Backdoors to Acyclic SAT [*]

Serge Gaspers   Stefan Szeider

Institute of Information Systems
Vienna University of Technology
Vienna, Austria.
gaspers@kr.tuwien.ac.at
stefan@szeider.net

Abstract

Backdoor sets, a notion introduced by Williams *et al.* in 2003, are certain sets of key variables of a CNF formula F that make it easy to solve the formula; by assigning truth values to the variables in a backdoor set, the formula gets reduced to one or several polynomial-time solvable formulas. More specifically, a *weak backdoor set* of F is a set X of variables such that there exits a truth assignment τ to X that reduces F to a satisfiable formula $F[\tau]$ that belongs to a polynomial-time decidable base class $\mathcal{C}$. A *strong backdoor set* is a set X of variables such that for all assignments τ to X, the reduced formula $F[\tau]$ belongs to $\mathcal{C}$.

We study the problem of finding backdoor sets of size at most k with respect to the base class of CNF formulas with acyclic incidence graphs, taking k as the parameter. We show that

1. the detection of weak backdoor sets is W[2]-hard in general but fixed-parameter tractable for r-CNF formulas, for any fixed $r \geq 3$, and

2. the detection of strong backdoor sets is fixed-parameter approximable.

Result 1 is the the first positive one for a base class that does not have a characterization with obstructions of bounded size. Result 2 is the first positive one for a base class for which strong backdoor sets are more powerful than deletion backdoor sets.

Not only SAT, but also #SAT can be solved in polynomial time for CNF formulas with acyclic incidence graphs. Hence Result 2 establishes a new structural parameter that makes #SAT fixed-parameter tractable and that is incomparable with known parameters such as treewidth and cliquewidth.

We obtain the algorithms by a combination of an algorithmic version of the Erdős-Pósa Theorem, Courcelle's model checking for monadic second order logic, and new combinatorial results on how disjoint cycles can interact with the backdoor set. These new combinatorial arguments come into play when the incidence graph of F has many vertex-disjoint cycles. As only few of these cycles can vanish by assigning a value to a variable from the cycle, many cycles need to vanish by assigning values to variables that are in clauses of these cycles. These external variables are either so rare or structured that our combinatorial arguments can identify a small set of variables such that any backdoor set of size at most k contains at least one of these variables, or they are so abundant and unstructured that they themselves create cycles in the incidence graph in such a way that F cannot have a backdoor set of size at most k.

Keywords: SAT, model counting, Erdős-Pósa Theorem, monadic second-order logic, cycle cut-sets, parameterized complexity.

Base Class	Description
HORN	Horn formulas, i.e., CNF formulas where each clause contains at most one positive literal.
RHORN	Renamable Horn formulas, i.e., CNF formulas that can be made Horn by flipping literals.
2-CNF	Krom formulas, i.e., CNF formulas where each clause contains at most two literals.
CLU	Cluster formulas, i.e., CNF formulas that are variable disjoint unions of hitting formulas. A formula is *hitting* if any two of its clauses clash in at least one variable.
UP	CNF formulas from which the empty formula or an empty clause can be derived by unit propagation.
FOREST	Acyclic formulas, i.e., CNF formulas whose incidence graphs are forests. The incidence graph is the undirected bipartite graph on clauses and variables where a variable is incident with all the clauses in which it occurs.

Table 1: Considered islands of tractability.

1   Introduction

Since the advent of computational complexity in the 1970s it quickly became apparent that a large number of important problems are intractable [20]. This predicament motivated significant efforts to identify tractable subproblems within intractable problems. For the propositional satisfiability problem (SAT), dozens of such "islands of tractability" have been identified [18]. Whereas it may seem unlikely that a real-world instance belongs to a known island of tractability, it may be "close" to one. In this paper we study the question of whether we can exploit the proximity of a SAT instance to the island of acyclic formulas algorithmically.

For SAT, the distance to an island of tractability (or *base class*) $\mathcal{C}$ is most naturally measured in terms of the number of variables that need to be instantiated to put the formula into $\mathcal{C}$. Williams *et al.* [41] introduced the term "*backdoor set*" for sets of such variables, and distinguished between weak and strong backdoor sets. A set B of variables is a *weak $\mathcal{C}$-backdoor set* of a CNF formula F if for at least one partial truth assignment $\tau : B \to \{0, 1\}$, the restriction $F[\tau]$ is satisfiable and belongs to the base class $\mathcal{C}$. ($F[\tau]$ is obtained from F by removing all clauses that contain a literal that is true under τ and by removing from the remaining clauses all literals that are false under τ.) The set B is a *strong $\mathcal{C}$-backdoor set* of F if for every partial truth assignment $\tau : B \to \{0, 1\}$ the restriction $F[\tau]$ belongs to $\mathcal{C}$. The base classes considered in the sequel are defined in Table 1.

1.1   Weak Backdoor Sets

If we are given a weak $\mathcal{C}$-backdoor set of F of size k, we know that F is satisfiable, and we can verify the satisfiability of F by checking whether at least one of the 2^k assignments to the backdoor variables leads to a formula that belongs to $\mathcal{C}$ and is satisfiable. If the base class allows to find an actual satisfying assignment in polynomial time, as is usually the case, we can find a satisfying assignment of F in $2^k n^{O(1)}$ time. Can we find such a backdoor set quickly if it exists? For all reasonable base classes $\mathcal{C}$ it is NP-hard to decide, given a CNF formula F and an integer k, whether F has a strong or weak $\mathcal{C}$-backdoor set of size at most k. On the other hand, the problem is clearly solvable in time $n^{k+O(1)}$. The question is whether we can get k out of the exponent, and find a backdoor set in time $f(k)n^{O(1)}$, i.e., is weak backdoor set detection *fixed-parameter tractable (FPT)* in k? Over the last couple of years, this question has been answered for various base classes $\mathcal{C}$; Table 2 gives an overview of some of the known results.

Base Class	Weak		Strong	
	CNF	r-CNF	CNF	r-CNF
HORN	W[2]-h [29]	FPT	FPT [29]	FPT [29]
2-CNF	W[2]-h [29]	FPT	FPT [29]	FPT [29]
UP	W[P]-c [39]	W[P]-c [39]	W[P]-c [39]	W[P]-c [39]
RHORN	W[2]-h [21]	W[2]-h [21]	W[2]-h [21]	open
CLU	W[2]-h [30]	FPT	W[2]-h [30]	FPT [30]

Table 2: The parameterized complexity of finding weak and strong backdoor sets of CNF formulas and r-CNF formulas, where $r \geq 3$ is a fixed integer. See [21] for a survey.

For general CNF, the detection of weak $\mathcal{C}$-backdoor sets is W[2]-hard for all reasonable base classes $\mathcal{C}$. For some base classes the problem becomes FPT if clause lengths are bounded. All fixed-parameter tractability results for weak backdoor set detection in Table 2 are due to the fact that for r-CNF formulas, where $r \geq 3$ is a fixed constant, membership in the considered base class can be characterized by certain obstructions of bounded size. Formally, say that a base class $\mathcal{C}$ has the *small obstruction property* if there is a family $\mathcal{F}$ of CNF formulas, each with a finite number of clauses, such that for any CNF formula F, $F \in \mathcal{C}$ iff F contains no subset of clauses isomorphic to a formula in $\mathcal{F}$. Hence, if a base class $\mathcal{C}$ has this property, fixed-parameter tractability for weak $\mathcal{C}$-backdoor set detection for r-CNF formulas can be established by a bounded search tree algorithm.

