theoretic framework* to reason about backdoors of bounded depth. With this notion, we can compute winning strategies instead of explicitly constructing backdoors, greatly simplifying the presentation of our algorithms.

2. Preliminaries

Satisfiability. A *literal* is a propositional variable x or a negated variable $\neg x$. A *clause* is a finite set of literals that does not contain a complementary pair x and $\neg x$ of literals. A propositional formula in conjunctive normal form, or *CNF formula* for short, is a set of clauses. We denote by $\mathcal{CNF}$ the class of all CNF formulas. Let $F \in \mathcal{CNF}$ and $c \in F$. We denote by $var(c)$ the set of all variables occurring in c, i.e., $var(c) = \{\, x \mid x \in c \vee \neg x \in c \,\}$ and we set $var(F) = \bigcup_{c \in F} var(c)$. For a set of literals L, we denote by $\overline{L} = \{\, \neg l \mid l \in L \,\}$, the set of complementary literals of the literals in L. The *size* of a CNF formula F is $\|F\| = \sum_{c \in F} |c|$.

Let $\tau : X \to \{0,1\}$ be an assignment of some set X of propositional variables. If $X = \{x\}$ and $\tau(x) = \epsilon$, we will sometimes also denote the assignment τ by $x = \epsilon$ for brevity. We denote by $\text{true}(\tau)$ ($\text{false}(\tau)$) the set of all literals satisfied (falsified) by τ, i.e., $\text{true}(\tau) = \{\, x \in X \mid \tau(x) = 1 \,\} \cup \{\, \neg x \in \overline{X} \mid \tau(x) = 0 \,\}$ ($\text{false}(\tau) = \overline{\text{true}(\tau)}$). We denote by $F[\tau]$ the formula obtained from F after removing all clauses that are satisfied by τ and from the remaining clauses removing all literals that are falsified by τ, i.e., $F[\tau] = \{\, c \setminus \text{false}(\tau) \mid c \in F \text{ and } c \cap \text{true}(\tau) = \emptyset \,\}$. We say that an assignment satisfies F if $F[\tau] = \emptyset$. We say that F is *satisfiable* if there is some assignment $\tau : var(F) \to \{0,1\}$ that satisfies F, otherwise F is *unsatisfiable*. CnfSat denotes the propositional satisfiability problem, which takes a CNF formula as input, and asks whether the formula is satisfiable.

The *incidence graph* of a CNF formula F is the bipartite graph G_F whose vertices are the variables and clauses of F, and where a variable x and a clause c are adjacent if and only if $x \in var(c)$. We identify a subgraph G' of the incidence graph G_F with the formula F' consisting of all the clauses of F that are in G', each restricted to the adjacent variables in G'. With slight abuse of notation, we define $var(F')$ to be the variables occurring in G'. Via incidence graphs, graph theoretic concepts directly translate to CNF formulas. For instance, we say that F is *connected* if G_F is connected, and F' is a *connected component* of F if F' is a maximal connected subset of F. $\text{Conn}(F)$ denotes the set of connected components of F. Moreover, the *primal graph* of a CNF formula F has the vertex set $var(F)$ and an edge between vertices $x, z \in var(F)$ if and only if there exists a clause $c \in F$ containing both x and y.

Base classes. Let $\alpha \subseteq \{+, -\}$ with $\alpha \neq \emptyset$, let $F \in \mathcal{CNF}$ and $c \in F$. We say that a literal l is an *α-literal* if is a positive literal and $+ \in \alpha$ or it is a negative literal and $- \in \alpha$. We say that a variable v *α-occurs* in a clause c, if v or $\neg v$ is an α-literal that is contained in c. We denote by $var_\alpha(c)$ the set of variables that α-occur in c. For $\alpha \subseteq \{+, -\}$ with $\alpha \neq \emptyset$ and $s \in \mathbb{N}$, let $\mathcal{C}_{\alpha,s}$ be the class of all CNF formulas F such that every clause of F contains at most s α-literals. For $\mathcal{C} \subseteq \mathcal{CNF}$, we say that a clause c is *$\mathcal{C}$-good* if $\{c\} \in \mathcal{C}$. Otherwise, c is *$\mathcal{C}$-bad*. Let τ be any (partial) assignment of the variables of F. We will frequently make use of the fact that $\mathcal{C}_{\alpha,s}$ is *closed under assignments*, i.e., if $F \in \mathcal{C}_{\alpha,s}$, then also $F[\tau] \in \mathcal{C}_{\alpha,s}$. Therefore, whenever a clause $c \in F$ is $\mathcal{C}_{\alpha,s}$-good it will remain $\mathcal{C}_{\alpha,s}$-good in $F[\tau]$ and conversely whenever a clause is $\mathcal{C}_{\alpha,s}$-bad in $F[\tau]$ it is also $\mathcal{C}_{\alpha,s}$-bad in F.

The classes $\mathcal{C}_{\alpha,s}$ capture (according to Schaefer's Dichotomy Theorem [32]) the largest syntactic classes of CNF formulas for which the satisfiability problem can be solved in polynomial time: The class $\mathcal{C}_{\{+\},1} = \textsc{Horn}$ of *Horn formulas*, the class of $\mathcal{C}_{\{-\},1} = \textsc{dHorn}$ of *dual Horn formulas*, and the class $\mathcal{C}_{\{+,-\},2} = \textsc{Krom}$ of *Krom* (or *2CNF*) *formulas*. Note also that the class $\textsc{Null}$ of formulas containing no variables considered by Mählmann et al. [22] is equal to $\mathcal{C}_{\{+,-\},0}$. We follow Williams et al. [34] and focus on classes that are closed under assignments and therefore we do not consider the classes of 0/1-valid and affine formulas.

Note that every class $\mathcal{C}_{\alpha,s}$ (and therefore also the classes of Krom, Horn, and dual Horn formulas) is trivially *linear-time recognizable*, i.e., membership in the class can be tested in linear-time. We say that a class $\mathcal{C}$ of formulas is *tractable* or *linear-time tractable*, if $\textsc{CnfSat}$ restricted to formulas in $\mathcal{C}$ can be solved in polynomial-time or linear-time, respectively. The classes $\textsc{Horn}, \textsc{dHorn}, \textsc{Krom}$ are linear-time tractable [1,7].

3. Backdoor depth

A *binary decision tree* is a rooted binary tree T. Every inner node t of T is assigned a propositional variable, denoted by $var(t)$, and has exactly one left and one right child, which corresponds 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 $var(T)$ the set of all variables assigned to any node of T. Finally, we associate with each node t of T, the truth assignment τ_t that is defined on all the variables $var(P) \setminus \{var(t)\}$ occurring on the unique path P from the root of T to t such that $\tau_t(v) = 0$ ($\tau_t(v) = 1$) if $v \in var(P) \setminus \{var(t)\}$ and P contains the left child (right child) of the node t' on P with $var(t') = v$. Let $\mathcal{C}$ be a base class, F be a CNF formula, and T be a decision tree with $var(T) \subseteq var(F)$. Then T is a *$\mathcal{C}$-backdoor tree of F* if $F[\tau_t] \in \mathcal{C}$ for every leaf t of T [29].

Component backdoor trees generalize backdoor trees as considered by Samer and Szeider [29] by allowing an additional node type, *component nodes*, where the current instance is split into connected components. More precisely, let $\mathcal{C}$ be a base class and F be a CNF formula. A *component $\mathcal{C}$-backdoor tree for F* is a pair (T, φ), where T is a rooted tree and φ is a mapping that assigns each node t a CNF formula $\varphi(t)$ such that the following conditions are satisfied:

1. For the root r of T, we have $\varphi(r) = F$.
2. For each leaf ℓ of T, we have $\varphi(\ell) \in \mathcal{C}$.
3. For each non-leaf t of T, there are two possibilities:

   (a) t has exactly two children t_0 and t_1, where for some variable $x \in var(\varphi(t))$ we have $\varphi(t_i) = \varphi(t)[x = i]$; in this case we call t a *variable node*.
   (b) $\mathrm{Conn}(\varphi(t)) = \{F_1, \ldots, F_k\}$ for $k \geq 2$ and t has exactly k children $t_1, \ldots, t_k$ with $\varphi(t_i) = F_i$; in this case we call t a *component node*.

For an example see Fig. 2. Thus, a backdoor tree is just a component backdoor tree without component nodes. The *depth* of a $\mathcal{C}$-backdoor is the largest number of variable nodes on any root-to-leaf path. The *$\mathcal{C}$-backdoor depth* $\mathrm{depth}_{\mathcal{C}}(F)$ of a formula F into a base class $\mathcal{C}$ is the smallest depth over all component $\mathcal{C}$-backdoor trees of F. Alternatively, we can define $\mathcal{C}$-backdoor depth recursively as in equation (1) from the introduction. For a component backdoor tree (T, φ) let $var(T, \varphi)$ be the set of all variables x such that some variable node t of T branches on x. We observe that one can use component $\mathcal{C}$-backdoor trees to decide the satisfiability of a formula.

Lemma 1. *Let $\mathcal{C} \subseteq \mathcal{CNF}$ be tractable, let $F \in \mathcal{CNF}$, and let (T, φ) be a component $\mathcal{C}$-backdoor tree of F of depth d. Then, we can decide the satisfiability of F in time $(2^d \|F\|)^{\mathcal{O}(1)}$. Moreover, if $\mathcal{C}$ is linear-time tractable, then the same can be done in time $\mathcal{O}(2^d \|F\|)$.*

Proof. Let $m = \|F\|$. We start by showing that $\sum_{\ell \in L(T)} \|\varphi(\ell)\|| \leq 2^d m$, where $L(T)$ denotes the set of leaves of T, using induction on d and m. The statement holds if $d = 0$ or $m \leq 1$. We show that it also holds for larger d and m. If the root is a variable node, then it has two children c_0, c_1, and the subtree rooted at any of these children represents a component $\mathcal{C}$-backdoor tree for the CNF formula $\varphi(c_i)$ of depth $d-1$. Therefore, by the induction hypothesis, we obtain that $s_i = \sum_{\ell \in L(T_i)} \|\varphi(\ell)\| \leq 2^{d-1} m$, for the subtree T_i rooted at c_i, $i \in \{0, 1\}$. Consequently, $\sum_{\ell \in L(T)} \|\varphi(\ell)\| = s_0 + s_1 \leq 2 \cdot 2^{d-1} m = 2^d m$, as required. If, on the other hand, the root is a component node, then its children, say $c_1, \ldots, c_k$, are labeled with CNF formulas of sizes $m_1 + \cdots + m_k = m$. Therefore, for every subtree T_i of T rooted at c_i, we have that T_i is a component

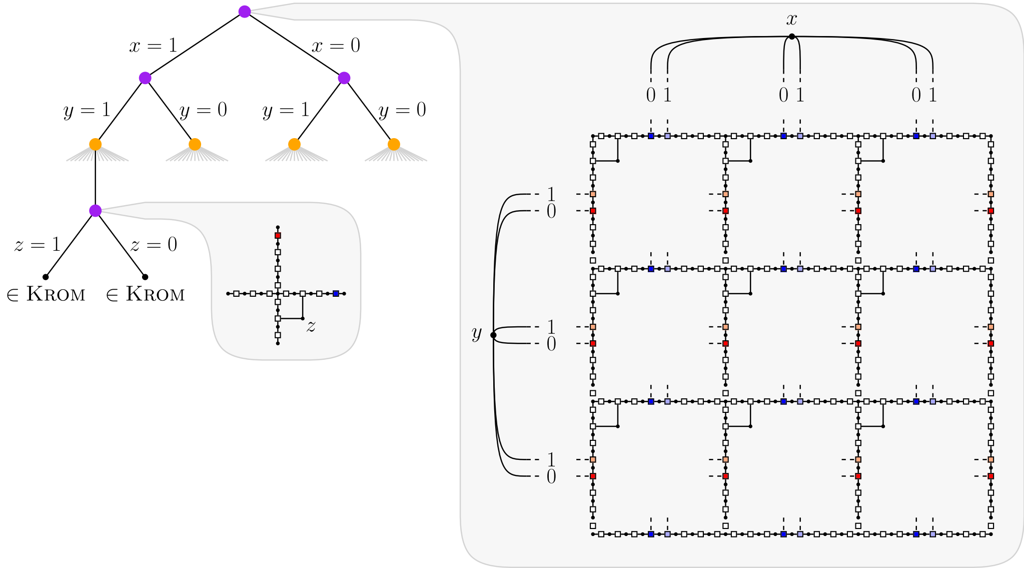

Fig. 2. A component $\mathcal{C}$-backdoor tree of depth four, for $\mathcal{C} = \text{Krom}$. Variable nodes are purple, component nodes are orange, and leafs are black. The gray boxes show the associated CNF formulas. Within each CNF formula, boxes depict clauses, circles depict variables and edges depict containment. The highlighted variable x is contained negatively in all dark blue clauses and positively in all light blue clauses. Similarly, y is contained negatively in all dark red clauses and positively in all light red clauses. After branching over x and y, the formula decomposes into 16 components at the third level of the tree. After branching over the depicted variable z, all clauses contain at most two variables. Hence, the leaves of the tree correspond to Krom formulas. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

$\mathcal{C}$-backdoor tree of depth d for $\varphi(c_i)$, which using the induction hypothesis implies that $\sum_{\ell \in L(T_i)} \|\varphi(\ell)\| \leq 2^d m_i$. Hence, we obtain $\sum_{\ell \in L(T)} \|\varphi(\ell)\| \leq 2^d m$ in total.

To decide the satisfiability of F, we first decide the satisfiability of all the formulas associated with the leaves of T. Because, as shown above, their total size is at most $2^d m$, this can be achieved in time $(2^d m)^{\mathcal{O}(1)}$ if CnfSat restricted to formulas in $\mathcal{C}$ is polynomial-time solvable and in time $\mathcal{O}(2^d m)$ if CnfSat restricted to formulas in $\mathcal{C}$ is linear-time solvable. Let us call a leaf true/false if it is labeled by a satisfiable/unsatisfiable CNF formula, respectively. We now propagate the truth values upwards to the root, considering a component node as the logical *and* of its children, and a variable node as the logical *or* of its children. F is satisfiable if and only if the root of T is true. We can carry out the propagation in time linear in the number of nodes of T, which is linear in the number of leaves of T, which is at most $2^d m$. □

In Section 8, we will need the following simple observation. Let $\mathcal{C} \subseteq \mathcal{CNF}$ and $F \in \mathcal{CNF}$. A (strong) $\mathcal{C}$-*backdoor* of F is a set $B \subseteq var(F)$ such that $F[\tau] \in \mathcal{C}$ for each $\tau : B \to \{0, 1\}$. Assume $\mathcal{C}$ is closed under partial assignments (which is the case for many natural base classes and the classes $\mathcal{C}_{\alpha,s}$) and (T, φ) a component $\mathcal{C}$-backdoor tree of F. We argue that $var(T, \varphi)$ is a $\mathcal{C}$-backdoor of F: Fix a partial assignment τ to $var(T, \varphi)$. Traversing T downwards from the root according to τ yields a leaf t and sub-assignment τ_t of τ with $F[\tau_t] \in \mathcal{C}$. Since $\mathcal{C}$ is closed under partial assignments, we also have $F[\tau] \in \mathcal{C}$.