The base class FOREST is another class for which the detection of weak backdoor sets is W[2]-hard for general CNF formulas (Theorem 4). For r-CNF formulas the above argument does not apply because FOREST does not have the small obstruction property. Nevertheless, we can still show that the weak FOREST backdoor set detection problem is fixed-parameter tractable for r-CNF formulas, for any fixed $r \geq 3$ (Theorem 5). This is our first main result.

1.2 Strong Backdoor Sets

Given a strong $\mathcal{C}$-backdoor set of size k of a formula F, one can decide whether F is satisfiable by 2^k polynomial checks. In Table 2, HORN and 2-CNF are the only base classes for which strong backdoor set detection is FPT in general. A possible reason for the special status of these two classes is the fact that they have the *deletion property*: for $\mathcal{C} \in \{\text{HORN}, 2\text{-CNF}\}$ a set X of variables is a strong $\mathcal{C}$-backdoor set of a CNF formula F iff X is a *deletion $\mathcal{C}$-backdoor set* of F, i.e., the formula $F - X$, obtained from F by deleting all positive and negative occurrences of the variables in X, is in $\mathcal{C}$. The advantage of the deletion property is that it simplifies the search for a strong backdoor set. Its disadvantage is that the backdoor set cannot "repair" the given formula F differently for different truth assignments of the backdoor variables, and thus it does not use the full power of all the partial assignments. Indeed, for other base classes one can construct formulas with small strong backdoor sets whose smallest deletion backdoor sets are arbitrarily large. In view of these results, one wonders whether a small strong backdoor set can be found efficiently for a base class that does not have the deletion property. Our second main result provides a positive answer. Namely we exhibit an FPT algorithm, which, for a CNF formula F and a positive integer parameter k, either concludes that F has no strong FOREST-backdoor set of size at most k or concludes that F has a strong FOREST-backdoor set of size at most 2^k (Theorem 6).

This FPT-approximation result is interesting for several reasons. First, it implies that SAT and #SAT are FPT, parameterized by the size of a smallest strong FOREST-backdoor set. Sec-

ond, (unlike the size of a smallest deletion FOREST-backdoor set) the size of a smallest strong FOREST-backdoor set is incomparable to the treewidth of the incidence graph. Hence the result applies to formulas that cannot be solved efficiently by other known methods. Finally, it exemplifies a base class that does not satisfy the deletion property, for which strong backdoor sets are FPT-approximable.

1.3 #SAT and Implied Cycle Cutsets

Our second main result, Theorem 6, has applications to the model counting problem #SAT, a problem that occurs, for instance, in the context of Bayesian Reasoning [2, 35]. #SAT is #P-complete [40] and remains #P-hard even for monotone 2-CNF formulas and Horn 2-CNF formulas, and it is NP-hard to approximate the number of models of a formula with n variables within $2^{n^{1-\epsilon}}$ for $\epsilon > 0$, even for monotone 2-CNF formulas and Horn 2-CNF formulas [35]. A common approach to solve #SAT is to find a small *cycle cutset* (or feedback vertex set) of variables of the given CNF formula, and by summing up the number of satisfying assignments of all the acyclic instances one gets by setting the cutset variables in all possible ways [11]. Such a cycle cutset is nothing but a deletion FOREST-backdoor set. By considering strong FOREST-backdoor sets instead, one can get super-exponentially smaller sets of variables, and hence a more powerful method. A strong FOREST-backdoor set can be considered as a an *implied cycle cutset* as it can cut cycles by removing clauses that are satisfied by certain truth assignments to the backdoor variables. Theorem 6 states that we can find a small implied cycle cutset efficiently if one exists.

2 Preliminaries

Parameterized Complexity Parameterized Complexity [12, 16, 28] is a two-dimensional framework to classify the complexity of problems based on their input size n and some additional parameter k. It distinguishes between running times of the form $f(k)n^{g(k)}$ where the degree of the polynomial depends on k and running times of the form $f(k)n^{O(1)}$ where the exponential part of the running time is independent of n. The fundamental hierarchy of parameterized complexity classes is

$$\text{FPT} \subseteq \text{W}[1] \subseteq \text{W}[2] \cdots \subseteq \text{XP}.$$

An algorithm for a parameterized problem is an FPT algorithm if there is a function f such that the running time of the algorithm is upper bounded by $f(k)n^{O(1)}$. A parameterized problem is in FPT (fixed-parameter tractable) if it has an FPT algorithm, a problem is in XP if there are functions f, g such that the problem can be solved in time $f(k)n^{g(k)}$, and W[t], $t \geq 1$, are parameterized intractability classes giving strong evidence that a parameterized problem that is hard for any of these classes is not in FPT. These classes are closed under parameterized reductions, which are $f(k)n^{O(1)}$ time reductions where the target parameter is upper bounded by a function of the source parameter. All classes in this hierarchy are believed to be distinct. If FPT = W[1], then the Exponential Time Hypothesis [25] fails [7].

Backdoors A *literal* is a propositional variable x or its negation $\neg x$. A *clause* is a disjunction of literals that does not contain a complementary pair x and $\neg x$. A *propositional formula* in *conjunctive normal form* (CNF formula) is a conjunction of clauses. An r-CNF formula is a CNF formula where each clause contains at most r literals.

For a clause c, we write $\mathsf{lit}(c)$ and $\mathsf{var}(c)$ for the sets of literals and variables occurring in c, respectively. For a CNF formula F we write $\mathsf{cla}(F)$ for its set of clauses, $\mathsf{lit}(F) = \bigcup_{c \in \mathsf{cla}(F)} \mathsf{lit}(c)$ for its set of literals, and $\mathsf{var}(F) = \bigcup_{c \in \mathsf{cla}(F)} \mathsf{var}(c)$ for its set of variables.

Let F be a CNF formula and $X \subseteq \mathsf{var}(F)$. We denote by 2^X the set of all mappings $\tau : X \to \{0, 1\}$, the *truth assignments* on X. A truth assignment on X can be extended to the literals over X by setting $\tau(\neg x) = 1 - \tau(x)$ for all $x \in X$. Given a truth assignment $\tau \in 2^X$ we define $F[\tau]$ to be the formula obtained from F by removing all clauses c such that τ sets a literal of c to 1, and removing the literals set to 0 from all remaining clauses. A CNF formula F is *satisfiable* if there is some $\tau \in 2^{\mathsf{var}(F)}$ with $F[\tau] = \emptyset$. SAT is the NP-complete problem of deciding whether a given CNF formula is satisfiable [9, 27]. #SAT is the #P-complete problem of determining the number of distinct $\tau \in 2^{\mathsf{var}(F)}$ with $F[\tau] = \emptyset$ [40].

Backdoor Sets (BDSs) are defined with respect to a fixed class $\mathcal{C}$ of CNF formulas, the *base class*. From a base class we require the following properties:

1. $\mathcal{C}$ can be recognized in polynomial time,

2. the satisfiability of formulas in $\mathcal{C}$ can be decided in polynomial time, and

3. $\mathcal{C}$ is closed under isomorphisms (i.e., if two formulas differ only in the names of their variables, then either both or none belong to $\mathcal{C}$).

A polynomial time algorithm that determines the satisfiability of any CNF formula from $\mathcal{C}$ is called a sub-solver [23, 41].