4. Technical overview

We present all our algorithms in this work within a game-theoretic framework. This framework builds upon the following equivalent formulation of backdoor depth using splitter games. Similar games can be used to describe treedepth and other graph classes [20].

Definition 2. Let $\mathcal{C} \subseteq \mathcal{CNF}$ and $F \in \mathcal{CNF}$. We denote by Game$(F, \mathcal{C})$ the so-called $\mathcal{C}$-*backdoor depth game on* F. The game is played between two players, the *connector* and the *splitter*. The *positions* of the game are CNF formulas. At first, the connector chooses a connected component of F to be the *starting position* of the game. The game is over once a position in the base class $\mathcal{C}$ is reached. We call these positions the *winning positions* (of the splitter). In each *round* the game progresses from a *current position* J to a *next position* as follows:

- The splitter chooses a variable $v \in var(J)$.
- The connector chooses an assignment $\tau : \{v\} \to \{0, 1\}$ and a connected component J' of $J[\tau]$. The *next position* is J'.

In the (unusual) case that a position J contains no variables anymore but J is still not in $\mathcal{C}$, the splitter looses. For a position J, we denote by τ_J the assignment of all variables assigned up to position J.

The following observation follows easily from the definition of the game and the fact that the (strategy) tree obtained by playing all possible plays of the connector against a given d-round winning strategy for the splitter forms a component backdoor tree of depth d, and vice versa. In particular, the splitter choosing a variable v at position J corresponds to a variable node and the subsequent choice of the connector for an assignment τ of v and a component of $J[\tau]$ corresponds to a component node (and a subsequent variable or leaf node) in a component backdoor tree.

Observation 3. The splitter has a strategy for the game $\textsc{Game}(F, \mathcal{C})$ to reach within at most d rounds a winning position if and only if F has $\mathcal{C}$-backdoor depth at most d.

Using backdoor depth games, we no longer have to explicitly construct a backdoor. Instead, we present so-called *splitter-algorithms* that play the backdoor depth game from the perspective of the splitter. These algorithms will have some auxiliary internal state that they modify with each move. Formally, a splitter-algorithm for the $\mathcal{C}$-backdoor depth game to a base class $\mathcal{C}$ is a procedure that

- gets as input a (non-winning) position J of the game, together with an internal state
- and returns a valid move for the splitter at position J, together with an updated internal state.

It can be understood as a function $(\text{position}, \text{state}) \to (\text{new position}, \text{new state})$. We will usually use the internal state to hold an obstruction that the splitter will periodically increase in size. Assume we have a game $\textsc{Game}(F, \mathcal{C})$ and some additional input S. For a given strategy of the connector, the splitter-algorithm plays the game as one would expect: In the beginning, the internal state is initialized with S (if no additional input is given, the state is initialized empty). Whenever the splitter should make its next move, the splitter-algorithm is queried using the current position and internal state, and afterwards the internal state is updated accordingly.

Definition 4. We say a splitter-algorithm *implements a strategy to reach for a game* $\textsc{Game}(F, \mathcal{C})$ *and input S within at most d rounds a position and internal state with some property* if initializing the internal state with S and then playing $\textsc{Game}(F, \mathcal{C})$ according to the splitter-algorithm leads—no matter what strategy the connector is using—after at most d rounds to a position and internal state with said property.

The following observation converts splitter-algorithms into algorithms for bounded depth backdoors. It builds component backdoor trees by trying all moves of the connector.

Lemma 5. *Let $\mathcal{C} \subseteq \mathcal{CNF}$ and $f_{\mathcal{C}} : \mathbb{N} \to \mathbb{N}$. Assume there exists a splitter-algorithm that implements a strategy to reach in each play in the game $\textsc{Game}(F, \mathcal{C})$ and non-negative integer d within at most $f_{\mathcal{C}}(d)$ rounds either:*

i) a winning position, or
ii) (an internal state representing) a proof that the $\mathcal{C}$-backdoor depth of F is at least d.

Further assume this splitter-algorithm always takes at most $\mathcal{O}(\|F\|)$ time to compute its next move. Then there is an algorithm that, given F and d, in time at most $3^{f_{\mathcal{C}}(d)}\mathcal{O}(\|F\|)$ either:

i) returns a component $\mathcal{C}$-backdoor tree of depth at most $f_{\mathcal{C}}(d)$, or
ii) concludes that the $\mathcal{C}$-backdoor depth of F is at least d.

Proof. We compute a component $\mathcal{C}$-backdoor tree of depth at most $f_{\mathcal{C}}(d)$ by starting at the root and then iteratively expanding the leaves, using the splitter-algorithm to compute the next variable to branch over. For each position we reach, we store the internal state of the splitter-algorithm in a look-up table, indexed by the position. This way, we can easily build the component $\mathcal{C}$-backdoor tree, e.g., in a depth-first way. If we encounter at any time an internal state representing a proof that the $\mathcal{C}$-backdoor depth of F is at least d, we can abort. If this is not the case, then we are guaranteed that every leaf represents a winning position and therefore an instance in $\mathcal{C}$. We have therefore found a component $\mathcal{C}$-backdoor tree of depth at most $f_{\mathcal{C}}(d)$.

Without loss of generality, we can assume that F is connected. We need to expand the root node F into a tree of depth $f_{\mathcal{C}}(d)$. We show by induction on $i \geq 0$ that we can expand a node J into a tree of depth i in time at most $3^i c\|J\|$ for some constant c, thereby proving the lemma. As a base case, it takes no time to expand a node zero times. To expand a node J $i > 0$ times, we run the splitter algorithm in time $c\|J\|$ to get the next variable and create a subtree for both assignments of this variable. For each of the two assignments, the instance splits into some components $J_1, \ldots, J_k$ with $\|J_1\| + \cdots + \|J_k\| \leq \|J\|$ (for one assignment) and components $J'_1, \ldots, J'_{k'}$ with $\|J'_1\| + \cdots + \|J'_{k'}\| \leq \|J\|$ (for the other assignment). By induction, we can expand each such component J_j $i-1$ times in time $3^{i-1}c\|J_j\|$. This yields a total run time of at most

$$c\|J\| + \sum_j 3^{i-1}c\|J_j\| + \sum_j 3^{i-1}c\|J'_j\| \le c\|J\| + 2\cdot 3^{i-1}c\|J\| \le 3^i c\|J\|. \quad \square$$

For the sake of readability, we may present splitter-algorithms as continuously running algorithms that periodically output moves (via some output channel) and always immediately as a reply get the next move of the connector (via some input channel). Such an algorithm can easily be converted into a procedure that gets as input a position and internal state and outputs a move and a modified internal state: The internal state encodes the whole state of the computation, (e.g., the current state of a Turing machine together with the contents of the tape and the position of the head). Whenever the procedure is called, it "unfreezes" this state, performs the computation until it reaches its next move and then "freezes" and returns its state together with the move.

Our main result is an approximation algorithm (Theorem 19) that either concludes that there is no backdoor of depth d, or computes a component backdoor tree of depth at most $2^{2^{\mathcal{O}(d)}}$. By Lemma 5, this is equivalent to a splitter-algorithm that plays for $2^{2^{\mathcal{O}(d)}}$ rounds to either reach a winning position or a proof that the backdoor depth is larger than d.

Following the approach of Mählmann et al. [22], our proofs of high backdoor depth come in the form of so-called *obstruction trees*. These are trees in the incidence graph of a CNF formula. Their node set therefore consists of both variables and clauses. Obstruction trees of depth d describe parts of an instance for which the splitter needs more than d rounds to win the backdoor depth game. For depth zero, we simply take a single (bad) clause that is not allowed by the base class. Roughly speaking, an obstruction tree of depth $d > 0$ is built from two "separated" obstruction trees T_1, T_2 of depth $d-1$ that are connected by a path. As we will see later in Section 6, these conditions ensure for any variable v that the splitter may play, that there is a response of the connector (i.e., an assignment of v and a component) in which either T_1 or T_2 is whole. From this position, the splitter needs by induction still more than $d-1$ additional rounds to win the game, and thus needs more than d rounds to win the game as a whole.

Definition 6. Let $F \in \mathcal{CNF}$ and $\mathcal{C} = \mathcal{C}_{\alpha,s}$ with $\alpha \subseteq \{+,-\}$, $\alpha \neq \emptyset$, and $s \in \mathbb{N}$. We inductively define $\mathcal{C}$-obstruction trees T for F of increasing depth.

- Let c be a $\mathcal{C}$-bad clause of F. The set $T = \{c\}$ is a *$\mathcal{C}$-obstruction tree in F of depth 0*.
- Let T_1 be a $\mathcal{C}$-obstruction tree of depth i in F. Let β be a partial assignment of the variables in F. Let T_2 be an obstruction tree of depth i in $F[\beta]$ such that no variable $v \in var(F[\beta])$ that is contained in a clause of T_2 is contained in T_1 or α-occurs in a clause of T_1. Let further P be (a CNF formula representing) a path that connects T_1 and T_2 in F. Then $T = T_1 \cup T_2 \cup var(P) \cup P$ is a *$\mathcal{C}$-obstruction tree in F of depth $i+1$*.

We prove the following central lemma in Section 6.

Lemma 7. *Let $F \in \mathcal{CNF}$ and $\mathcal{C} = \mathcal{C}_{\alpha,s}$ with $\alpha \subseteq \{+,-\}$, $\alpha \neq \emptyset$, and $s \in \mathbb{N}$. If there is a $\mathcal{C}$-obstruction tree of depth d in F, then the $\mathcal{C}$-backdoor depth of F is larger than d.*

Our splitter-algorithm will construct obstruction trees of increasing depth by a recursive procedure (Lemma 18) that we outline now. We say a splitter-algorithm satisfies *Property i* if it reaches in each game $\textsc{Game}(F,\mathcal{C})$ within $g_{\mathcal{C}}(i,d)$ rounds (for some function $g_{\mathcal{C}}(i,d)$) either

1) a winning position, or
2) a position J and a $\mathcal{C}$-obstruction tree T of depth i in F such that no variable in $var(J)$ occurs in T or α-occurs in a clause of T, or
3) a proof that the $\mathcal{C}$-backdoor depth of F is at least d.

If we have a splitter algorithm satisfying Property $d+1$ then our main result, the approximation algorithm for backdoor depth, directly follows from Lemma 7 and Lemma 5. Assume we have a strategy satisfying Property $i-1$, let us describe how to use it to satisfy Property i. If at any point we reach a winning position or a proof that the $\mathcal{C}$-backdoor depth of F is at least d, then we are done. Let us assume that this does not happen, so we can focus on the much more interesting case 2).

We use Property $i-1$ to construct a first tree T_1 of depth $i-1$, and reach a position J_1. We use it again, starting at position J_1 to construct a second tree T_2 of depth $i-1$ that is completely contained in position J_1. Since in the beginning the connector selected a connected component, T_1 and T_2 are in the same component of F and we can find a path P connecting them. Let β be the assignment that assigns all the variables the splitter has chosen until reaching position J_1. Then T_2 is an obstruction tree not only in J_1 but also in $F[\beta]$. In order to join both trees together into an obstruction of depth i, we have to show, according to Definition 6, that no variable $v \in var(F[\beta])$ that is contained in a clause of T_2 is contained in T_1 or α-occurs in a clause of T_1. Since no variable in $var(J_1)$ occurs in T_1 or α-occurs in a clause of T_1 (Property $i-1$), and T_2 was built only from J_1, this is the case. The trees T_1 and T_2 are "separated" and can be safely joined into a new obstruction tree T of depth i (see also Fig. 9 on page 17 and the proof of Lemma 18 for details).

The last thing we need to ensure is that we reach a position J such that no variable in $var(J)$ occurs in T or α-occurs in a clause of T. This then guarantees that T is "separated" from all future obstruction trees that we may want to join it with to satisfy Property $i+1$, $i+2$ and so forth. This is the major difficulty and main technical contribution of this paper.

It is important to note here that the exact notion of "separation" between obstruction trees plays a crucial role for our approach and is one of the main differences to Mählmann et al. [22]. Mählmann et al. solve the separation problem in a "brute-force" manner: If we translate their approach to the language of splitter-algorithms, then the splitter simply selects all variables that occur in a clause of T. For their base class—the class Null of formulas without variables—there are at most $2^{\mathcal{O}(d)}$ variables that occur in an obstruction tree of depth d. Thus, in only $2^{\mathcal{O}(d)}$ rounds, the splitter can select all of them, fulfilling the separation property. This completes the proof for the base class Null.

However, already for backdoor depth to Krom, this approach cannot work since instances in the base class have obstruction trees with arbitrarily many clauses. Moreover, the situation becomes even more difficult for backdoors to Horn, since additionally clauses are allowed to contain arbitrary many literals. Mählmann et al. acknowledge this as a central problem and ask for an alternative approach to the separation problem that works for more general base classes.

5. Separator obstructions

The main technical contribution of this work is a separation technique that works for the base classes $\mathcal{C}=\mathcal{C}_{\alpha,s}$. The separation technique is based on a novel form of obstruction, which we call *separator obstruction*. Obstruction trees are made up of paths, therefore, it is sufficient to separate each new path P that is added to an obstruction. Note that P can be arbitrarily long and every clause on P can have arbitrary many variables and therefore the splitter cannot simply select all variables in (clauses of) P. Instead, given such a path P that we want to separate, we will use separator obstructions to develop a splitter-algorithm (Lemma 16) that reaches in each game $\text{Game}(F,\mathcal{C})$ within a bounded number of rounds either

1) a winning position, or
2) a position J such that no variable in $var(J)$ occurs in P or α-occurs in a clause of P, or
3) a proof that the $\mathcal{C}$-backdoor depth of F is at least d.

Informally, a separator obstruction is a sequence $\langle P_1,\dots,P_\ell\rangle$ of paths that together form a tree T_ℓ, in combination with an assignment τ of certain *important* variables occurring in T_ℓ. The variables of τ correspond to the variables chosen by the splitter-algorithm and the assignment τ corresponds to the assignment chosen by the connector. Each path P_i adds at least one $\mathcal{C}$-bad clause b_i to the separator obstruction, which is an important prerequisite to increase the backdoor depth by growing the obstruction. Moreover, by choosing the important variables and the paths carefully, we ensure that for every *outside* variable, i.e., any variable that is not an important variable assigned by τ, there is an assignment and a component (which can be chosen by the connector) that leaves a large enough part of the separator obstruction intact. Thus, if a separator obstruction is sufficiently large, the connector can play such that even after d rounds a non-empty part of the separator obstruction is still intact. This means a large separator obstruction is a proof that the backdoor depth is larger than d.