Let B be a set of propositional variables and F be a CNF formula. B is a *strong $\mathcal{C}$-BDS* of F if $F[\tau] \in \mathcal{C}$ for each $\tau \in 2^B$. B is a *weak $\mathcal{C}$-BDS* of F if there is an assignment $\tau \in 2^B$ such that $F[\tau]$ is satisfiable and $F[\tau] \in \mathcal{C}$. B is a *deletion $\mathcal{C}$-BDS* of F if $F - B \in \mathcal{C}$, where $F - B = \{C \setminus \{x, \neg x : x \in B\} : C \in F\}$.

The challenging problem is to find a strong, weak, or deletion $\mathcal{C}$-BDS of size at most k if it exists. This leads to the following backdoor detection problems for any base class $\mathcal{C}$.

STRONG $\mathcal{C}$-BDS DETECTION

Input:　　　A CNF formula F and an integer $k \geq 0$.
Parameter:　The integer k.
Question:　Does F have a strong $\mathcal{C}$-backdoor set of size at most k?

The problems WEAK $\mathcal{C}$-BDS DETECTION and DELETION $\mathcal{C}$-BDS DETECTION are defined similarly.

Graphs　Let $G = (V, E)$ be a simple, finite graph. Let $S \subseteq V$ be a subset of its vertices and $v \in V$ be a vertex. We denote by $G - S$ the graph obtained from G by removing all vertices in S and all edges incident to a vertex in S. We denote by $G[S]$ the graph $G - (V \setminus S)$. The (open) neighborhood of v is $N(v) = \{u \in V : uv \in E\}$, the (open) neighborhood of S is $N(S) = \bigcup_{u \in S} N(u) \setminus S$, and their closed neighborhoods are $N[v] = N(v) \cup \{v\}$ and $N[S] = N(S) \cup S$, respectively. The set S is a *feedback vertex set* if $G - S$ is acyclic, and S is an *independent set* if $G[S]$ has no edge.

A *tree decomposition* of G is a pair $(\{X_i : i \in I\}, T)$ where $X_i \subseteq V$, $i \in I$, and T is a tree with elements of I as nodes such that:

1. $\bigcup_{i \in I} X_i = V$;

2. $\forall uv \in E$, $\exists i \in I$ such that $\{u, v\} \subseteq X_i$;

3. $\forall i, j, k \in I$, if j is on the path from i to k in T then $X_i \cap X_k \subseteq X_j$.

The *width* of a tree decomposition is $\max_{i \in I} |X_i| - 1$. The *treewidth* [33] of G is the minimum width taken over all tree decompositions of G and it is denoted by $\mathsf{tw}(G)$.

Acyclic Formulas The *incidence graph* of a CNF formula F is the bipartite graph $\mathsf{inc}(F) = (V, E)$ with $V = \mathsf{var}(F) \cup \mathsf{cla}(F)$ and for a variable $x \in \mathsf{var}(F)$ and a clause $c \in \mathsf{cla}(F)$ we have $xc \in E$ if $x \in \mathsf{var}(c)$. The edges of G may be annotated by a function $\mathsf{sign} : E \to \{+, -\}$. The *sign* of an edge xc is

$$\mathsf{sign}(xc) = \begin{cases} + & \text{if } x \in \mathsf{lit}(c), \text{ and} \\ - & \text{if } \neg x \in \mathsf{lit}(c) \ . \end{cases}$$

A *cycle* in F is a cycle in $\mathsf{inc}(F)$. The formula F is *acyclic* if $\mathsf{inc}(F)$ is acyclic [14]. We denote by FOREST the set of all acyclic CNF formulas.

The satisfiability of formulas from FOREST can be decided in polynomial time, and even the number of satisfying assignments of formulas from FOREST can be determined in polynomial time [15, 37].

The *strong clause-literal graph* of F is the graph $\mathsf{slit}(F) = (V, E)$ with $V = \mathsf{lit}(F) \cup \mathsf{cla}(F)$. There is an edge $uc \in E$, with $u \in \mathsf{lit}(F)$ and $c \in \mathsf{cla}(F)$ if $u \in \mathsf{lit}(c)$ and there is an edge $uv \in E$, with $u, v \in \mathsf{lit}(F)$ if $u = \neg v$ or $\neg u = v$. The following lemma clarifies the relation of the strong clause-literal graph with FOREST-BDSs.

Lemma 1. *Let F be a CNF formula, τ be an assignment to $B \subseteq \mathsf{var}(F)$. The formula $F[\tau]$ is acyclic iff $\mathsf{slit}(F) - N[\mathsf{true}(\tau)]$ is acyclic.*

Proof. There is a one-to-one correspondence between cycles in $\mathsf{inc}(F)$ and cycles in $\mathsf{slit}(F)$. Indeed, a cycle C in $\mathsf{inc}(F)$ can be obtained from a cycle C' in $\mathsf{slit}(F)$ by replacing each literal by its variable, and removing a variable x if it is preceded by x, and vice-versa. The correspondence is one-to-one as no clause contains complementary literals. Let C be a cycle in $\mathsf{inc}(F)$ which correponds to the cycle C' in $\mathsf{slit}(F)$. We have that C is not a cycle in $\mathsf{inc}(F[\tau])$ if there is a variable $x \in C \cap B$ or there is a clause $c \in C$ and a variable $x \in B$ such that $\tau(x) \in \mathsf{lit}(c)$. In the first case, C' is not a cycle in $\mathsf{slit}(F) - N[\mathsf{true}(\tau)]$ as $\{x, \neg x\} \subseteq N[\mathsf{true}(\tau)]$. In the second case, C' is not a cycle in $\mathsf{slit}(F) - N[\mathsf{true}(\tau)]$ as $c \in N[\mathsf{true}(\tau)]$. The reverse direction follows similarly. $\square$

It follows that there is a bijection between assignments τ such that $F[\tau]$ is acyclic and independent sets $Y \subseteq \mathsf{lit}(F)$ in $\mathsf{slit}(F)$ such that $\mathsf{slit}(F) - N[Y]$ is acyclic.

3 Background and Methods

The simplest type of FOREST-BDSs are deletion FOREST-BDSs. In the incidence graph, they correspond to feedback vertex sets that are subsets of $\mathsf{var}(F)$. Therefore, algorithms solving slight generalizations of FEEDBACK VERTEX SET can be used to solve the DELETION FOREST-BDS DETECTION problem. By results from [6] and [17], DELETION FOREST-BDS DETECTION is FPT and can be solved in time $5^k \cdot \|F\|^{O(1)}$ and in time $1.7548^n \cdot \|F\|^{O(1)}$, where n is the number of variables of F and $\|F\| = \sum_{c \in \mathsf{cla}(F)} |\mathsf{lit}(c)|$ denotes the formula length.

Any deletion FOREST-BDS B of a CNF formula F is also a strong FOREST-BDS of F and if F is satisfiable, then B is also a weak FOREST-BDS. In recent years SAT has been studied with respect to several width parameters of its primal, dual, and incidence graph [1, 15, 19, 31, 37, 38]. Several such parameters are more general than the size of a smallest deletion FOREST-BDS, such as the treewidth or the cliquewidth of the incidence graph. Parameterized by the treewidth of the incidence graph SAT is fixed-parameter tractable [15, 37], but the parameterization by cliquewidth is W[1]-hard, even when an optimal cliquewidth expression is provided [31]. Parameterized by the cliquewidth of the *directed* incidence graph (the orientation of an edge indicates whether the variable occurs positively or negatively), SAT becomes fixed-parameter tractable [15, 19]. It is not

known whether the problem of computing an optimal cliquewidth expression of a directed graph is FPT parameterized by the cliquewidth, but it has an FPT approximation algorithm [26], which is sufficient to state that SAT is FPT parameterized by the cliquewidth of the directed incidence graph.