To illustrate the growth of a separator obstruction (and motivate its definition) suppose that our splitter-algorithm is at position J of the game $\text{Game}(F,\mathcal{C})$ and has already built a separator obstruction $X=\langle\langle P_1,\dots,P_i\rangle,\tau\rangle$ (with corresponding tree T_i) containing $\mathcal{C}$-bad clauses $b_1,\dots,b_i$; note that τ is compatible with τ_J (i.e., τ and τ_J agree on the common assigned variables) and we can assume without loss of generality that J is connected. If J is already a winning position, then Property i is satisfied. Therefore, J has to contain a $\mathcal{C}$-bad clause. Note that if J does not contain a variable that is either in T_i or α-occurs in a clause of T_i, then J satisfies 2) of Property i and we are done. Otherwise, let Y be the set of all such variables in J and let b_{i+1} be a $\mathcal{C}$-bad clause in J that is closest to any variable in Y. Note that it can happen that b_{i+1} is in T_i in which case, we let P_{i+1} be the path that only contains b_{i+1}. Otherwise, let P be a shortest path from b_{i+1} to Y in J and let $y\in Y$ be the endpoint of P in Y. Let P_{i+1} be the path that is equal to P if $y\in T_i$ and otherwise P_{i+1} is obtained from P_i after adding an edge from y to a clause c in T_i such that y α-occurs in c. Then, we extend our separator obstruction X by attaching the path P_{i+1} to T_i (and obtain the tree T_{i+1}). Our next order of business is to choose a bounded number of important variables occurring on P_{i+1} that we will add to X. Those variables need to be chosen such that no outside variable can destroy too much of the separator obstruction. Apart from destroying the paths of the separator obstruction, we also need to avoid that assigning any outside variable makes too many of the $\mathcal{C}$-bad clauses $b_1,\dots,b_{i+1}$ $\mathcal{C}$-good. Therefore, a natural choice would be to add all variables of b_{i+1} to X, i.e., to mark those variables as important. Unfortunately, this is not possible since b_{i+1} can contain arbitrarily many literals. Instead, we will only add the variables of b_{i+1} to X that α-occur in b_{i+1}. By the following lemma, the number of those variables is bounded.

Lemma 8. *Let $F\in\mathcal{CNF}$ and $\mathcal{C}=\mathcal{C}_{\alpha,s}$ with $\alpha\subseteq\{+,-\}$, $\alpha\neq\emptyset$, and $s\in\mathbb{N}$. If F has $\mathcal{C}$-backdoor depth at most some integer d, then every clause of F contains at most $d+s$ α-literals.*

Proof. As stated in the preliminaries, we can assume that every variable occurs at most once in every clause. Suppose that F contains a clause c containing more than $d+s$ α-literals. If the splitter chooses a variable from c, the connector will

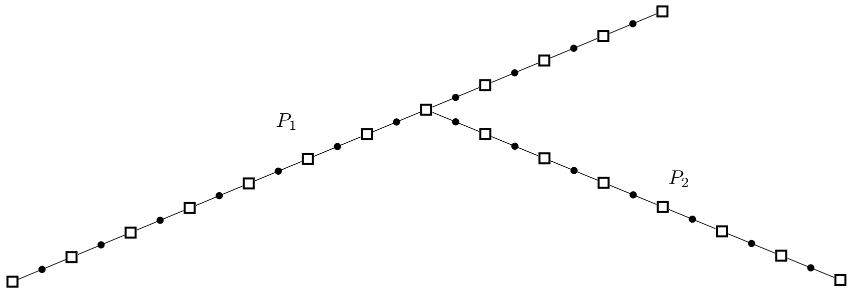

Fig. 3. A step-by-step example illustrating Definition 9. The recursive definition starts with a tree T_{i-1}, in this case with $i = 3$.

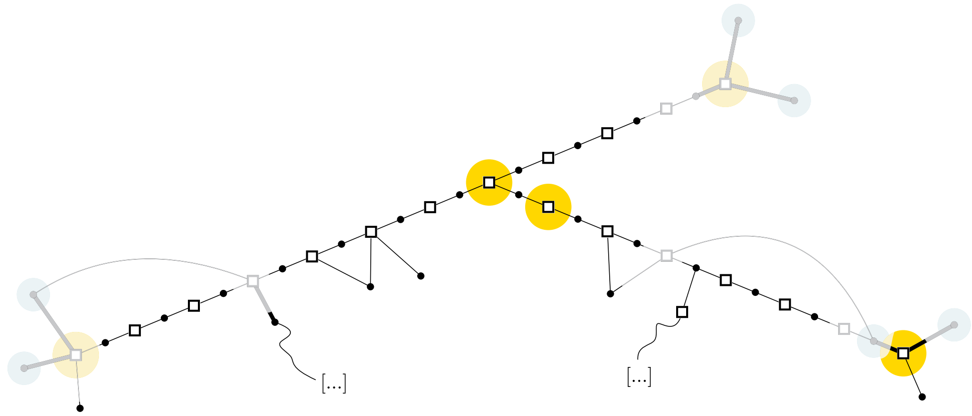

Fig. 4. The assignment τ_{i-1} assigns the variables in V_{i-1}. We illustrate the reduced formula $F[\tau_{i-1}]$ in our example. Omitted parts are indicated by "[...]". We highlight in orange the clauses in B_{i-1} and in light blue the variables in V_{i-1}. The grayed out variables and clauses are present in F but no longer in $F[\tau_{i-1}]$. This means grayed out variables are assigned by τ_{i-1} and grayed out clauses were satisfied by τ_{i-1}. Bold edges indicate that a variable is α-contained in a clause.

assign it to zero if it occurs positively in c and to one otherwise. Thus, the connector can play such that after d rounds, c still has more than s α-literals and therefore still is $\mathcal{C}$-bad. By Observation 3, F has backdoor depth larger than d. $\square$

While this still allows for outside variables to occur in many of the $\mathcal{C}$-bad clauses $b_1, \ldots, b_{i+1}$, it already ensures that no outside variable can α-occur in any of these clauses. This helps us, since when $|\alpha| = 1$ (i.e., the only case where α-occurs means something different then just occurs), it provides us with an assignment of any such outside variable that the connector can play without making the $\mathcal{C}$-bad clauses in which it occurs $\mathcal{C}$-good. For instance, if $\alpha = \{+\}$, then any outside variable v can only occur negatively in a $\mathcal{C}$-bad clause and moreover setting v to 1 ensures that the $\mathcal{C}$-bad clauses remain $\mathcal{C}$-bad.

Next, we need to ensure that any outside variable cannot destroy too many paths. By choosing a *shortest* path P_{i+1}, we have already ensured that no variable occurs on more than two clauses of P_{i+1} (such a variable would be a shortcut, meaning P_{i+1} was not a shortest path). Moreover, because P_{i+1} is a shortest path from b_{i+1} to T_i, every variable that occurs on T_i and on P_{i+1} must occur in the clause c in P_{i+1} that is closest to T_i but not in T_i itself. Similarly, to how we dealt with the $\mathcal{C}$-bad clauses, we will now add all variables that α-occur in c to X. This ensures that no outside variable can α-occur in both T_i and P_{i+1}, which (by induction over i) implies that every outside variable α-occurs in at most two clauses (either from T_i or from P_{i+1}) and therefore provides us with an assignment for the outside variables that removes at most two clauses from X. However, since removing any single clause can be arbitrarily bad if the clause has a high degree in the separator obstruction, we further need to ensure that all clauses of the separator obstruction in which outside variables α-occur have small degree. We achieve this by adding the variables α-occurring in any clause as soon as its degree (in the separator obstruction) becomes larger than two, which happens whenever the endpoint of P_{i+1} in T_i is a clause. Finally, if the endpoint of P_{i+1} in T_i is a variable, we also add this variable to the separator obstruction to ensure that no variable has degree larger than three in T_{i+1}. This leads us to the following definition of separator obstructions (see also Figs. 3–7 for an illustration).

Definition 9. Let $F \in \mathcal{CNF}$ and $\mathcal{C} = \mathcal{C}_{\alpha,s}$ with $\alpha \subseteq \{+,-\}$, $\alpha \neq \emptyset$, and $s \in \mathbb{N}$. A *$\mathcal{C}$-separator obstruction* for F is a tuple $X = \langle\langle P_1, \ldots, P_\ell\rangle, \tau\rangle$ (where $P_1, \ldots, P_\ell$ are paths in F and τ is an assignment of variables of F) satisfying the following recursive definition.

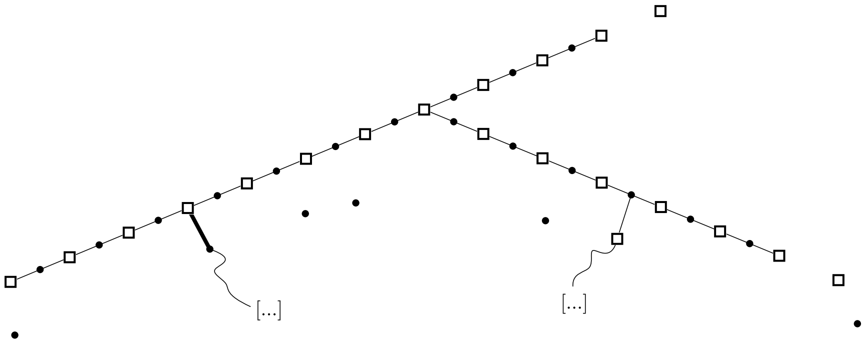

Fig. 5. The graph G_{i-1} is obtained from $F[\tau_{i-1}]$ by adding all clauses of T_{i-1} and keeping precisely the edges corresponding to α-occurring variables (that is, bold edges). In the following two illustrations, we make a case distinction on how the clause b_i connects to G_{i-1}.

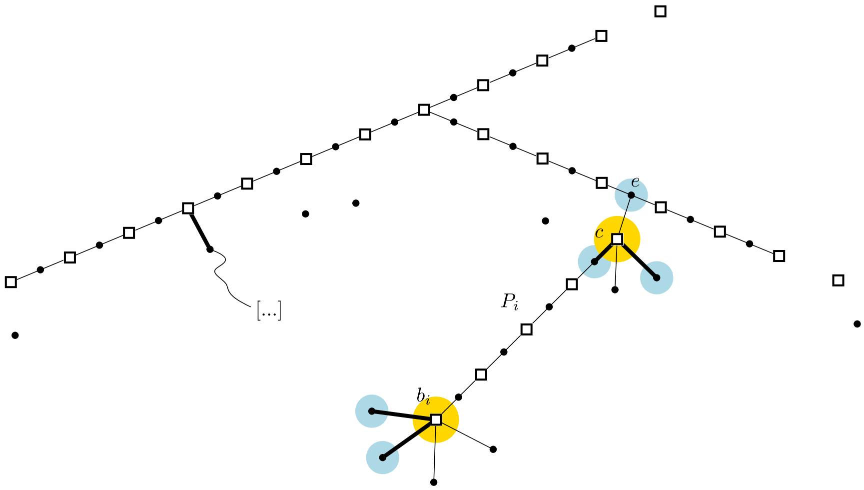

Fig. 6. First case: e is a variable. We highlight in orange the clauses in $B_i \setminus B_{i-1}$ and in light blue the variables in $V_i \setminus V_{i-1}$.

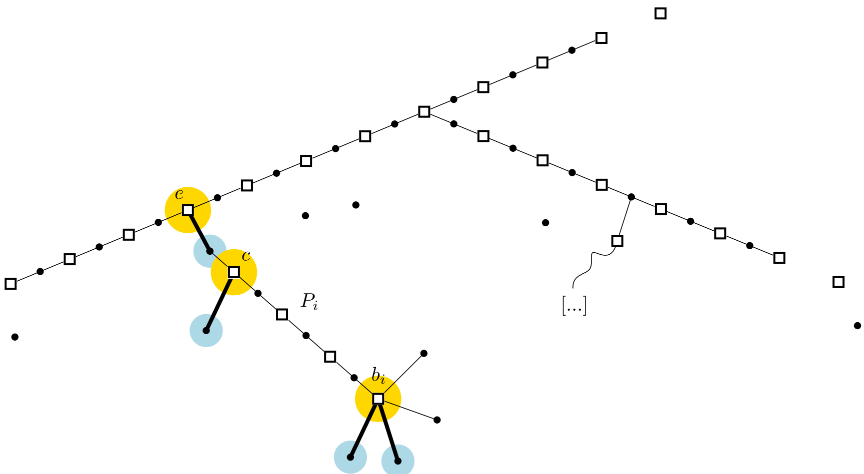

Fig. 7. Second case: e is a clause. We again highlight in orange the clauses in $B_i \setminus B_{i-1}$ and in light blue the variables in $V_i \setminus V_{i-1}$.

- P_1 is a shortest path between two $\mathcal{C}$-bad clauses b_0 and b_1 in F. Let $B_1 = \{b_0, b_1\}$, let V_1 be the set of all variables that α-occur in any clause in B_1, let $\tau_1 : V_1 \to \{0, 1\}$ be any assignment of the variables in V_1, and let $T_1 = P_1$.
- Let G_{i-1} be the graph obtained from the incidence graph of $F[\tau_{i-1}]$ after:
  - adding all clauses in T_{i-1} that are not in $F[\tau_{i-1}]$,

- adding an edge between any variable v of $F[\tau_{i-1}]$ and any clause of c of T_{i-1} such that v α-occurs in c in F.
- removing all edges between any clause c in T_{i-1} and any variable v of $F[\tau_{i-1}] \setminus T_{i-1}$ such that v does not α-occurs in c.

For every i with $1 < i \leq \ell$, let b_i be a $\mathcal{C}$-bad clause in $F[\tau_{i-1}]$ of minimal distance to T_{i-1} in G_{i-1}. Then, P_i is a shortest path (of possibly length zero) in G_{i-1} from b_i to T_{i-1} and $T_i = T_{i-1} \cup P_i$. Moreover, let e be the variable or clause that is both in T_{i-1} and P_i. We define B_i and V_i by initially setting $B_i = B_{i-1} \cup \{b_i\}$ and $V_i = V_{i-1} \cup var_\alpha(b_i)$ and distinguishing two cases:

- If e is a variable, then let c be the clause on P_i incident with e (note that it is possible that $c = b_i$). Then, we add c to B_i and we add $\{e\} \cup var_\alpha(c)$ to V_i.
- If e is a clause, then either $e = b_i$ or $e \neq b_i$ and there is a clause c that is closest to e on P_i (it may be that $c = b_i$). In the former case we leave B_i and V_i unchanged and in the latter case, we add e and c to B_i and we add $var_\alpha(e) \cup var_\alpha(c)$ to V_i.

$\tau_i : V_i \to \{0, 1\}$ is any assignment of the variables in V_i that is compatible with τ_{i-1}.

We set $\tau = \tau_\ell$. The *size* of X is the number of paths in $T = T_\ell$, i.e., $\ell + 1$.

We comment on the role of the assignment τ as part of the above definition: In Lemma 11, the assignment will be used to define a winning strategy for connector, which will then show that separator obstructions indeed yield lower bounds on the backdoor depth.

We start by observing some simple but important properties of separator obstructions.

Lemma 10. *lemma Let $F \in \mathcal{CNF}$, $\mathcal{C} = \mathcal{C}_{\alpha,s}$ with $\alpha \subseteq \{+,-\}$, $\alpha \neq \emptyset$, and $s \in \mathbb{N}$, and let $X = \langle\langle P_1, \ldots, P_\ell\rangle, \tau\rangle$ be a $\mathcal{C}$-separator obstruction in F, then for every $i \in [\ell]$:*

(C1) T_i is a tree.
(C2) Every variable $v \notin V_i$ occurs in at most three clauses of P_j for every j with $1 \leq j \leq i+1$.
(C3) Every variable $v \notin V_i$ α-occurs in at most two clauses of P_j for every j with $1 \leq j \leq i+1$. Moreover, these clauses are consecutive in P_j.
(C4) Every variable $v \notin V_i$ α-occurs in at most two clauses of T_i and moreover those clauses are consecutively contained in one path of T_i.
(C5) Every variable $v \in V_i \setminus V_{i-1}$ α-occurs in most four clauses of T_i.
(C6) If a variable $v \notin V_i$ α-occurs in a clause c of T_i, then c has degree at most two in T_i.
(C7) The degree of every clause in T is at most equal to the number of variables that α-occur in c plus two.
(C8) Every variable of F has degree at most three in T.
(C9) If every clause of F contains at most x α-literals, then $|V_i \setminus V_{i-1}| \leq 2s + x + 1$.

Proof. We show (C1) by induction on i. (C1) clearly holds for $i = 1$. For $i > 1$, note that T_i is obtained from T_{i-1} by adding the path P_i that intersects T_{i-1} in at most one variable or clause. Since T_{i-1} is a tree so is T_i.

Towards showing (C2), observe first that because $v \notin V_i$, it holds that v is in $F[\tau_{j-1}]$ and therefore also in G_{j-1} for every $j \leq i+1$. Consider a clause c of P_j that is not in T_{j-1}; note that there is at most one clause on P_j that is in T_{j-1}. If v occurs in c in F, then G_{j-1} contains the edge between v and c. Therefore, if v is contained in at least three such clauses on P_j, then P_j would no longer be a shortest path in G_{j-1}. Consequently, v is contained in at most two such clauses in P_j (which are consecutive). Finally, because P_j contains at most one clause that is also in T_{i-1}, we obtain that in total v is contained in at most three clauses of P_j.

The proof for (C3) is very similar to the proof of (C2). One only needs to additionally observe that if v α-occurs in the at most one clause of P_j that is also in T_{j-1}, then G_{j-1} contains the edge between v and this clause; which is not the case if v occurs but does not α-occur in the clause.

We establish (C4) by induction on i. For $i = 1$, this follows immediately from (C3) because $T_1 = P_1$. Now suppose that the claim holds for $i - 1$. Then, v α-occurs in at most two consecutive clauses of some path of T_{i-1}. Moreover, because of (C3), v α-occurs in at most two consecutive clauses of P_i. We claim that it is not possible that v α-occurs both in a clause c_{i-1} of T_{i-1} that is not in P_i and in a clause c_i in P_i that is not in T_{i-1}. Clearly, $c_{i-1} \neq c_i$. Note that because $v \notin V_i$, it holds that v is in G_{i-1} and moreover because v α-occurs in c_{i-1} and c_i, it holds that G_{i-1} contains the edge between v and c_{i-1} and the edge between v and c_i. Therefore, because P_i is a shortest path in G_{i-1}, we obtain that c_i must be the clause in $P_i \setminus T_{i-1}$ that is closest to T_{i-1}. But then, $c_i \in B_i$, which contradicts our assumption that $v \notin V_i$.

Towards showing (C5), first note that because of (C4), and the fact that $v \notin V_{i-1}$, we obtain that v can α-occur in at most two clauses of T_{i-1}. Moreover, it follows from (C3) that v can α-occur in at most two (consecutive) clauses of P_i. Therefore, v α-occurs in at most four clauses of T_i.

Towards showing (C6), first observe that if c is a clause with degree larger than 2 in T_i, then $c \in B_i$. This is because for c to have degree larger than 2, it must be contained in more than one path of T_i, i.e., there must be an index $j \leq i$ such that c is contained in both T_{j-1} and P_j. But then, $c \in B_j \subseteq B_i$. Now suppose for a contradiction that there is a clause c with degree larger than two in which a variable $v \notin V_i$ α-occurs. Then, $c \in B_i$ and because $v \in var_\alpha(c)$, we obtain that $v \in V_i$, a contradiction.

Towards showing (C7), let c be a clause of T. If c occurs in only one path of T, then c has degree 2 in T. Otherwise, let i be the smallest integer such that c is contained in P_i. Then, for every $j > i$, it holds that if P_j contains c, then the variable on P_j that is adjacent to c α-occurs in c. Therefore, the degree of c in T is at most two plus the number of variables that α-occur in c.

Towards showing (C8), let v be any variable of F. If v occurs in at most one path of T, then v has degree at most two in T. Moreover, if not then let i be the smallest number such that v is contained in two paths of T_i. Then v has degree at most three in T_i and is the endpoint of the path P_i in T_{i-1} and therefore v is added to V_i. However, this implies that v will not appear on any path P_j for $j > i$ (because any such path P_j is a path in G_i, which does no longer contain v) and therefore the degree of v in T will be at most three.

We finish by showing (C9). We say that a path P of F (i.e., a path of G_F) is *$\mathcal{C}$-good* if so are all clauses occurring as inner vertices on P. Note that the paths P_i for any $i > 1$ in the above definition are necessarily $\mathcal{C}$-good paths due to the definition of b_i. Because of the definition of $\mathcal{C}$-separator obstructions, it holds that $V_i \setminus V_{i-1}$ is either equal to $var_\alpha(b_i) \cup \{a\} \cup var_\alpha(c)$ or equal to $var_\alpha(b_i) \cup var_\alpha(e) \cup var_\alpha(c)$ for some $\mathcal{C}$-bad clause b_i, variable a, and $\mathcal{C}$-good clauses c and e; note that we can assume that c and e are $\mathcal{C}$-good since for every $i > 2$, P_i is a $\mathcal{C}$-good path and therefore all clauses on P_i apart from b_i are $\mathcal{C}$-good. Since $\mathcal{C}$-good clauses contain at most s α-literals and by assumption every other clause contains at most x α-literals, we obtain that $|V_i \setminus V_{i-1}| \leq x + 2s + 1$. $\square$

We now show the main result of this subsection, namely, that also separator obstructions can be used to obtain a lower bound on the backdoor depth of CNF formulas.

Lemma 11. *Let $\mathcal{C} = \mathcal{C}_{\alpha,s}$ with $\alpha \subseteq \{+,-\}$, $\alpha \neq \emptyset$, and $s \in \mathbb{N}$ and $F \in \mathcal{CNF}$. If F has a $\mathcal{C}$-separator obstruction of size at least $\ell = \left((6s+6d+19)^d(80+7d)\right)^{2^d}$, then F has $\mathcal{C}$-backdoor depth at least d.*

Proof. Let $X = \langle\langle P_1, \ldots, P_\ell\rangle, \tau\rangle$ be a $\mathcal{C}$-separator obstruction for F of size at least ℓ with V_i, B_i, G_i, T_i, T as in Definition 9. Consider the following three definitions that will be used extensively throughout the whole proof.

- Let J be a position in the game $\textsc{Game}(F, \mathcal{C})$. We say that a subtree T' of $T = T_\ell$ is *contained* in J if every variable and clause of T' occurs in J.
- Let T' be a subtree of T that is contained in J. Let P_j be a path of X. We say that P_j is *active* in T' if either $V(P_j) = \{b_j\}$ and T' contains b_j or T' contains a vertex in $V(P_j) \setminus V(T_{j-1})$.
- Moreover, we say that P_j is *intact* in T' at position J if $V(P_j) \subseteq V(T')$ and b_j is a $\mathcal{C}$-bad clause in J. Otherwise, we say that P_j is *broken* in T' at position J.

For the remainder of the proof, we will prove the following.

> **Main claim.** For every $0 \leq i \leq d$ and position J reached after i rounds in the game $\textsc{Game}(F, \mathcal{C})$ against S, there is a subtree T' of T contained in J, such that T' contains at least $\ell_i = \ell^{(1/2)^i}/(6s+6d+19)^i$ intact paths and at most $z_i = 7i$ broken paths of X.

Note that $\ell_d = \ell^{1/2^d}/(6s+6d+19)^d = 80+7d \geq 1$ and that indicate paths contain $\mathcal{C}$-bad clauses. Hence, for $i = d$, the main claim implies that any position J reached after d rounds in the game $\textsc{Game}(F, \mathcal{C})$ contains at least one clause that is $\mathcal{C}$-bad in J. This then implies the statement of the lemma.

We will prove the main claim by induction on i. It clearly holds for $i = 0$ since $\ell_0 = \ell$ and $z_0 = 0$ and the connector can choose the component of F containing T. Assume now that $i > 0$ and let J be the position reached after $i-1$ rounds. By the induction hypothesis, at position J there is a subtree T' of T contained in J containing at least $\ell_{i-1} = \ell^{(1/2)^{i-1}}/(6s+6d+19)^{i-1}$ intact paths and at most $z_{i-1} = 7(i-1)$ broken paths of X. Suppose that the splitter chooses variable v as its next move. Moreover, let o be the smallest integer such that $v \in V_o$; if $v \notin V_\ell$ we set $o = \ell + 1$. Note that $v \notin V_j$ for every $j < o$. Let I be the set of all paths P_j of X that are intact in T' at position J and let $I_{<o}$ ($I_{>o}$) be the subset of I containing only the paths P_j with $j < o$ ($j > o$). Finally, let $T'_{<o}$ be the subtree of T' restricted to the paths P_j of X with $j < o$. Note that at position J, $T'_{<o}$ is connected and the paths in $I_{<o}$ are intact also in $T'_{<o}$. Then, the connector chooses the assignment $\beta : \{v\} \to \{0, 1\}$ such that:

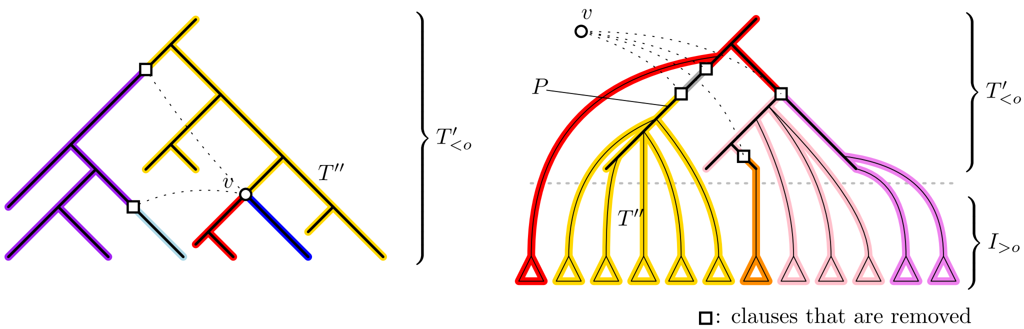

Fig. 8. Left: Case 1. The set $I_{<o}$ is large. Assigning v to $\beta(v)$ decomposes the tree $T'_{<o}$ into at most seven components. The largest component T'' is still large. Right: Case 2. The set $I_{<o}$ is small. There is a path P to which many paths are weakly attached, forming a tree T_P. Assigning v to $\beta(v)$ splits T_P into few parts. The largest component T'' of T_P is still large.

$$\beta(v) = \begin{cases} \tau(v) & |I_{<o}| < \sqrt{\ell_{i-1}}, \\ 1 & |I_{<o}| \geq \sqrt{\ell_{i-1}} \text{ and } + \in \alpha, \\ 0 & \text{otherwise.} \end{cases}$$

As we will show below, β is defined in such a manner that the position $J' = J[\beta]$ reached after the next round of the game $\textsc{Game}(F, \mathcal{C})$ contains a subtree T'' of T' containing at least $\ell_i = \sqrt{\ell_{i-1}}/(6s + 6d + 19)$ paths that are intact in J' and at most $z_i = z_{i-1} + 7$ broken paths. This then completes the proof of the main claim, since the connector can now choose the component of J' containing T''. We distinguish the following cases; refer also to Fig. 8 for an illustration of the two cases.

Case 1: $|I_{<o}| \geq \sqrt{\ell_{i-1}}$. We will show that T'' can be obtained as a subtree of $T'_{<o}$.

Note first that all clauses b_j with $j < o$ that are $\mathcal{C}$-bad in J are also $\mathcal{C}$-bad in J'. This is because $v \notin V_j$ (since $j < o$ and $v \notin V_{o-1}$) and therefore v cannot α-occur in b_j, which implies that b_j remains $\mathcal{C}$-bad and not satisfied after setting v to $\beta(v)$.

The tree $T'_{<o}$ in J may decompose into multiple components in J'. We will argue that one of these components contains many intact paths and only at most two more broken paths than $T'_{<o}$. Since the $\mathcal{C}$-bad clauses of an intact path remain $\mathcal{C}$-bad in J', the only way in which an intact path can become broken is if some parts of the path get removed, i.e., either v or clauses satisfied by setting v to $\beta(v)$.

If $\beta(v) = 1$ then $+ \in \alpha$. If $\beta(v) = 0$ then $+ \notin \alpha$, and since $\alpha \neq \emptyset$, then $- \in \alpha$. Thus, in $J' = J[\beta]$, the only elements that are removed are the variable v as well as clauses in which v α-occurs. By Lemma 10 (C4), v α-occurs in at most two clauses of $T'_{<o}$ and because of (C6) those clauses have degree at most two in $T'_{<o}$. Therefore, setting v to $\beta(v)$ removes at most two clauses from $T'_{<o}$, each of which having degree at most two. Moreover, according to Lemma 10 (C8), v itself has degree at most three in $T'_{<o}$. This implies that setting v to $\beta(v)$ splits $T'_{<o}$ into at most $2 \cdot 2 + 3 = 7$ components. Moreover, using the same arguments one can show that assigning v to $\beta(v)$ can create at most $2 * 2 + 3 = 7$ new broken paths among the paths in $T'_{<o}$. Therefore, there is a component of J' that contains a subtree of $T'_{<o}$ that contains at least $|I_{<o}|/7 - 7 \geq \sqrt{\ell_{i-1}}/7 - 7$ intact paths and at most $z_{i-1} + 7 \leq 7i = z_i$ broken paths of X. Note that $\sqrt{\ell_{i-1}} \geq \ell_d \geq 80 + 7d \geq 80$ and therefore $\sqrt{\ell_{i-1}}/7 - 7 \geq \sqrt{\ell_{i-1}}/(6s + 6d + 19) = \ell_i$.

Case 2: $|I_{<o}| < \sqrt{\ell_{i-1}}$. This means $\beta(v) = \tau(v)$. In this case, we will build the subtree T'' by picking only one path from $T'_{<o}$ and the remaining paths from $P_{o+1}, \ldots, P_\ell$. Let A be the set of all paths of X that are active in T' and let $A_{>o}$ ($A_{<o}$) be the subset of A containing only the paths P_j with $j > o$ ($j < o$). We say that a path P_a of X is *attached* to a path P_b of X if $a > b$, $V(P_a) \cap V(P_b) \neq \emptyset$ and there is no $b' < b$ with $V(P_a) \cap V(P_{b'}) \neq \emptyset$. We say that a path P_a in $A_{>o}$ is *weakly attached* to a path P_b in $A_{<o}$ if either:

- P_a is attached to P_b or
- P_a is attached to a path P_c in $A_{>o}$ that is in turn weakly attached to P_b.

Note that because T' is a tree, every path in $A_{>o}$ is weakly attached to exactly one path in $A_{<o}$. Moreover, for the same reason, any path in $A_{<o}$, together with all paths in $A_{>o}$ that are weakly attached to it, induces a subtree of T'.