The size of a smallest weak and strong FOREST-BDS is incomparable to treewidth and cliquewidth. On one hand, one can construct formulas with arbitrary large FOREST-BDSs by taking the disjoint union of formulas with bounded width. On the other hand, consider an $r \times r$ grid of variables and subdivide each edge by a clause. Now, add a variable x that is contained positively in all clauses subdividing horizontal edges and negatively in all other clauses. The set $\{x\}$ is a weak and strong FOREST-BDS of this formula, but the treewidth and cliquewidth of the formula depend on r. Therefore, weak and strong FOREST-BDSs have the potential of augmenting the tractable fragments of SAT formulas.

It would be tempting to use Chen *et al.*'s FPT algorithm for DIRECTED FEEDBACK VERTEX SET [8] for the detection of deletion BDSs. The corresponding base class would contain all CNF formulas with acyclic directed incidence graphs. Unfortunately this class is not suited as a base class since it contains formulas where each clause contains either only positive literals or only negative literals, and SAT is well known to be NP-hard for such formulas [20].

In the remainder of this section we outline our algorithms. To find a weak or strong FOREST-BDS, consider the incidence graph $G = \mathsf{inc}(F)$ of the input formula F. By Robertson and Seymour's Grid Minor Theorem [34] there is a function $f : \mathbb{N} \to \mathbb{N}$ such that for every integer r, either $\mathsf{tw}(G) \leq f(r)$ or G has an $r \times r$ grid minor. Choosing r to be a function of the parameter k, it suffices to solve the problems for incidence graphs whose treewidth is upper bounded by a function of k, and for incidence graphs that contain an $r \times r$ grid minor, where r is lower bounded by a function of k. The former case can be solved by invoking Courcelle's theorem [10] as the FOREST-BDS DETECTION problems can be defined in Monadic Second Order Logic. In the latter case we make use of the fact that G contains many vertex-disjoint cycles and we consider several cases how these cycles might disappear from $\mathsf{inc}(F)$ by assigning values to variables.

In order to obtain slightly better bounds, instead of relying on the Grid Minor Theorem we use the Erdős-Pósa Theorem [13] and an algorithmization by Bodlaender [4] to distinguish between the cases where G has small treewidth (in fact, a small feedback vertex set) or many vertex-disjoint cycles.

Theorem 1 ([13]). *Let $k \geq 0$ be an integer. There exists a function $f(k) = O(k \log k)$ such that every graph either contains k vertex-disjoint cycles or has a feedback vertex set of size $f(k)$.*

Theorem 2 ([4]). *Let $k \geq 2$ be an integer. There exists an $O(n)$ time algorithm, taking as input a graph G on n vertices, that either finds k vertex-disjoint cycles in G or finds a feedback vertex set of G of size at most $12k^2 - 27k + 15$.*

We will use Theorem 2 to distinguish between the case where G has a feedback vertex set of size $\mathsf{fvs}(k)$ and the case where G has $\mathsf{cycles}(k)$ vertex-disjoint cycles, for some function $\mathsf{cycles} : \mathbb{N} \to \mathbb{N}$, where $\mathsf{fvs}(k) = 12(\mathsf{cycles}(k))^2 - 27\mathsf{cycles}(k) + 15$.

Suppose G has a feedback vertex set W of size $\mathsf{fvs}(k)$. By adding W to every bag of an optimal tree decomposition of $G - W$, we obtain a tree decomposition of G of width at most $\mathsf{fvs}(k) + 1$. We use Courcelle's theorem [10], stating that every problem that can be defined in Monadic Second Order Logic (MSO) can be solved in linear time on structures of bounded treewidth. We use the notation of [16].

7

```
tw-MSO
  Input:       A relational structure $\mathcal{A}$ and an MSO-sentence $\varphi$.
  Parameter:   $\mathsf{tw}(A) + |\varphi|$.
  Question:    Decide whether $\mathcal{A} \models \varphi$.
```

Theorem 3 ([10]). *tw-MSO is fixed-parameter tractable.*

In Lemmas 2 and 5 we will define the WEAK and STRONG FOREST-BDS DETECTION problems as MSO-sentences, and Theorem 3 can then be used to solve the problems when a feedback vertex set of size $\mathsf{fvs}(k)$ is part of the input.

Our main arguments come into play when Bodlaender's algorithm returns a set $\mathcal{C}$ of $\mathsf{cycles}(k)$ vertex-disjoint cycles of G. The algorithms will then compute a set $S^* \subseteq \mathsf{var}(F)$ whose size is upper bounded by a function of k such that every weak/strong FOREST-BDS of size at most k contains a variable from S^*. A standard branching argument will then be used to recurse. In the case of WEAK FOREST-BDS DETECTION, F has a weak FOREST-BDS of size at most k iff there is a variable $x \in S^*$, such that $F[x = 0]$ or $F[x = 1]$ has a weak FOREST-BDS of size at most $k - 1$. In the case of STRONG FOREST-BDS DETECTION, F has no strong FOREST-BDS of size at most k if for every variable $x \in S^*$, $F[x = 0]$ or $F[x = 1]$ has no strong FOREST-BDS of size at most $k - 1$, and if $F[x = 0]$ and $F[x = 1]$ have strong FOREST-BDSs B and B' of size at most $2^{k-1} - 1$, then $B \cup B' \cup \{x\}$ is a strong FOREST-BDS of F of size at most $2^k - 1$, leading to a factor $2^k/k$ approximation.

In order to compute the set S^*, the algorithms consider how the cycles in $\mathcal{C}$ can interact with a BDS. Let x be a variable and C a cycle in G. In the case of weak FOREST-BDSs, we say that x *kills*[1] C if either $\mathsf{inc}(F[x = 1])$ or $\mathsf{inc}(F[x = 0])$ does not contain C. In the case of strong FOREST-BDSs, we say that x *kills* C if neither $\mathsf{inc}(F[x = 1])$ nor $\mathsf{inc}(F[x = 0])$ contain C. We say that x kills C *internally* if $x \in C$, and that x kills C *externally* if x kills C but does not kill it internally. In any FOREST-BDS of size at most k, at most k cycles from $\mathcal{C}$ can be killed internally, since all cycles from $\mathcal{C}$ are vertex-disjoint. The algorithms go over all possible choices of selecting k cycles from $\mathcal{C}$ that may be killed internally. All other cycles $\mathcal{C}'$ need to be killed externally. The algorithms now aim at computing a set S such that any weak/strong FOREST-BDS of size at most k which is a subset of $\mathsf{var}(F) \setminus \bigcup_{C \in \mathcal{C}'} \mathsf{var}(C)$ contains a variable from S. Computing the set S is the most challenging part of this work. In the algorithm for weak FOREST-BDSs there is an intricate interplay between several cases, making use of bounded clause lengths. In the algorithm for strong FOREST-BDSs a further argument is needed to obtain a more structured interaction between the considered cycles and their external killers.

4   Weak Forest-BDSs

By a parameterized reduction from HITTING SET, WEAK FOREST-BDS DETECTION is easily shown to be W[2]-hard.

Theorem 4. WEAK FOREST-BDS DETECTION *is* W[2]-*hard.*

Proof. We give a parameterized reduction from the W[2]-complete HITTING SET (HS) problem [12]. HS has as input a collection $\mathcal{S} = \{S_1, \dots, S_m\}$ of subsets S_i of a universe U and an integer parameter k. The question is whether there is a set $Y \subseteq U$ of size k such that every set from $\mathcal{S}$ contains an element from Y. In this case, Y is a *hitting set* of $\mathcal{S}$.

---

[1]We apologize for the violent language.

Create an instance F for WEAK FOREST-BDS DETECTION with variables $U \cup \{z_i, z_i' : S_i \in \mathcal{S}\}$ and for each $S_i \in \mathcal{S}$, add the clauses $c_i = \{z_i, z_i'\}$ and $c_i' = S_i \cup \{\neg z_i, \neg z_i'\}$. We claim that $\mathcal{S}$ has a hitting set of size k iff F has a weak FOREST-BDS of size k. Let Y be a hitting set of $\mathcal{S}$ of size k. Consider the formula $F' = F[\{y = 1 : y \in Y\}]$. F' contains no clause c_i', for any $1 \leq i \leq m$, as Y is a hitting set of $\mathcal{S}$. Thus, F' contains only clauses c_i, which are all variable-disjoint. Therefore, F' is acyclic and satisfiable. It follows that Y is a weak FOREST-BDS for F. On the other hand, suppose τ is an assignment to k variables such that $F[\tau]$ is acyclic and satisfiable. Obtain τ' from τ by replacing each assignment to z_i or z_i' by an assignment setting a literal from S_i to 1. $F[\tau']$ is also acyclic because any cycle passing through z_i, z_i', or c_i, also passes through c_i', and c_i' is removed from $F[\tau']$. Let $Y = (\tau')^{-1}(1)$. Then each clause c_i' contains a variable from Y, otherwise the cycle $(c_i', z_i', c_i, z_i, c_i')$ remains. Thus, Y is a hitting set of $\mathcal{S}$ of size at most k. □

In the remainder of this section, we consider the WEAK FOREST-BDS DETECTION problem for r-CNF formulas, for any fixed integer $r \geq 3$. Let F be an r-CNF formula, and consider its incidence graph $G = (V, E) = \mathsf{inc}(F)$. We use Theorem 2 to distinguish between the case where G has many vertex-disjoint cycles and the case where G has a small feedback vertex set. If G has a small feedback vertex set, the problem is expressed in MSO and solved by Courcelle's theorem.

Lemma 2. *Given a feedback vertex set of $\mathsf{inc}(F)$ of size $\mathsf{fvs}(k)$, WEAK FOREST-BDS DETECTION is fixed-parameter tractable.*

Proof. For any formula F, we define a relational structure A_F. The vocabulary of A_F is $\{\mathrm{LIT}, \mathrm{CLA}\}$, with $\mathrm{LIT} = \mathsf{lit}(F)$ and $\mathrm{CLA} = \mathsf{cla}(F)$. There is a unary relation $\mathrm{VAR} = \mathsf{var}(F)$, and symmetric binary relations $\mathrm{NEG} = \{x\neg x : x \in \mathsf{var}(F)\}$ and $\mathrm{EDGE} = \mathrm{NEG} \cup \{xc : x \in \mathrm{LIT}, c \in \mathrm{CLA}, x \in \mathsf{lit}(c)\}$.

Let S be a feedback vertex set of $\mathsf{inc}(F)$ of size at most $\mathsf{fvs}(k)$. A tree decomposition for the graph $\mathsf{inc}(F)$ can be obtained by starting from a trivial tree decomposition of width 1 for $\mathsf{inc}(F) - S$ and adding S to every bag of this tree decomposition. A tree decomposition for A_F can then be obtained by replacing each vertex by both its literals. This tree decomposition has width at most $2\mathsf{fvs}(k) + 3$.

To determine whether F has a weak FOREST-BDS of size k, we define an MSO-sentence $\varphi(Y)$, checking whether $F[\tau]$ is acyclic, where $\mathsf{true}(\tau) = Y$. Invoking Theorem 3 with the sentence $\exists y_1 \ldots \exists y_k(\varphi(\{y_1, \ldots, y_k\}))$ will then enable us to find a weak FOREST-BDS of F of size k if one exists.

Note that A_F encodes the graph $H = \mathsf{slit}(F)$, and by Lemma 1 it suffices to find an independent set $Y \subseteq \mathrm{LIT}$ of size k such that $\mathsf{slit}(F) - N[Y]$ is acyclic.

We break up φ into several simpler sentences. The following sentence checks whether Y is an assignment.

$$\varphi_{\mathrm{ass}}(Y) = \forall y(Yy \rightarrow (\mathrm{LIT}y \wedge (\neg \exists z(Yz \wedge \mathrm{NEG}yz))))$$

To make sure that $H - N[Y]$ is acyclic, it is sufficient that every subgraph of H with minimum degree at least 2 has a vertex from Y in its closed neighborhood. The following sentence checks whether the set C induces a subgraph with minimum degree at least 2.

$$\varphi_{\mathrm{deg2}}(C) = \forall x(Cx \rightarrow \exists y_1 \exists y_2(Cy_1 \wedge Cy_2 \wedge y_1 \neq y_2 \wedge \mathrm{EDGE}xy_1 \wedge \mathrm{EDGE}xy_2))$$

The following sentence checks whether C has a vertex from $N[Y]$.

$$\varphi_{\mathrm{kills}}(Y, C) = \exists x \exists y(Cx \wedge Yy \wedge (x = y \vee \mathrm{EDGE}xy))$$

9

Our final MSO-sentence checks whether Y is an independent set of LIT such that $H - N[Y]$ is acyclic.

$$\varphi(Y) = \varphi_{\mathrm{ass}}(Y) \wedge \forall C(\varphi_{\mathrm{deg2}}(C) \to \varphi_{\mathrm{kills}}(Y, C))$$

This proves the lemma. □

Let $\mathcal{C} = \{C_1, \ldots, C_{\mathsf{cycles}(k)}\}$ denote vertex-disjoint cycles in G, with $\mathsf{cycles}(k) = 2k + 1$. We describe an algorithm that finds a set S^* of $O(r 4^k k^6)$ variables from $\mathsf{var}(F)$ such that any weak FOREST-BDS of F of size at most k contains a variable from S^*.