Therefore, there is a path P in $A_{<o}$ such that at least $|I_{>o}|/|A_{<o}|$ paths in $I_{>o}$ are weakly attached to P. Let T_P be obtained by inducing T' on P and all paths in $A_{>o}$ that are weakly attached to P. As argued above, T_P is a tree. Note that T_P has at least $|I_{>o}|/|A_{<o}|$ paths that are intact in T_P and at most z_{i-1} paths that are broken in T_P at position J. Since $\sqrt{\ell_{i-1}} \geq \ell_d = 80 + 7d \geq 7d$, $z_{i-1} \leq z_d = 7d$, and $|I_{<o}| \leq \sqrt{\ell_{i-1}}$, it holds that $|I_{<o}| + z_{i-1} \leq 2\sqrt{\ell_{i-1}}$. Therefore, at least

$$\begin{aligned}
|I_{>o}|/|A_{<o}| &\geq (\ell_{i-1} - |I_{<o}|)/(|I_{<o}| + z_{i-1}) \\
&\geq (\ell_{i-1} - |I_{<o}|)/2|I_{<o}| \\
&\geq \ell_{i-1}/2\sqrt{\ell_{i-1}} - 1/2 \\
&\geq \sqrt{\ell_{i-1}}/2 - 1/2 \\
&\geq (6s + 6d + 19)\ell_i/2 - 1/2 \\
&\geq (3s + 3d + 9)\ell_i.
\end{aligned}$$

paths in $I_{>o}$ are weakly attached to P. Let us now analyse what happens with T_P when going from J to J'. First consider a path P_j with $j > o$. Because $\beta(v) = \tau(v)$, all clauses in P_j apart from the at most one clause that is also in P remain in $J' = J[\beta]$, moreover, if such a clause is $\mathcal{C}$-bad in J, then it is also $\mathcal{C}$-bad in J'. Therefore, if P_j does not contain a clause from P, then its status in J' is the same as its status in J, i.e., if P is active (intact) in J, then P_j is active (intact) in J'. Let O be the set of all clauses of P in which v occurs. Because of Lemma 10 (C2), $|O| \leq 3$. Because of Lemma 8, we can assume that every clause in F contains at most $s + d$ α-literals and therefore we obtain from Lemma 10 (C7) that every clause in O has degree at most $s+d+2$ in T and therefore also in T_P. Since v has degree at most three in T_P (Lemma 10 (C8)), we obtain that T_P splits into at most $3(s+d+2)+3 = 3s+3d+9$ components in J'. Therefore, $J' = J[\beta]$ contains a component that contains a subtree T'' of T_P with at least ℓ_i paths that are intact in T'' and at most $z_{i-1} + 1 \leq z_i$ paths that are broken in T''. □

6. Basic properties of obstruction trees

In this section, we prove with Lemma 7 that obstruction trees yield indeed a lower bound on the backdoor depth.

Lemma 12. *Let $F \in \mathcal{CNF}$ and $\mathcal{C} = \mathcal{C}_{\alpha,s}$ with $\alpha \subseteq \{+,-\}$, $\alpha \neq \emptyset$, and $s \in \mathbb{N}$. Let β be a partial assignment of the variables in F and T be a $\mathcal{C}$-obstruction tree of depth d in $F[\beta]$. Then, T is also a $\mathcal{C}$-obstruction tree of depth d in F.*

Proof. We use induction on d. If $d = 0$, then there is a $\mathcal{C}$-bad clause c of $F[\beta]$ such that $T = \{c\}$. Therefore, c is also a $\mathcal{C}$-bad clause in F and T is a $\mathcal{C}$-obstruction tree of depth 0 in F.

Towards showing the induction step, let $d > 0$. Then there is a $\mathcal{C}$-obstruction tree T_1 in $F[\beta]$ of depth $d-1$, an assignment β' of the variables in $F[\beta]$ and a $\mathcal{C}$-obstruction tree T_2 in $F[\beta \cup \beta']$ of depth $d-1$ such that no variable of $F[\beta \cup \beta']$ that is contained in a clause of T_2 is contained in T_1 or α-occurs in a clause of T_1. Moreover, there is a path P connecting T_1 and T_2 in F. Because of the induction hypothesis, T_1 is a $\mathcal{C}$-obstruction tree in F of depth $d-1$. Therefore, $T = T_1 \cup T_2 \cup var(P) \cup P$ is a $\mathcal{C}$-obstruction tree in F of depth d. □

Lemma 13. *Let $F \in \mathcal{CNF}$ and $\mathcal{C} = \mathcal{C}_{\alpha,s}$ with $\alpha \subseteq \{+,-\}$, $\alpha \neq \emptyset$, and $s \in \mathbb{N}$. A set T is a $\mathcal{C}$-obstruction tree of depth d in F if and only if there is a component F' of F such that T is a $\mathcal{C}$-obstruction tree of depth d in F'.*

Proof. This follows because all variables and clauses belonging to T induce a connected subgraph of F. □

Lemma 14. *Let $F \in \mathcal{CNF}$ and $\mathcal{C} = \mathcal{C}_{\alpha,s}$ with $\alpha \subseteq \{+,-\}$, $\alpha \neq \emptyset$, and $s \in \mathbb{N}$. Let T be a $\mathcal{C}$-obstruction tree of depth d in F, $v \in var(F)$ be a variable not occurring in any clause of T, and τ be an assignment to that variable. Then, T is a $\mathcal{C}$-obstruction tree of depth d in $F[\tau]$.*

Proof. Because v does not appear in any clause of T, all clauses of T in F are still present in $F[\tau]$ and contain the same literals as in F; this also implies that every such clause is $\mathcal{C}$-good ($\mathcal{C}$-bad) in F if and only if it is in $F[\tau]$. Moreover, because every variable of T occurs in some clause of T, it also follows that v is not contained as a variable in T. Therefore, T has the same set of variables and clauses in F as in $F[\tau]$ and moreover all clauses remain the same, which shows the lemma. □

Lemma 15. *Let $F \in \mathcal{CNF}$ and $\mathcal{C} = \mathcal{C}_{\alpha,s}$ with $\alpha \subseteq \{+,-\}$, $\alpha \neq \emptyset$, and $s \in \mathbb{N}$. Let T be a $\mathcal{C}$-obstruction tree of depth d in F, $v \in var(F)$ be a variable that neither occurs in T nor α-occurs in any clause of T, and τ be the assignment with $\tau(v) = 1$ if $+ \in \alpha$ and $\tau(v) = 0$, otherwise. Then, T is a $\mathcal{C}$-obstruction tree of depth d in $F[\tau]$.*

Proof. If $\alpha = \{+,-\}$, the statement of the lemma follows immediately from Lemma 14. Otherwise, we can assume without loss of generality that $\alpha = \{+\}$; for the only remaining case that $\alpha = \{-\}$, the proof is analogous.

Thus, v does not occur positively in any clause of T and $\tau(v) = 1$, which means that all clauses of T are still present in $F[\tau]$ and moreover any $\mathcal{C}$-bad clauses of T are still $\mathcal{C}$-bad in $F[\tau]$. Because v does not occur in T, all variables and clauses of T are still contained in $F[\tau]$ and moreover all $\mathcal{C}$-bad clauses of T are still $\mathcal{C}$-bad in $F[\tau]$, which shows the lemma. □

We are now ready to show the most crucial property of obstruction trees, namely, that they can be used to obtain lower bounds for the backdoor depth of a CNF formula.

Lemma 7. *Let $F \in \mathcal{CNF}$ and $\mathcal{C} = \mathcal{C}_{\alpha,s}$ with $\alpha \subseteq \{+,-\}$, $\alpha \neq \emptyset$, and $s \in \mathbb{N}$. If there is a $\mathcal{C}$-obstruction tree of depth d in F, then the $\mathcal{C}$-backdoor depth of F is larger than d.*

Proof. Assume there exists a $\mathcal{C}$-obstruction tree of depth d in F. We will show the following claim by induction on i.

> **Claim.** Let $0 \leq i \leq d$. The connector has a strategy for the game $\textsc{Game}(F, \mathcal{C})$ to reach within i rounds a position J such that there is a $\mathcal{C}$-obstruction tree in J of depth $d - i$.

For $i = d$, this claim allows the connector to reach after d rounds a position that contains a $\mathcal{C}$-obstruction tree of depth 0, i.e., a $\mathcal{C}$-bad clause. Thus, the splitter has no strategy to win the game after at most d rounds and by Observation 3, the statement of this lemma is proven. It hence remains to prove the claim by induction on i.

For $i = 0$, the claim holds trivially. So suppose that $i > 0$ and let J be the position reached by the connector after $i - 1$ rounds. Then, by the induction hypothesis, J contains a $\mathcal{C}$-obstruction tree T of depth $d - i + 1 > 1$.

Since the depth is at least one, there further exist (by Definition 6) a $\mathcal{C}$-obstruction tree T_1 of depth $d - i$ in J, a partial assignment β of the variables in J, and a $\mathcal{C}$-obstruction tree T_2 of depth $d - i$ in $J[\beta]$ such that no variable $v \in var(J[\beta])$ that is contained in a clause of T_2 is in T_1 or α-occurs in a clause of T_1. Now, let v be the next variable chosen by the splitter. We distinguish the following cases:

1. Assume $v \notin var(J[\beta])$. This means $v \in var(\beta)$. Then, the connector can choose the assignment $v = \beta(v)$ for v. Because of Lemma 12, T_2 is a $\mathcal{C}$-obstruction tree of depth $d - i$ in $J[v = \beta(v)]$. Next, the connector chooses the component of $J[v = \beta(v)]$ containing T_2. By Lemma 13, T_2 is also a $\mathcal{C}$-obstruction tree in this component. Since T_2 of depth $d - i$, this proves the claim.
2. Assume $v \in var(J[\beta])$ and v does not occur in a clause of T_2. Then, as the connector, we assign v an arbitrary value. Let τ be this assignment. By Lemma 14, T_2 is a $\mathcal{C}$-obstruction tree of depth $d - i$ in $J[\tau]$. We choose the connected component J' containing T_2 in $J[\tau]$. By Lemma 13 T_2 is also a $\mathcal{C}$-obstruction tree of depth $d - i$ in J'. This proves the claim.
3. Otherwise, $v \in var(J[\beta])$ and v occurs in a clause of T_2. Since T is a $\mathcal{C}$-obstruction tree in J, it follows by Definition 6 that v does neither occur in T_1 nor does it α-occur in a clause of T_1. Let τ be the assignment with $\tau(v) = 1$ if $+ \in \alpha$ and $\tau(v) = 0$, otherwise. By Lemma 15, T_1 is a $\mathcal{C}$-obstruction tree of depth $d - i$ in $J[v = \tau(v)]$. Next, the connector chooses the component of $J[v = \tau(v)]$ containing T_1. By Lemma 13, T_1 is also a $\mathcal{C}$-obstruction tree in this component. Since T_1 of depth $d - i$, this proves the claim. □

7. Winning strategies and algorithms

We are ready to present our algorithmic results. Earlier, we discussed that separator obstructions are used to separate existing obstruction trees from future obstruction trees. As all obstruction trees are built only from shortest paths, it is sufficient to derive a splitter-algorithm that takes a shortest path P and separates it from all future obstructions. By reaching a position J such that no variable in $var(J)$ occurs in a clause of P, we are guaranteed that all future obstructions are separated from P, as future obstructions will only contain clauses and variables from J.

Lemma 16. *Let $\mathcal{C} = \mathcal{C}_{\alpha,s}$ with $\alpha \subseteq \{+,-\}$, $\alpha \neq \emptyset$, and $s \in \mathbb{N}$. There exists a splitter-algorithm that implements a strategy to reach for each game $\textsc{Game}(F, \mathcal{C})$, non-negative integer d, and shortest path P between two $\mathcal{C}$-bad clauses in F within at most*

$$(3s + d + 1)\big((6s + 6d + 19)^d (80 + 7d)\big)^{2^d}$$

rounds either:

1) a winning position, or
2) a position J such that no variable in $var(J)$ occurs in P or α-occurs in a clause of P, or
3) a proof that the $\mathcal{C}$-backdoor depth of F is at least d.

This algorithm takes at most $\mathcal{O}(\|F\|)$ time per move.

Proof. If a clause of F contains more than $d + s$ α-literals, then this constitutes by Lemma 8 a proof that the $\mathcal{C}$-backdoor depth of F is at least d and we achieve case 3) of the lemma. Thus, we can assume that every clause in every position of the game contains at most $d + s$ α-literals.

Let $\langle\langle P_1, \ldots, P_\ell\rangle, \tau\rangle$ be a $\mathcal{C}$-separator obstruction for F and let τ' be a sub-assignment of τ assigning at least all variables in $V_{\ell-1}$. Then, we call $\langle\langle P_1, \ldots, P_\ell\rangle, \tau'\rangle$ a *partial $\mathcal{C}$-separator obstruction* for F. Consider the following splitter-algorithm, where we associate with each position J of the game $\textsc{Game}(F, \mathcal{C})$ a partial $\mathcal{C}$-separator obstruction $X(J)$ of the form $X(J) = \langle\langle P_1, \ldots, P_\ell\rangle, \tau_J\rangle$ with $P_1 = P$. We set $X(S) = \langle\langle P\rangle, \emptyset\rangle$ for the starting position S of the game.

Then, the splitter-algorithm does the following for a position J in $\textsc{Game}(F,\mathcal{C})$. Let $X(J) = \langle\langle P_1,\ldots,P_\ell\rangle, \tau_J\rangle$ and V_i, B_i, G_i, T_i, T as in Definition 9. If there is at least one variable in $V_\ell \setminus V_{\ell-1}$ (where we set $V_0 = \emptyset$) that has not yet been assigned by τ_J, the splitter chooses any such variable. Otherwise, $X(J)$ is a $\mathcal{C}$-separator obstruction and we distinguish the following cases:

1. If there is a $\mathcal{C}$-bad clause in J that has a path to some vertex of T_ℓ in G_ℓ, then let $b_{\ell+1}$ be a $\mathcal{C}$-bad clause that is closest to any vertex of T_ℓ in G_ℓ and let $P_{\ell+1}$ be a shortest path from $b_{\ell+1}$ to some vertex of T_ℓ in G_ℓ. Note that $\langle\langle P_1,\ldots,P_\ell,P_{\ell+1}\rangle,\tau_J\rangle$ is a partial $\mathcal{C}$-separator obstruction for F. The splitter now chooses any variable in $V_{\ell+1}\setminus V_\ell$ and assigns $X(J') = \langle\langle P_1,\ldots,P_\ell,P_{\ell+1}\rangle,\tau_{J'}\rangle$ for the position J' resulting from this move.
2. Otherwise, $X(J)$ can no longer be extended and either: (1) there is no $\mathcal{C}$-bad clause in J, or (2) every $\mathcal{C}$-bad clause of J has no path to T_ℓ in G_ℓ. In situation (1), we reached a winning position and therefore achieved case 1) of this lemma. In situation (2), no variable of J occurs in T_ℓ or α-occurs in a clause of T_ℓ. Hence, the same holds for P, and we achieved case 2) of this lemma.