We will use several functions of k in our arguments. Let

$$
\begin{aligned}
\mathsf{ext\text{-}cycles}(k) &:= \mathsf{cycles}(k) - k, \\
\mathsf{multi}(k) &:= 4k, \\
\mathsf{supp}(k) &:= (r - 3) \cdot (k^3 + 9) + 4k^2 + k, \text{ and} \\
\mathsf{overlap}(k) &:= (r - 2) \cdot (k \cdot \mathsf{multi}(k))^2 + k \ .
\end{aligned}
$$

Let C be a cycle in G and $x \in \mathsf{var}(F)$. Recall that x *kills* C *internally* if $x \in C$. In this case, x is an *internal killer* for C. We say that x *kills* C *externally* if $x \notin C$ and there is a clause $u \in \mathsf{cla}(F) \cap C$ such that $xu \in E$. In this case, x is an *external killer* for C. We first dispense with cycles that are killed internally. Our algorithm goes through all $\binom{\mathsf{cycles}(k)}{k}$ ways to choose k cycles from $\mathcal{C}$ that may be killed internally. W.l.o.g., let $C_{\mathsf{ext\text{-}cycles}(k)+1}, \ldots, C_{\mathsf{cycles}(k)}$ denote the cycles that may be killed internally. All other cycles $\mathcal{C}' = \{C_1, \ldots, C_{\mathsf{ext\text{-}cycles}(k)}\}$ need to be killed externally. Let $\mathsf{var}'(F) = \mathsf{var}(F) \setminus \bigcup_{i=1}^{\mathsf{ext\text{-}cycles}(k)} \mathsf{var}(C_i)$ denote the variables that may be selected in a weak FOREST-BDS killing no cycle from $\mathcal{C}'$ internally. From now on, consider only external killers from $\mathsf{var}'(F)$. The algorithm will find a set S of $O(r k^6)$ variables such that S contains a variable from any weak FOREST-BDS $B \subseteq \mathsf{var}'(F)$ of F with $|B| \le k$. The algorithm first computes the set of external killers (from $\mathsf{var}'(F)$) for each of these cycles. Then the algorithm applies the first applicable from the following rules.

Rule 1 (No External Killer). *If there is a $C_i \in \mathcal{C}'$ that has no external killer, then set $S := \emptyset$.*

For each $i \in \{1, \ldots, \mathsf{ext\text{-}cycles}(k)\}$, let x_i be an external killer of C_i that has a maximum number of neighbors in C_i.

Rule 2 (Multi-Killer Unsupported). *If there is a $C_i \in \mathcal{C}'$ such that x_i has $\ell \ge \mathsf{multi}(k)$ neighbors in C_i and at most $\mathsf{supp}(k)$ external killers of C_i have at least $\ell/(2k)$ neighbors in C_i, then include all these external killers in S.*

Rule 3 (Multi-Killer Supported). *If there is a $C_i \in \mathcal{C}'$ such that x_i has $\ell \ge \mathsf{multi}(k)$ neighbors in C_i and more than $\mathsf{supp}(k)$ external killers of C_i have at least $\ell/(2k)$ neighbors in C_i, then set $S := \{x_i\}$.*

Rule 4 (Large Overlap). *If there are two cycles $C_i, C_j \in \mathcal{C}'$, with at least $\mathsf{overlap}(k)$ common external killers, then set $S := \emptyset$.*

Rule 5 (Small Overlap). *Include in S all vertices that are common external killers of at least two cycles from $\mathcal{C}'$.*

Lemma 3. *Rules 1–5 are sound.*

Proof. We prove the correctness of Rules 1–5 in the order of their appearance.
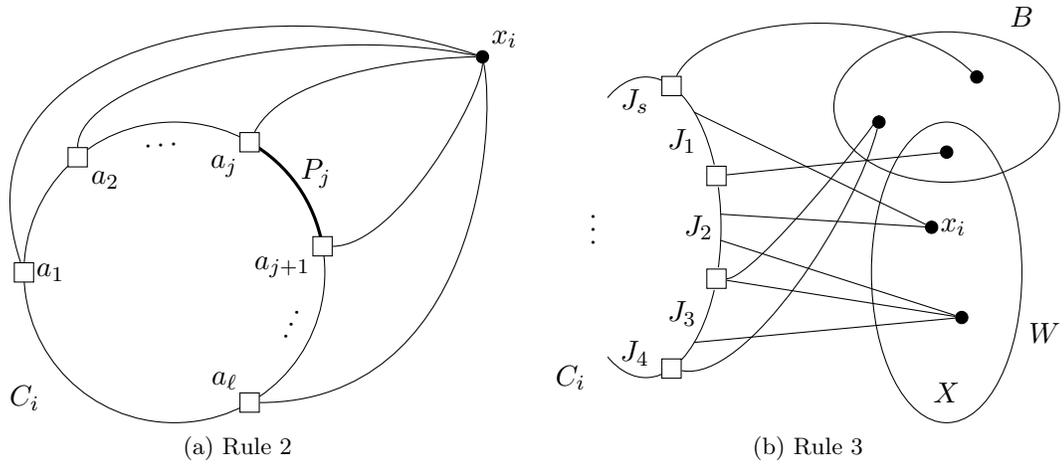
(a) Rule 2   (b) Rule 3

Figure 1: Helper figures for Rules 2 and 3. Clauses are represented by squares and variables by solid circles.

Rule 1 (No External Killer). *If there is a $C_i \in \mathcal{C}'$ that has no external killer, then set $S := \emptyset$.*

If C_i has no external killer (from $\mathsf{var}'(F)$), then F has no weak FOREST-BDS of size k which is a subset of $\mathsf{var}'(F)$.

Recall that for each $i \in \{1, \ldots, \mathsf{ext\text{-}cycles}(k)\}$, the variable x_i is an external killer of C_i that has a maximum number of neighbors in C_i.

Rule 2 (Multi-Killer Unsupported). *If there is a $C_i \in \mathcal{C}'$ such that x_i has $\ell \geq \mathsf{multi}(k)$ neighbors in C_i and at most $\mathsf{supp}(k)$ external killers of C_i have at least $\ell/(2k)$ neighbors in C_i, then include all these external killers in S.*

Consider a natural ordering $a_1, \ldots, a_\ell$ of the neighbors of x_i in C_i; i.e., $a_1, \ldots, a_\ell$ occur in this order on the cycle. See Figure 1a. For convenience, let $a_0 = a_\ell$ and $a_{\ell+1} = a_1$. Let P_j denote the set of vertices that are encountered when moving on the cycle from a_j to a_{j+1} without passing through a_{j-1}. The BDSs that do not contain x_i need to kill each of the cycles $P_j \cup \{x_i\}, 1 \leq j \leq \ell$, externally. All such BDSs of size at most k necessarily contain a vertex killing at least ℓ/k of these cycles, and such a vertex is an external killer of C_i with at least $\ell/(2k)$ neighbors in C_i.

Rule 3 (Multi-Killer Supported). *If there is a $C_i \in \mathcal{C}'$ such that x_i has $\ell \geq \mathsf{multi}(k)$ neighbors in C_i and more than $\mathsf{supp}(k)$ external killers of C_i have at least $\ell/(2k)$ neighbors in C_i, then set $S := \{x_i\}$.*

Let W denote the set of external killers of C_i with at least $\ell/(2k)$ neighbors in C_i. See Figure 1b. For the sake of contradiction, assume there exists a weak FOREST-BDS $B \subseteq \mathsf{var}'(F) \setminus \{x_i\}$ of F of size at most k. As x_i has a maximum number of neighbors in C_i, there are at most $k \cdot \ell$ edges connecting a vertex from B to a vertex from C_i. Consider the maximal segments $J_1, \ldots, J_s$ of C_i that do not contain a vertex adjacent to a vertex from B. By the previous observation, $s \leq k \cdot \ell$. Let H be an auxiliary bipartite graph with bipartition (J, X) of its vertex set, where $J = \{J_1, \ldots, J_s\}$ and $X = W \setminus B$, and an edge from $J_j \in J$ to $x \in X$ if x is adjacent to a vertex from J_j in G. The graph H is acyclic as any cycle in H could naturally be expanded into a cycle in G avoiding the neighborhood of all vertices in B by replacing vertices in J by paths in the corresponding segments of C_i. However, by counting the number of edges incident to X in H, which is the number of edges