This completes the description of the splitter-algorithm. Moreover, if every play against the splitter-algorithm ends after at most $(3s+d+1)\big((6s+6d+19)^d(80+7d)\big)^{2^d}$ rounds, every position is either of type 1) or type 2) and we are done. Otherwise, after playing for $(3s+d+1)\big((6s+6d+19)^d(80+7d)\big)^{2^d}$ rounds we reach a position J. As stated at the beginning of the proof, every clause contains at most $d+s$ α-literals, and therefore we obtain from Lemma 10 (C9) that $|V_{i+1}\setminus V_i| \le 3s+d+1$. This means that the size of the $\mathcal{C}$-separator obstruction increases by at least 1 after at most $3s+d+1$ rounds. Since J was reached after $(3s+d+1)\big((6s+6d+19)^d(80+7d)\big)^{2^d}$ rounds, the size has increased sufficiently often to guarantees the existence of a $\mathcal{C}$-separator obstruction $X(J)$ of size at least $\big((6s+6d+19)^d(80+7d)\big)^{2^d}$. By Lemma 11, this is a proof that F has $\mathcal{C}$-backdoor depth at least d.

Finally, the splitter-algorithm takes time at most $\mathcal{O}(\|F\|)$ per round since a $\mathcal{C}$-bad clause that is closest to the current $\mathcal{C}$-separator obstruction and the associated shortest path can be found using a simple breadth-first search. $\square$

Since selecting more variables can only help the splitter in achieving their goal, we immediately also get the following statement from Lemma 16.

Corollary 17. *Consider $\mathcal{C} = \mathcal{C}_{\alpha,s}$ with $\alpha \subseteq \{+,-\}$, $\alpha \neq \emptyset$, and $s \in \mathbb{N}$, a game $\textsc{Game}(F,\mathcal{C})$ and a position J' in this game, a non-negative integer d and shortest path P between two $\mathcal{C}$-bad clauses in F. There exists a splitter-algorithm that implements a strategy that continues the game from position J' and reaches within at most*

$$(3s+d+1)\big((6s+6d+19)^d(80+7d)\big)^{2^d}$$

rounds either:

1) a winning position, or
2) a position J such that no variable in $var(J)$ occurs in P or α-occurs in a clause of P, or
3) a proof that the $\mathcal{C}$-backdoor depth of F is at least d.

This algorithm takes at most $\mathcal{O}(\|F\|)$ time per move.

As described at the end of Section 4, we can now construct in the following lemma obstruction trees of growing size, using the previous corollary to separate them from potential future obstruction trees.

Lemma 18. *Let $\mathcal{C} = \mathcal{C}_{\alpha,s}$ with $\alpha \subseteq \{+,-\}$, $\alpha \neq \emptyset$, and $s \in \mathbb{N}$. There is a splitter-algorithm that implements a strategy to reach for a game $\textsc{Game}(F,\mathcal{C})$ and non-negative integers i, d with $0 \le i \le d$ within at most $2^i(3s+d+1)\big((6s+6d+19)^d(80+7d)\big)^{2^d}$ rounds either:*

1) a winning position, or
2) a position J and a $\mathcal{C}$-obstruction tree T of depth i in F such that no variable in $var(J)$ occurs in T or α-occurs in a clause of T, or
3) a proof that the $\mathcal{C}$-backdoor depth of F is at least d.

This algorithm takes at most $\mathcal{O}(\|F\|)$ time per move.

Proof. We will prove this lemma by induction over i. Our splitter-algorithm will try to construct an obstruction tree of depth i by first using the induction hypothesis to build two obstruction trees T_1 and T_2 of depth $i-1$ and then joining them together. After the construction of the first tree T_1, we reach a position J_1 and by our induction hypothesis no variable in $var(J_1)$ occurs in T_1 or α-occurs in a clause of T_1.

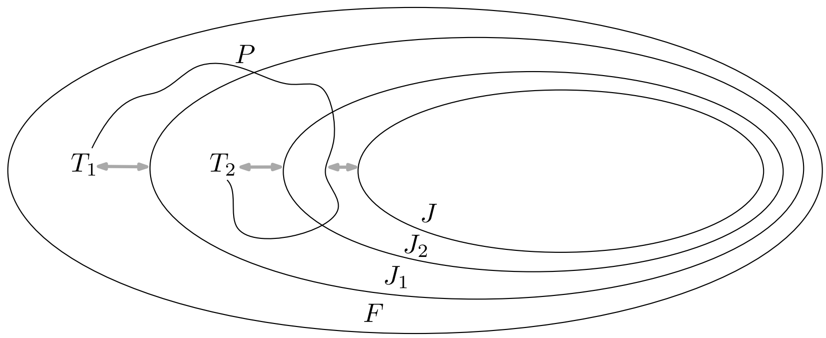

Fig. 9. Overview of the construction in Lemma 18. First, T_1 is chosen in F, yielding J_1. Then, T_2 is chosen in J_1, yielding J_2. In the end the connecting path P is chosen yielding J. A gray doublesided arrow between a position $\widehat{J}$ and structure $\widehat{T}$ symbolizes that no variable $v \in var(\widehat{J})$ occurs in a clause of $\widehat{T}$.

This encapsulates the core idea behind our approach, as it means that T_1 is separated from all potential future obstruction trees T_2 that we build from position J_1. Therefore, we can compute the next tree T_2 in J_1 and join T_1 and T_2 together in accordance with Definition 6 by a path P. At last, we use Corollary 17 to also separate this path from all future obstructions. If at any point of this process we reach a winning position or a proof that the $\mathcal{C}$-backdoor depth of F is at least d, we can stop. Let us now describe this approach in detail.

For convenience, let $x = (3s+d+1)\big((6s+6d+19)^d(80+7d)\big)^{2^d}$. We start our induction with $i = 0$. If there is no $\mathcal{C}$-bad clause in F, then it is a winning position and we can stop. Thus, we can assume that F contains at least one $\mathcal{C}$-bad clause c. If c contains more than $d+s$ α-literals, we have a proof that the $\mathcal{C}$-backdoor depth of F is at least d and we achieve case 3) of the lemma. Otherwise, c contains at most $d+s$ variables that α-occur in c and the splitter can now play by choosing those variables one-by-one to reach a position that satisfies 2) of the lemma after at most $d+s \le 2^i x$ rounds in time $\mathcal{O}(|F|)$.

We now assume the statement of this lemma to hold for $i-1$ and we show it also holds for i. To this end, we start playing the game $\textsc{Game}(F, \mathcal{C})$ according to the existing splitter-algorithm for $i-1$. If we reach (within at most $(2^{i-1}-1)x$ rounds) a winning position or a proof that the $\mathcal{C}$-backdoor depth of F is at least d then we are done. Assuming this is not the case, we reach a position J_1 and a $\mathcal{C}$-obstruction tree T_1 of depth $i-1$ in F such that no variable $v \in var(J_1)$ occurs in T_1 or α-occurs in a clause of T_1.

We continue playing the game at position J_1 according to the existing splitter-algorithm for $\textsc{Game}(J_1, \mathcal{C})$ and $i-1$. The $\mathcal{C}$-backdoor depth of F is larger or equal to the $\mathcal{C}$-backdoor depth of J_1. Thus again (after at most $(2^{i-1}-1)x$ rounds) we either are done (because we reach a winning position or can conclude that the $\mathcal{C}$-backdoor depth of J_1 is at least d) or we reach a position J_2 and a $\mathcal{C}$-obstruction tree T_2 of depth $i-1$ in J_1 such that no variable $v \in var(J_2)$ occurs in T_2 or α-occurs in a clause of T_2.

We pick two clauses $c_1 \in T_1$ and $c_2 \in T_2$ that are $\mathcal{C}$-bad in F and compute a shortest path P between c_1 and c_2 in F. We now argue that $T = T_1 \cup T_2 \cup var(P) \cup P$ is a $\mathcal{C}$-obstruction tree of depth i in F. Let $\beta = \tau_{J_1}$ be the assignment that assigns all the variables the splitter chose until reaching position J_1 to the value given by the connector. Note that J_1 is a connected component of $F[\beta]$. Since all variables and clauses belonging to T_2 induce a connected subgraph of J_1, T_2 is a $\mathcal{C}$-obstruction tree of depth $i-1$ not only in J_1, but also in $F[\beta]$. Let $v \in var(F[\beta])$. We show that v if v is contained in a clause of T_2, then v neither occurs in T_1 nor α-occurs in a clause of T_1. To this end, assume v is contained in a clause of T_2. Since all clauses of T_2 are in J_1 and J_1 is a connected component of $F[\beta]$, we further have $v \in var(J_1)$. On the other hand (as discussed earlier), no variable $v \in var(J_1)$ is contained in T_1 or α-occurs in a clause of T_1. Therefore, by Definition 6, $T = T_1 \cup T_2 \cup var(P) \cup P$ is a $\mathcal{C}$-obstruction tree of depth i in F.

We use Corollary 17 to continue playing the game at position J_2. Again, if we reach a winning position or a proof that the $\mathcal{C}$-backdoor depth of F is at least d we are done. So we focus on the third case that we reach (within at most x rounds) a position J such that no variable $v \in var(J)$ is contained in P or α-occurs in a clause of P.

We know already that no variable $v \in var(J_1)$ is contained in T_1 or α-occurs in a clause of T_1 and that no variable $v \in var(J_2)$ is contained in T_2 or α occurs in a clause of T_2. Since $var(J) \subseteq var(J_2) \subseteq var(J_1)$, and $T = T_1 \cup T_2 \cup var(P) \cup P$, we can conclude that no variable $v \in var(J)$ is contained in T or α-occurs in a clause of T.

In total, we played for $2^{i-1}x + 2^{i-1}x + x = 2^i x$ rounds. The splitter-algorithm in Corollary 17 takes at most $\mathcal{O}(\|F\|)$ time per move. The same holds for the splitter-algorithm for $i-1$ that we use as a subroutine. Thus, the whole algorithm takes at most $\mathcal{O}(\|F\|)$ time per move. □

The main results now follow easily by combining Lemmas 1, 5, 7 and 18.

Theorem 19. *Let $\mathcal{C} = \mathcal{C}_{\alpha,s}$ with $\alpha \subseteq \{+,-\}$, $\alpha \neq \emptyset$, and $s \in \mathbb{N}$. We can, for a given $F \in \mathcal{CNF}$ and a non-negative integer d, in time at most $2^{2^{2^{\mathcal{O}(d)}}}\|F\|$ either*

1) *compute a component $\mathcal{C}$-backdoor tree of F of depth at most $2^{2^{\mathcal{O}(d)}}$, or*
2) *conclude that the $\mathcal{C}$-backdoor depth of F is larger than d.*

Proof. We apply Lemma 18 with its parameters i and d both set to $d+1$. An obstruction tree of depth $d+1$ is, according to Lemma 7, a proof that the backdoor depth is at least $d+1$, thus the output of the splitter-algorithm in Lemma 18 after $2^{2^{\mathcal{O}(d)}}$ rounds reduces to either a winning position, or a proof that the $\mathcal{C}$-backdoor depth of F is at least $d+1$. The algorithm takes at most $\mathcal{O}(\|F\|)$ time per move. The statement then follows from Lemma 5. □

Corollary 20. *Let $\mathcal{C} \in \{\text{Horn}, \text{dHorn}, \text{Krom}\}$. The* CnfSat *problem can be solved in linear time for any class of formulas of bounded $\mathcal{C}$-backdoor depth.*

Proof. Let $F \in \mathcal{CNF}$. We use Theorem 19 to compute a component $\mathcal{C}$-backdoor tree for F of depth at most $2^{2^{\mathcal{O}(d)}}$ and then use Lemma 1 to decide the satisfiability of F in time $2^{2^{2^{\mathcal{O}(d)}}}\|F\|$. □

8. Comparison with other approaches

In this section, we compare the generality of backdoor depth with other parameters that admit a fixed-parameter tractable solution of the problem, as we have listed in the introduction. Our comparison is based on the concept of domination [31]. For two integer-valued parameters p and q, we say that p *dominates* q if every class of instances for which q is bounded, also p is bounded; p *strictly dominates* q if p dominates q but q does not dominate p. If p and q dominate each other, they are *domination equivalent*. If neither of them dominates the other, they are *domination orthogonal*.

We first define two sequences of formulas that we will use for several separation results below. For any integer $d \geq 1$, the CNF formula Q_d, consists of $n = 3 \cdot 2^d - 2$ clauses $c_1, \ldots, c_n$ over the variables $x_0, \ldots, x_n$ and $y_1, \ldots, y_n$, where $c_i = \{x_{i-1}, \neg y_i, x_i\}$. The formula Q'_d is defined similarly, except that the y_i variables are omitted, and hence $c_i = \{x_{i-1}, x_i\}$. By construction, we have $\mathsf{Q}_d \in \text{dHorn} \setminus (\text{Horn} \cup \text{Krom})$ and $\mathsf{Q}'_d \in \text{Krom} \cap \text{dHorn} \setminus \text{Horn}$.

Lemma 21. $\text{depth}_{\text{Horn}}(\mathsf{Q}_d) = \text{depth}_{\text{Krom}}(\mathsf{Q}_d) = \text{depth}_{\text{Horn}}(\mathsf{Q}'_d) = d$.

Proof. We focus on showing that $\text{depth}_{\mathcal{C}}(\mathsf{Q}_d) = d$ for $\mathcal{C} \in \{\text{Horn}, \text{Krom}\}$, the proof for $\text{depth}_{\text{Horn}}(\mathsf{Q}'_d) = d$ is similar. We proceed by induction on d. Since $\mathsf{Q}_1 = \{c_0\}$, the induction basis holds. Now assume $d > 1$. Since Q_d is connected, the root r of any component $\mathcal{C}$-backdoor tree for it is a variable node. Assume r branches on variable x_i. By the symmetry of Q_n, we can assume, w.l.o.g., that $i \geq n/2$. We observe that $\mathsf{Q}_d[x_i = 1] = \mathsf{Q}_d \setminus \{c_i, c_{i+1}\}$, hence $\mathsf{Q}_{d-1} \subseteq \mathsf{Q}_d[x_i = 1]$, where Q_{d-1} has $3 \cdot 2^{d-1} - 2$ clauses. If r branches on y_i, then $\mathsf{Q}_d[x_i] \subseteq \mathsf{Q}_d[y_i = 0]$, and so $\mathsf{Q}_{d-1} \subseteq \mathsf{Q}_d[y_i = 0]$ as well. Consequently, in any case, at least one child of r is labeled with a formula that contains Q_{d-1}. By induction hypothesis $\text{depth}_{\mathcal{C}}(\mathsf{Q}_{d-1}) = d - 1$. Thus $\text{depth}_{\mathcal{C}}(\mathsf{Q}_d) = d$ as claimed. □

8.1. *Backdoor size into* Horn, dHorn, *and* Krom

Let $\mathcal{C} \subseteq \mathcal{CNF}$. The *$\mathcal{C}$-backdoor size* of a CNF formula F, denoted $\text{size}_{\mathcal{C}}(F)$, is the size of a smallest (strong) $\mathcal{C}$-backdoor of F. Nishimura et al. [26] have shown that CnfSat is fixed-parameter tractable parameterized by $\mathcal{C}$-backdoor size for $\mathcal{C} \in \{\text{Horn}, \text{dHorn}, \text{Krom}\}$. A class $\mathcal{C}$ of CNF formulas is *nontrivial* if $\mathcal{C} \neq \mathcal{CNF}$.