11

from X to C_i minus the number of edges from X to a neighbor of B, we obtain that

$$
\begin{aligned}
|E(H)| &\geq \frac{\ell}{2k} \cdot |X| - (r-3) \cdot s && \text{(as any } u \in N(B) \cap C_i \text{ has at most } r-3 \text{ neighbors in } X) \\
&\geq |X| + \left(\frac{\ell}{2k} - 1\right) \cdot \left((r-3)(k^3+9) + 4k^2\right) - (r-3) \cdot s \\
&&& \text{(as } |X| \geq \mathsf{supp}(k) - k = (r-3)(k^3+9) + 4k^2) \\
&= |X| + 2k\ell - 4k^2 + (r-3)\left(\left(\frac{\ell}{2k} - 1\right)(k^3+9) - s\right) \\
&\geq |X| + k\ell + k \cdot (\ell - 4k) + (r-3)\left(\ell(k^2/2 - k) - k^3 + 9\right) \\
&&& \text{(as } \ell \geq \mathsf{multi}(k) = 4k \text{ and } s \leq k\ell) \\
&\geq |X| + k\ell + (r-3)\left(k^3 - 4k^2 + 9\right) && \text{(as } \ell \geq \mathsf{multi}(k) = 4k) \\
&\geq |X| + k\ell && \text{(as } r \geq 3 \text{ and } k^3 - 4k^2 + 9 \geq 0 \text{ for any integer } k \geq 1) \\
&\geq |X| + s && \text{(as } s \leq k \cdot \ell) \\
&= |V(H)| \ .
\end{aligned}
$$

Thus, H has a cycle, a contradiction.

Rule 4 (Large Overlap). *If there are two cycles $C_i, C_j \in \mathcal{C}'$, with at least $\mathsf{overlap}(k)$ common external killers, then set $S := \emptyset$.*

Consider any vertex subset $B \subseteq \mathsf{var}'(F)$ of size at most k. By the previous two rules, $|N[B] \cap C_i| \leq k \cdot (\mathsf{multi}(k) - 1)$ and $|N[B] \cap C_j| \leq k \cdot (\mathsf{multi}(k) - 1)$. We will show that there are two common external killers y_1 and y_2 of C_i and C_j such that $G[(\{y_1, y_2\} \cup C_i \cup C_j) \setminus N[B]]$ contains a cycle. Let us denote Y the set of common external killers of C_i and C_j. As there are at least $\mathsf{overlap}(k) - k$ edges between vertices from $Y \setminus B$ and vertices from C_i, the vertices from $Y \setminus B$ have at least $(\mathsf{overlap}(k) - k)/(r - 2)$ neighbors in C_i. The graph $G - N[B]$ contains at most $k \cdot (\mathsf{multi}(k) - 1)$ segments of the cycle C_i. There is at least one such segment with at least $\frac{\mathsf{overlap}(k) - k}{(r-2) \cdot k \cdot (\mathsf{multi}(k) - 1)} > k \cdot \mathsf{multi}(k)$ neighbors in $Y \setminus B$. At least two of these neighbors, y_1 and y_2, are adjacent to the same segment of $C_j \setminus N[B]$, creating a cycle in $G[(\{y_1, y_2\} \cup C_i \cup C_j) \setminus N[B]]$. As B was chosen arbitrarily, F has no weak FOREST-BDS that is a subset of $\mathsf{var}'(F)$.

Rule 5 (Small Overlap). *Include in S all vertices that are common external killers of at least two cycles from $\mathcal{C}'$.*

By the pigeonhole principle at least one variable of the BDS needs to kill at least two cycles from $\mathcal{C}' = \{C_1, \ldots, C_{k+1}\}$ externally. This vertex is among the common external killers of $C_1, \ldots, C_{k+1}$, whose number is upper bounded by $\frac{(k+1) \cdot k}{2} \cdot (\mathsf{overlap}(k) - 1)$ by the previous rule. $\qquad\square$

Lemma 4. *There is an FPT algorithm, which, given an r-CNF formula F, a positive integer parameter k, and $\mathsf{cycles}(k)$ vertex-disjoint cycles in $\mathsf{inc}(F)$, finds a set S^* of $O(r4^k k^6)$ variables in F such that every weak FOREST-BDS of F of size at most k contains a variable from S^*.*

Proof. The algorithm starts with $S^* = \emptyset$. For each choice $\mathfrak{C}$ among the $\binom{\mathsf{cycles}(k)}{k}$ cycles to be killed externally, the algorithm executes one of the described rules. It computes a set S such that every weak FOREST-BDS of F of size at most k respecting $\mathfrak{C}$ contains a variable from S. We set S^* to be the union of all S that are returned over all choices of cycles to be killed externally. As any weak FOREST-BDS respects at least one such choice, F has a weak FOREST-BDS of size at most k containing at least one variable from S^* if F has a weak FOREST-BDS of size at most k.

It remains to bound the size of S^*. The largest S are returned by Rule 5 and have size at most $O(rk^2 \cdot \mathsf{overlap}(k)) = O(rk^6)$. As $\binom{\mathsf{cycles}(k)}{k} \leq 2^{2k+1}$, the lemma follows. $\qquad\square$

Our FPT algorithm for WEAK FOREST-BDS DETECTION, restricted to r-CNF formulas, $r \geq 3$, is now easily obtained.

Theorem 5. *For any fixed $r \geq 3$, WEAK FOREST-BDS DETECTION is fixed-parameter tractable for r-CNF formulas.*

Proof. The final WEAK FOREST-BDS DETECTION algorithm for r-CNF formulas is recursive. Given an r-CNF formula F and an integer k, it computes the incidence graph $G = (V, E) = \mathsf{inc}(F)$. Then the algorithm from Theorem 2 is invoked with parameter $k' = \mathsf{cycles}(k)$. If that algorithm returns a feedback vertex set of size $O(k^2)$, we can conclude by Lemma 2. Otherwise, a set of $\mathsf{cycles}(k)$ vertex-disjoint cycles is returned. Then, Lemma 4 is used to compute a set S^* such that every weak FOREST-BDS of F of size at most k contains at least one variable from S^*. The algorithm recursively checks whether any formula $F[s = 0]$ or $F[s = 1]$, with $s \in S^*$, has a weak FOREST-BDS of size at most $k - 1$ and returns true is any such check was successful and false otherwise. □

5 Strong Forest-BDSs

In this section, we design an algorithm, which, for a CNF formula F and an integer k, either concludes that F has no strong FOREST-BDS of size at most k or concludes that F has a strong FOREST-BDS of size at most 2^k.

Let $G = (V, E) = \mathsf{inc}(F)$ denote the incidence graph of F. Again, we consider the cases where G has a small feedback vertex set or a large number of vertex-disjoint cycles separately. Let

$$\mathsf{cycles}(k) = k^2 2^{k-1} + k + 1,$$
$$\mathsf{ext\text{-}cycles}(k) = \mathsf{cycles}(k) - k, \text{ and}$$
$$\mathsf{fvs}(k) = 12(\mathsf{cycles}(k))^2 - 27\mathsf{cycles}(k) + 15 \ .$$

The case where G has a small feedback vertex set is again solved by formulating the problem in MSO and using Courcelle's theorem.