Proposition 22. *For every nontrivial class $\mathcal{C}$ of CNF formulas,* $\text{depth}_{\mathcal{C}}$ *strictly dominates* $\text{size}_{\mathcal{C}}$.

Proof. Clearly, $\text{depth}_{\mathcal{C}}$ dominates $\text{size}_{\mathcal{C}}$, since if we have a $\mathcal{C}$-backdoor B of some $F \in \mathcal{CNF}$, we can build a backdoor tree with at most $2^{|B|}$ variable nodes and depth $\leq |B|$.

To show that the domination is strict, let $F \in \mathcal{CNF} \setminus \mathcal{C}$ (it exists, since $\mathcal{C}$ is nontrivial) and let $F_1, \ldots, F_n$ be variable-disjoint copies of F. Since $F \notin \mathcal{C}$, we have $\text{size}_{\mathcal{C}}(F) \geq \text{depth}_{\mathcal{C}}(F) > 0$. Let $\text{size}_{\mathcal{C}}(F) = s$ and $\text{depth}_{\mathcal{C}}(F) = d$. Now $\text{size}_{\mathcal{C}}(F_n) = ns$, so we can choose n to make $\text{size}_{\mathcal{C}}(F_n)$ arbitrarily large. However, $\text{depth}_{\mathcal{C}}(F_n) = d$ remains bounded, since in a backdoor tree we can use a component node at the root which branches into (at least) n components, each of $\mathcal{C}$-backdoor depth at most d. □

8.2. *Number of leaves of backdoor trees into* Horn, dHorn, *and* Krom

Backdoor trees, introduced by Samer and Szeider [29,27], are a special case of component backdoor trees as defined in Section 3. A *$\mathcal{C}$-backdoor tree* for $F \in \mathcal{CNF}$ is a component $\mathcal{C}$-backdoor tree for F without component nodes. For a base class $\mathcal{C}$ and $F \in \mathcal{CNF}$, let $\text{leaves}_{\mathcal{C}}(F)$ denote the smallest number of leaves of any $\mathcal{C}$-backdoor tree (recall the definition in Section 3). CnfSat is fixed-parameter tractable parameterized by $\text{leaves}_{\mathcal{C}}(F)$ [27]. Since $\text{size}_{\mathcal{C}}(F) + 1 \leq \text{leaves}_{\mathcal{C}}(F) \leq 2^{\text{size}_{\mathcal{C}}(F)}$, $\text{leaves}_{\mathcal{C}}$ and $\text{size}_{\mathcal{C}}$ are domination equivalent [29]. Hence we have the following corollary to Proposition 22.

Corollary 23. *For every nontrivial class $\mathcal{C}$ of CNF formulas,* $\mathrm{depth}_{\mathcal{C}}$ *strictly dominates* $\mathrm{leaves}_{\mathcal{C}}$.

8.3. Backdoor depth into Null

Mählmann et al. [22] considered the base class $\textsc{Null} = \{\emptyset, \{\emptyset\}\}$ and showed that CnfSat is fixed-parameter tractable parameterized by $\mathrm{depth}_{\textsc{Null}}$.

Proposition 24. $\mathrm{depth}_{\mathcal{C}}$ *strictly dominates* $\mathrm{depth}_{\textsc{Null}}$ *for* $\mathcal{C} \in \{\textsc{Horn}, \textsc{dHorn}, \textsc{Krom}\}$.

Proof. It suffices to consider $\mathcal{C} \in \{\textsc{dHorn}, \textsc{Krom}\}$, since the cases dHorn and Horn are symmetric. Since $\textsc{Null} \subseteq \textsc{dHorn} \cap \textsc{Krom}$, $\mathrm{depth}_{\mathcal{C}}$ dominates $\mathrm{depth}_{\textsc{Null}}$. To show that the domination is strict, consider the CNF formulas Q_d and Q'_d from above, with $\mathrm{depth}_{\textsc{Horn}}(\mathsf{Q}_d) = \mathrm{depth}_{\textsc{Horn}}(\mathsf{Q}'_d) = d$ by Proposition 21. Since $\textsc{Null} \subseteq \textsc{Horn}$, $\mathrm{depth}_{\textsc{Null}}(\mathsf{Q}_d) = \mathrm{depth}_{\textsc{Null}}(\mathsf{Q}'_d) \geq d$. However, $\mathrm{depth}_{\textsc{dHorn}}(\mathsf{Q}_d) = \mathrm{depth}_{\textsc{Krom}}(\mathsf{Q}'_d) = 0$ since $\mathsf{Q}_d \in \textsc{dHorn}$ and $\mathsf{Q}'_d \in \textsc{Krom}$. □

8.4. Backdoor treewidth into Horn, dHorn, and Krom

Backdoor treewidth is another general parameter for CnfSat and the constraint satisfaction problem (CSP) defined with respect to a base class $\mathcal{C}$ [14,13]. Let F be a CSP instance and $X \subseteq V(F)$ a subset of its variables. The *torso graph* of F with respect to X, denoted $\mathcal{T}_F(X)$, has as vertices the variables in X and contains an edge $\{x, y\}$ if and only if x and y appear together in the scope of a constraint of F or x and y are in the same connected component of the graph obtained from the incidence graph of F after deleting all the variables in X. Let $G = (V, E)$ be a graph. A tree decomposition of G is a pair $(T, \mathcal{X})$, $\mathcal{X} = \{X_t\}_{t\in V(T)}$, where T is a tree and $\mathcal{X}$ is a collection of subsets of V such that: (i) for each edge $\{u, v\} \in E$ there exists a node t of T such that $\{u, v\} \subseteq X_t$, and (ii) for each $v \in V$, the set $\{t \mid v \in X_t\}$ induces in T a nonempty connected subtree. The width of $(T, \mathcal{X})$ is equal to $\max\{|X_t| - 1 \mid t \in V(T)\}$ and the *treewidth* of G is the minimum width over all tree decompositions of G. Let $F \in \mathcal{CNF}$ and X a $\mathcal{C}$-backdoor set of F. The *torso treewidth* of X is the treewidth of the torso graph $\mathcal{T}_F(X)$, and the $\mathcal{C}$-*backdoor treewidth* of F, denoted $\mathrm{bdtw}_{\mathcal{C}}(F)$, is the smallest torso treewidth over all $\mathcal{C}$-backdoor sets of F. It is known that $\mathrm{bdtw}_{\mathcal{C}}$ dominates the treewidth of the formula's primal graph and $\mathrm{size}_{\mathcal{C}}$ [14,13].

Proposition 25. *For* $\mathcal{C} \in \{\textsc{Horn}, \textsc{dHorn}, \textsc{Krom}\}$*, the parameters* $\mathrm{depth}_{\mathcal{C}}$ *and* $\mathrm{bdtw}_{\mathcal{C}}$ *are domination orthogonal.*

Proof. Since the cases dHorn and Horn are symmetric, we may assume that $\mathcal{C} \in \{\textsc{Horn}, \textsc{Krom}\}$.

First we construct a sequence of formulas of constant $\mathrm{depth}_{\mathcal{C}}$ but unbounded $\mathrm{bdtw}_{\mathcal{C}}$ (similar constructions have been used before [18,22]). For any $n \geq 2$ we construct a CNF formula F_n based on an $n \times n$ grid graph G_n whose edges are oriented arbitrarily. We take a special variable x. For each vertex $v \in V(G_n)$ we introduce three variables y_v^i, $1 \leq i \leq 3$, and for each oriented edge $(u, v) \in E(G_n)$ we introduce a variable $z_{u,v}$ and the clauses $c_{u,v} = \{x, y_u^1, y_u^2, y_u^3, z_{u,v}\}$ and $d_{u,v} = \{\neg x, \neg y_v^1, \neg y_v^2, \neg y_v^3, \neg z_{u,v}\}$. Let X be a $\mathcal{C}$-backdoor of F_n. We observe that for each $v \in V(G_n)$, X must contain at least one of the variables y_v^1, y_v^2, y_v^3 since, otherwise, $F[X \mapsto 0]$ would contain a clause (a subset of $c_{u,v}$) that is neither Horn nor Krom. W.l.o.g., we assume $y_v^1 \in X$ for all $v \in V(G_n)$. Furthermore, for each edge $(u, v) \in E(G_n)$, the torso graph $\mathcal{T}_F(X)$ will contain the edge $\{y_u^1, y_v^1\}$ or the edges $\{y_u^1 z_{u,v}\}$ and $\{z_{u,v}, y_v^1\}$. Consequently, $\mathcal{T}_F(X)$ has a subgraph that is isomorphic to a subdivision of G_n. Since the treewidth of G_n is n, and subdividing edges does not change the treewidth, we conclude that $\mathrm{bdtw}_{\mathcal{C}}(F_n) \geq n$.

It remains to show that $\mathrm{depth}_{\mathcal{C}}$ does not dominate $\mathrm{bdtw}_{\mathcal{C}}$. Consider again the formula Q_d from above with $\mathrm{depth}_{\mathcal{C}}(\mathsf{Q}_d) = d$ by Lemma 21. The set $X = \{x_0, \dots, x_n\}$, $n = 3 \cdot 2^d - 2$, is a $\mathcal{C}$-backdoor of F_d. The torso graph $\mathcal{T}_X(F_d)$ is a path of length n, which has treewidth 1. Thus $\mathrm{bdtw}_{\mathcal{C}}(F_d) = 1$. □

8.5. Backdoor size into heterogeneous base classes based on Horn, dHorn, and Krom

Consider the possibility that different assignments to the backdoor variables move the formula into different base classes, e.g., for $B \subseteq var(F)$ and $\tau, \tau' \to \{0, 1\}$, we have $F[\tau] \in \textsc{Horn}$ and $F[\tau'] \in \textsc{Krom}$. We can see such a backdoor B as a $\mathcal{C}$-backdoor for $\mathcal{C} = \textsc{Horn} \cup \textsc{Krom}$. Gaspers et al. [16] have shown that $\mathrm{depth}_{\textsc{Horn}\cup\textsc{Krom}}$ strictly dominates $\mathrm{depth}_{\textsc{Horn}}$ and $\mathrm{depth}_{\textsc{Krom}}$. They also showed that computing $\mathrm{depth}_{\textsc{Horn}\cup\textsc{Krom}}$ is FPT, but computing $\mathrm{depth}_{\textsc{Horn}\cup\textsc{dHorn}}$ is W[2]-hard (i.e., unlikely to be FPT).

Proposition 26. *For any two different classes* $\mathcal{C} \neq \mathcal{C}' \in \{\textsc{Horn}, \textsc{dHorn}, \textsc{Krom}\}$, $\mathrm{size}_{\mathcal{C}\cup\mathcal{C}'}$ *and* $\mathrm{depth}_{\mathcal{C}}$ *are domination orthogonal.*

Proof. Let $\mathcal{C} \neq \mathcal{C}' \in \{\textsc{Horn}, \textsc{dHorn}, \textsc{Krom}\}$.

With the same padding argument as used in the proof of Proposition 22, we can show that there are formulas of constant $\mathrm{depth}_{\mathcal{C}}$ and arbitrarily large $\mathrm{size}_{\mathcal{C}\cup\mathcal{C}'}$.

For the converse direction, we utilize the formulas Q and Q' from above, as well as the formulas $\overline{\mathsf{Q}}$ and $\overline{\mathsf{Q}'}$ obtained from Q and Q', respectively, by flipping all literals to the opposite polarity. For any pair $\mathcal{C}, \mathcal{C}'$, we can choose formulas

$F_d, F'_d \in \{\mathsf{Q}, \mathsf{Q}', \overline{\mathsf{Q}}, \overline{\mathsf{Q}'}\}$, such that $F_d \in \mathcal{C} \setminus \mathcal{C}'$ and $F'_d \in \mathcal{C}' \setminus \mathcal{C}$. By Lemma 21 (applied directly or via symmetry) we have $\mathrm{depth}_{\mathcal{C}'}(F_d) = \mathrm{depth}_{\mathcal{C}}(F'_d) = d$. We may assume that F_d and F'_d have disjoint sets of variables. We pick a new variable z and add it positively to all clauses of F_d, and negatively to all clauses of F'_d, obtaining the formulas G_d and G'_d, respectively. Let $G^*_d = G_d \cup G'_d$. On the one hand, $\mathrm{depth}_{\mathcal{C}}(G^*_d) \geq d$, since any component $\mathcal{C}$-backdoor tree of G^*_d must contain a subtree which induces a $\mathcal{C}$-backdoor tree of F'_d, whose depth is $\geq \mathrm{depth}_{\mathcal{C}}(F'_d) = d$. On the other hand, $\mathrm{size}_{\mathcal{C}\cup\mathcal{C}'}(G^*) = 1$, since $G^*_d[z=0] = F_d \in \mathcal{C} \subseteq \mathcal{C} \cup \mathcal{C}'$ and $G^*_d[z=1] = F'_d \in \mathcal{C}' \subseteq \mathcal{C} \cup \mathcal{C}'$. □

8.6. *Backdoor size into scattered base classes based on* Horn, dHorn, *and* Krom

For base classes $\mathcal{C}_1, \ldots, \mathcal{C}_r \subseteq \mathcal{CNF}$, let $\mathcal{C}_1 \oplus \cdots \oplus \mathcal{C}_r \subseteq \mathcal{CNF}$ denote the class of CNF formulas F with the property that each $F' \in \mathrm{Conn}(F)$ belongs to $\mathcal{C}_i$ for some $i \in \{1, \ldots, r\}$. For example, if $F \in \textsc{Horn} \oplus \textsc{Krom}$, then each connected component of F is either Horn or Krom. Such *scattered* base classes where introduced by Ganian et al. [15] for constraint satisfaction, but the concept naturally extends to CnfSat.

Proposition 27. *For any two different classes $\mathcal{C} \neq \mathcal{C}' \in \{\textsc{Horn}, \textsc{dHorn}, \textsc{Krom}\}$, $\mathrm{size}_{\mathcal{C}\oplus\mathcal{C}'}$ and $\mathrm{depth}_{\mathcal{C}}$ are domination orthogonal.*

Proof. Let $\mathcal{C} \neq \mathcal{C}' \in \{\textsc{Horn}, \textsc{dHorn}, \textsc{Krom}\}$. Again, we can use the padding argument from the proof of Proposition 22 to show that there are formulas of constant $\mathrm{depth}_{\mathcal{C}}$ and arbitrarily large $\mathrm{size}_{\mathcal{C}\oplus\mathcal{C}'}$. For the converse direction, we argue as in the proof of Proposition 27 that there are variable-disjoint formulas $F_d \in \mathcal{C} \setminus \mathcal{C}'$ and $F'_d \in \mathcal{C}' \setminus \mathcal{C}$ with $\mathrm{depth}_{\mathcal{C}'}(F_d) = \mathrm{depth}_{\mathcal{C}}(F'_d) = d$. For $F^*_d = F_d \cup F'_d$ we have $\mathrm{depth}_{\mathcal{C}}(F^*_d) \geq d$ but $\mathrm{size}_{\mathcal{C}\oplus\mathcal{C}'}(F^*) = 0$ since $F^*_d \in \mathcal{C} \oplus \mathcal{C}'$. □

8.7. *Deletion backdoor size into* Q-Horn

A CNF formula F is *quadratic Horn* if there exists a mapping $f : var(F) \to [0,1]$ such that for each clause $c \in F$ we have $\sum_{x \in C \cap var(C)} f(x) + \sum_{\neg x \in C \setminus var(C)} 1 - f(x) \leq 1$. The class Q-Horn of all quadratic Horn formulas properly contains Horn, dHorn, and Krom (which can be seen by taking f to be the constant mapping to 1, to 0, and to 1/2, respectively). Satisfiability and recognition of quadratic Horn formulas can be decided in polynomial time [3,4]. Gaspers et al. [17] considered *deletion* Q-Horn-backdoors, since computing $\mathrm{size}_{\textsc{Q-Horn}}$ is W[2]-hard. A deletion Q-Horn-backdoor of $F \in \mathcal{CNF}$ is a set $B \subseteq var(F)$ such that $F - B := \{c \setminus (B \cup \overline{B}) \mid c \in F\} \in \textsc{Q-Horn}$. We denote the size of a smallest deletion Q-Horn-backdoor of F, i.e., the deletion Q-Horn-backdoor size of F, by $\mathrm{del}_{\textsc{Q-Horn}}(F)$. Ramanujan and Saurabh [28] showed that computing $\mathrm{del}_{\textsc{Q-Horn}}(F)$ is fixed-parameter tractable, improving upon an FPT-approximation result by Gaspers et al. [17]. Every deletion Q-Horn-backdoor is a Q-Horn-backdoor, but the converse does not hold. Hence $\mathrm{size}_{\textsc{Q-Horn}}$ strictly dominates $\mathrm{del}_{\textsc{Q-Horn}}$.

Proposition 28. *For $\mathcal{C} \in \{\textsc{Horn}, \textsc{dHorn}, \textsc{Krom}\}$, the parameters $\mathrm{depth}_{\mathcal{C}}$ and $\mathrm{del}_{\textsc{Q-Horn}}$ are domination orthogonal.*

Proof. The padding argument from the proof of Proposition 22 shows that there are formulas of constant $\mathrm{depth}_{\mathcal{C}}$ and arbitrarily large $\mathrm{size}_{\textsc{Q-Horn}}$, and therefore of arbitrarily large $\mathrm{del}_{\textsc{Q-Horn}}$.

For the reverse direction, consider the formula Q_d. By Lemma 21, $\mathrm{depth}_{\mathcal{C}}(\mathsf{Q}_d) = d$ for $\mathcal{C} \in \{\textsc{Horn}, \textsc{Krom}\}$. By construction, $\mathsf{Q}_d \in \textsc{dHorn} \subseteq \textsc{Q-Horn}$, thus $\mathrm{del}_{\textsc{Q-Horn}}(\mathsf{Q}_d) = 0$. Hence neither $\mathrm{depth}_{\textsc{Horn}}$ nor $\mathrm{depth}_{\textsc{Krom}}$ dominates $\mathrm{del}_{\textsc{Q-Horn}}$. A symmetric argument shows that $\mathrm{depth}_{\textsc{dHorn}}$ does not dominate $\mathrm{del}_{\textsc{Q-Horn}}$. □

8.8. *Backdoor size into bounded incidence treewidth*

The *incidence treewidth* of a formula F is the treewidth of its incidence graph. Let $\mathcal{W}_t$ denote the class of all CNF formulas of incidence treewidth $\leq t$. CnfSat can be solved in polynomial time for formulas in $\mathcal{W}_t$ [30,33]. Gaspers and Szeider [18] showed that for every constant $t \geq 1$, $\mathrm{size}_{\mathcal{W}_t}$ can be FPT approximated. Hence CnfSat is FPT parameterized by $\mathrm{size}_{\mathcal{W}_t}$.

Proposition 29. *For $\mathcal{C} \in \{\textsc{Horn}, \textsc{dHorn}, \textsc{Krom}\}$ and every $t \geq 1$, the parameters $\mathrm{depth}_{\mathcal{C}}$ and $\mathrm{size}_{\mathcal{W}_t}$ are domination orthogonal.*

Proof. Let $t \geq 1$ be a constant and $\mathcal{C} \in \{\textsc{Horn}, \textsc{Krom}\}$; the cases dHorn and Horn are symmetric.

Consider the $(t+1) \times (t+1)$ grid graph G_{t+1}. We construct the CNF formula T by introducing a variable x_v for every $v \in V(G_{t+1})$ and a clause $\{\neg x_u, \neg x_v\}$ for every edge $uv \in E(G_{t+1})$. Let F_n be the formula consisting of n variable-disjoint copies of T. By construction, $F_n \in \mathcal{C}$, hence $\mathrm{depth}_{\mathcal{C}}(F_n) = 0$. The incidence graph of T contains G_{t+1} as a minor, hence T's incidence treewidth is $\geq t+1$. Consequently, any $\mathcal{W}_t$-backdoor of F_n must contain at least one variable from each of the n copies of F_{t+1}. Thus $\mathrm{size}_{\mathcal{W}_t}(F_n) \geq n$. It follows that $\mathrm{size}_{\mathcal{W}_t}$ does not dominate $\mathrm{depth}_{\mathcal{C}}$.

For the converse direction, consider again the formula Q_d with $\mathrm{depth}_{\mathcal{C}}(\mathsf{Q}_d) = d$ (Lemma 21). The incidence treewidth of Q_d is 2, hence $\mathrm{size}_{\mathcal{W}_t}(\mathsf{Q}_t) = 0$ for $t \geq 2$. Consequently, $\mathrm{depth}_{\mathcal{C}}$ does not dominate $\mathrm{size}_{\mathcal{W}_t}$. □

9. Conclusion

We showed that CnfSat can be solved in linear-time for formulas of bounded $\mathcal{C}$-backdoor depth whenever $\mathcal{C}$ is any of the well-known Schaefer classes. We achieved this by showing that $\mathcal{C}$-backdoor depth can be FPT-approximated for any class $\mathcal{C} = \mathcal{C}_{\alpha,s}$. This allowed us to extend the results of Mählmann et al. [22] for the class of variable-free formulas to all Schaefer classes. Our results provide an important milestone towards generalizing and unifying the various tractability results based on variants of $\mathcal{C}$-backdoor size (see also future work below) to the only recently introduced and significantly more powerful $\mathcal{C}$-backdoor depth.

Let us finish with some natural and potentially significant extensions of backdoor depth that can benefit from our approach based on separator obstructions. Two of the probably most promising ones that have already been successfully employed as extensions of backdoor size are the so-called *scattered* and *heterogeneous* backdoor sets [16,15]; also refer to Sections 8.5 and 8.6. Interestingly, while those two notions lead to orthogonal tractable classes in the context of backdoor size, they lead to the same tractable class for backdoor depth. Therefore, lifting these two extensions to backdoor depth, would result in a unified and significantly more general approach. While we are hopeful that our techniques can be adapted to this setting, one of the main remaining obstacles is that obstructions of depth 0 no longer are single (bad) clauses. For instance, consider the heterogeneous class $\mathcal{C} = \text{Horn} \cup \text{Krom}$. Here, a CNF formula may not be in $\mathcal{C}$ due to a pair of clauses, one in $\text{Horn} \setminus \text{Krom}$ and another one in $\text{Krom} \setminus \text{Horn}$. Finally, an even more general but also more challenging tractable class to consider for backdoor depth is the class of Q-Horn formulas (see Section 8.7), which generalizes the heterogeneous class obtained as the union of all considered Schaefer classes.

CRediT authorship contribution statement

Jan Dreier: Writing – review & editing, Writing – original draft, Visualization, Methodology, Investigation, Conceptualization. **Sebastian Ordyniak:** Writing – review & editing, Writing – original draft, Visualization, Methodology, Investigation, Conceptualization. **Stefan Szeider:** Writing – review & editing, Writing – original draft, Visualization, Methodology, Investigation, Conceptualization.

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

No data was used for the research described in the article.

Acknowledgments

Sebastian Ordyniak: Supported by the Engineering and Physical Sciences Research Council (EPSRC, project EP/V00252X/1). Stefan Szeider: supported by the Austrian Science Fund (FWF, projects 32441 and 36688) and the Vienna Science and Technology Fund (WWTF, project ICT19-065).

This research was funded in whole or in part by the Austrian Science Fund (FWF) DOI10.55776/P36420 and DOI10.55776/P36688. For open access purposes, the author has applied a CC BY public copyright license to any author accepted manuscript version arising from this submission.

References

[1] B. Aspvall, M.F. Plass, R.E. Tarjan, A linear-time algorithm for testing the truth of certain quantified Boolean formulas, Inf. Process. Lett. 8 (3) (1979) 121–123.
[2] F. Bacchus, S. Dalmao, T. Pitassi, Algorithms and complexity results for #SAT and Bayesian inference, in: 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'03), 2003, pp. 340–351.
[3] E. Boros, Y. Crama, P.L. Hammer, Polynomial-time inference of all valid implications for horn and related formulae, Ann. Math. Artif. Intell. 1 (1990) 21–32.
[4] E. Boros, P.L. Hammer, X. Sun, Recognition of q-Horn formulae in linear time, Discrete Appl. Math. 55 (1) (1994) 1–13.
[5] J. Bulian, A. Dawar, Graph isomorphism parameterized by elimination distance to bounded degree, Algorithmica 75 (2) (2016) 363–382.
[6] S.A. Cook, The complexity of theorem-proving procedures, in: Proc. 3rd Annual Symp. on Theory of Computing, Shaker Heights, Ohio, 1971, pp. 151–158.
[7] W.F. Dowling, J.H. Gallier, Linear-time algorithms for testing the satisfiability of propositional Horn formulae, J. Log. Program. 1 (3) (1984) 267–284.
[8] J. Dreier, S. Ordyniak, S. Szeider, SAT backdoors: depth beats size, in: S. Chechik, G. Navarro, E. Rotenberg, G. Herman (Eds.), 30th Annual European Symposium on Algorithms (ESA 2022), in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 244, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2022, pp. 46:1–46:18.
[9] J. Dreier, S. Ordyniak, S. Szeider, CSP beyond tractable constraint languages, Constraints 28 (3) (2023) 450–471.
[10] J.K. Fichte, D. Le Berre, M. Hecher, S. Szeider, The silent (r)evolution of SAT, Commun. ACM 66 (6) (2023) 64–72.

[11] F.V. Fomin, P.A. Golovach, D.M. Thilikos, Parameterized complexity of elimination distance to first-order logic properties, arXiv preprint, arXiv:2104.02998, 2021.
[12] V. Ganesh, M.Y. Vardi, On the unreasonable effectiveness of SAT solvers, in: T. Roughgarden (Ed.), Beyond the Worst-Case Analysis of Algorithms, Cambridge University Press, 2020, pp. 547–566.
[13] R. Ganian, M.S. Ramanujan, S. Szeider, Backdoor treewidth for SAT, in: S. Gaspers, T. Walsh (Eds.), Theory and Applications of Satisfiability Testing - SAT 2017 - 20th International Conference, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings, in: Lecture Notes in Computer Science, vol. 10491, Springer Verlag, 2017, pp. 20–37.
[14] R. Ganian, M.S. Ramanujan, S. Szeider, Combining treewidth and backdoors for CSP, in: H. Vollmer, B. Vallée (Eds.), 34th Symposium on Theoretical Aspects of Computer Science (STACS 2017), in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 66, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017, pp. 36:1–36:17.
[15] 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. Full version of a SODA'16 paper.
[16] 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.
[17] S. Gaspers, S. Ordyniak, M.S. Ramanujan, S. Saurabh, S. Szeider, Backdoors to q-Horn, Algorithmica 74 (1) (2016) 540–557.
[18] S. Gaspers, S. Szeider, Strong backdoors to bounded treewidth SAT, in: 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26–29 October 2013, Berkeley, CA, USA, IEEE Computer Society, 2013, pp. 489–498.
[19] C.P. Gomes, H. Kautz, A. Sabharwal, B. Selman, Satisfiability solvers, in: Handbook of Knowledge Representation, in: Foundations of Artificial Intelligence, vol. 3, Elsevier, 2008, pp. 89–134.
[20] M. Grohe, S. Kreutzer, S. Siebertz, Deciding first-order properties of nowhere dense graphs, J. ACM 64 (3) (2017) 17:1–17:32.
[21] L. Levin, Universal sequential search problems, Probl. Inf. Transm. 9 (3) (1973) 265–266.
[22] N. Mählmann, S. Siebertz, A. Vigny, Recursive backdoors for SAT, in: F. Bonchi, S.J. Puglisi (Eds.), 46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021, August 23-27, 2021, Tallinn, Estonia, in: LIPIcs, vol. 202, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 73:1–73:18.
[23] D. Marx, Parameterized complexity and approximation algorithms, Comput. J. 51 (1) (2008) 60–78.
[24] J. Nesetril, P.O. de Mendez, Tree-depth, subgraph coloring and homomorphism bounds, Eur. J. Comb. 27 (6) (2006) 1022–1041.
[25] J. Nešetřil, P.O. de Mendez, Sparsity - Graphs, Structures, and Algorithms, Algorithms and Combinatorics, vol. 28, Springer, 2012.
[26] N. Nishimura, P. Ragde, S. Szeider, Detecting backdoor sets with respect to Horn and binary clauses, in: Proceedings of SAT 2004, Seventh International Conference on Theory and Applications of Satisfiability Testing, 10–13 May, 2004, Vancouver, BC, Canada, 2004, pp. 96–103.
[27] S. Ordyniak, A. Schidler, S. Szeider, Backdoor DNFs, in: Z. Zhou (Ed.), Proceeding of IJCAI-2021, the 30th International Joint Conference on Artificial Intelligence, 2021, pp. 1403–1409.
[28] M.S. Ramanujan, S. Saurabh, Linear-time parameterized algorithms via skew-symmetric multicuts, ACM Trans. Algorithms 13 (4) (2017) 46:1–46:25.
[29] M. Samer, S. Szeider, Backdoor trees, in: AAAI 08, Twenty-Third Conference on Artificial Intelligence, Chicago, Illinois, July 13–17, 2008, AAAI Press, 2008, pp. 363–368.
[30] M. Samer, S. Szeider, Algorithms for propositional model counting, J. Discret. Algorithms 8 (1) (2010) 50–64.
[31] M. Samer, S. Szeider, Constraint satisfaction with bounded treewidth revisited, J. Comput. Syst. Sci. 76 (2) (2010) 103–114.
[32] T.J. Schaefer, The complexity of satisfiability problems, in: Conference Record of the Tenth Annual ACM Symposium on Theory of Computing, San Diego, Calif., 1978, ACM, 1978, pp. 216–226.
[33] S. Szeider, On fixed-parameter tractable parameterizations of SAT, in: E. Giunchiglia, A. Tacchella (Eds.), Theory and Applications of Satisfiability, 6th International Conference, SAT 2003, Selected and Revised Papers, in: Lecture Notes in Computer Science, vol. 2919, Springer Verlag, 2004, pp. 188–202.
[34] R. Williams, C. Gomes, B. Selman, Backdoors to typical case complexity, in: G. Gottlob, T. Walsh (Eds.), Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, IJCAI 2003, Morgan Kaufmann, 2003, pp. 1173–1178.