Lemma 5. *Given a feedback vertex set of $\mathsf{inc}(F)$ of size $\mathsf{fvs}(k)$, STRONG FOREST-BDS DETECTION is fixed-parameter tractable.*

Proof. We will use Theorem 3 and the relational structure A_F, defined in the proof of Lemma 2, to solve this problem. For a set $X = \{x_1, \ldots, x_k\}$ our MSO-sentence $\varphi(X)$ will decide whether X is a strong FOREST-BDS of F. It reuses several subformulas from the proof of Lemma 2 and checks, for each assignment to X, whether the resulting formula is acyclic. Invoking Theorem 3 with the sentence $\exists x_1 \ldots \exists x_k (\varphi(\{x_1, \ldots, x_k\}))$ will then enable us to find a strong FOREST-BDS of F of size k if one exists.

The following sentence checks whether X is a subset of variables.

$$\varphi_{\mathrm{var}}(X) = \forall x(Xx \to \mathrm{VAR}x)$$

An assignment of X is a subset of LIT containing no complementary literals such that every selected literal is a variable from X or its negation, and for every variable x from X, x or $\neg x$ is in Y. The following sentence checks whether Y is an assignment of X.

$$\varphi_{\mathrm{ass}}(X, Y) = \forall y(Yy \to ((Xy \vee (\exists z(Xz \wedge \mathrm{NEG}yz)))$$
$$\wedge (\forall z(Yz \to \neg\mathrm{NEG}yz))))$$
$$\wedge \forall x(Xx \to (Yx \vee \exists y(Yy \wedge \mathrm{NEG}xy)))$$

13

Our final sentence checks whether X is a set of variables such that each assignment to X kills all cycles in $\mathsf{inc}(F)$.

$$\phi(X) = \varphi_{\mathrm{var}}(X) \wedge \forall Y (\varphi_{\mathrm{ass}}(X, Y) \rightarrow (\forall C(\phi_{\mathrm{deg2}}(C) \rightarrow \phi_{\mathrm{kills}}(Y, C))))$$

As we can obtain a tree decomposition for A_F of width $2\mathsf{fvs}(k) + 3$ in polynomial time, and the length of ϕ is a function of k, the lemma follows by Theorem 3. $\qquad\square$

Let $\mathcal{C} = \left\{C_1, \ldots, C_{\mathsf{cycles}(k)}\right\}$ denote vertex-disjoint cycles in G. We refer to these cycles as $\mathcal{C}$-cycles. The aim is to compute a set $S^* \subseteq \mathsf{var}(F)$ of size $O(k^{2k} 2^{k^2 - k})$ such that every strong FOREST-BDS of F of size at most k contains a variable from S^*.

Let C be a cycle in G and $x \in \mathsf{var}(F)$. Recall that x *kills* C *internally* if $x \in C$. In this case, x is an *internal killer* for C. We say that x *kills* C *externally* if $x \notin C$ and there are two clauses $u, v \in \mathsf{cla}(F) \cap C$ such that $x \in \mathsf{lit}(u)$ and $\neg x \in \mathsf{lit}(v)$. In this case, x is an *external killer* for C and x kills C externally *in u and v*. As described earlier, our algorithm goes through all $\binom{\mathsf{cycles}(k)}{k}$ ways to choose k $\mathcal{C}$-cycles that may be killed internally. W.l.o.g., let $C_{\mathsf{ext\text{-}cycles}(k)+1}, \ldots, C_{\mathsf{cycles}}(k)$ denote the cycles that may be killed internally. All other cycles $\mathcal{C}' = \left\{C_1, \ldots, C_{\mathsf{ext\text{-}cycles}(k)}\right\}$ need to be killed externally. We refer to these cycles as $\mathcal{C}'$-cycles. Let $\mathsf{var}'(F) = \mathsf{var}(F) \setminus \bigcup_{i=1}^{\mathsf{ext\text{-}cycles}(k)} \mathsf{var}(C_i)$ denote the variables that may be selected in a strong FOREST-BDS killing no $\mathcal{C}'$-cycle internally. From now on, consider only external killers from $\mathsf{var}'(F)$. The algorithm will find a set S of at most 2 variables such that S contains a variable from any strong FOREST-BDS $B \subseteq \mathsf{var}'(F)$ of F with $|B| \leq k$. External killers and $\mathcal{C}'$-cycles might be adjacent in many different ways. The following procedure defines $\mathcal{C}x$-cycles that have a much more structured interaction with their external killers.

For each cycle $C_i \in \mathcal{C}'$ consider vertices x_i, u_i, v_i such that $x_i \in \mathsf{var}'(F)$ kills C_i externally in u_i and v_i and there is a path P_i from u_i to v_i along the cycle C_i such that if any variable from $\mathsf{var}'(F)$ kills C_i externally in two clauses u_i' and v_i' such that $u_i', v_i' \in P_i$, then $\{u_i, v_i\} = \{u_i', v_i'\}$. Let Cx_i denote the cycle $P_i \cup x_i$. We refer to the cycles in $\mathcal{C}x = \left\{Cx_1, \ldots, Cx_{\mathsf{ext\text{-}cycles}(k)}\right\}$ as $\mathcal{C}x$-cycles.

Observation 1. *Every external killer y of a $\mathcal{C}x$-cycle Cx_i is incident to u_i and v_i and $\mathsf{sign}(yu_i) \neq \mathsf{sign}(yv_i)$.*

Indeed, an external killer of C_i that is adjacent to two vertices from P_i with distinct signs is adjacent to u_i and v_i. Moreover, any external killer of Cx_i is a killer for C_i that is adjacent to two vertices from P_i with different signs. Thus, any external killer of Cx_i is adjacent to u_i and v_i.

We will be interested in external killers for $\mathcal{C}'$-cycles that also kill the corresponding $\mathcal{C}x$-cycles. That is, we are going to restrict our attention to vertices in $\mathsf{var}'(F)$ that kill Cx_i. An external killer of a $\mathcal{C}'$-cycle C_i is *interesting* if it is in $\mathsf{var}'(F)$ and it kills Cx_i. As each variable that kills a $\mathcal{C}x$-cycle Cx_i also kills C_i, and each $\mathcal{C}x$ cycle needs to be killed by a variable from any strong FOREST-BDS, we may indeed restrict our attention to interesting external killers of $\mathcal{C}'$-cycles.

We are now ready to formulate the rules to construct the set S containing at least one variable from any strong FOREST-BDS $B \subseteq \mathsf{var}'(F)$ of F of size at most k. These rules are applied in the order of their appearance, which means that a rule is only applicable if all previous rules are not.

Rule 6 (No External Killer). *If there is a $Cx_i \in \mathcal{C}x$ such that Cx_i has no external killer, then set $S := \{x_i\}$.*

Rule 7 (Killing Same Cycles). *If there are vertices y and z and at least $2^{k-1} + 1$ $\mathcal{C}'$-cycles such that both y and z are intersting external killers of each of these $\mathcal{C}'$-cycles, then set $S := \{y, z\}$.*

Rule 8 (Killing Many Cycles). *If there is a $y \in \mathsf{var}'(F)$ such that y is an interesting external killer of at least $k \cdot 2^{k-1} + 1$ $\mathcal{C}'$-cycles, then set $S := \{y\}$.*

Rule 9 (Too Many Cycles). *Set $S := \emptyset$.*

Lemma 6. *Rules 6–9 are sound.*

Proof. We prove the correctness of Rules 6–9 in the order of their appearance.

Rule 6 (No External Killer). *If there is a $Cx_i \in \mathcal{C}x$ such that Cx_i has no external killer, then set $S := \{x_i\}$.*

The correctness of Rule 6 follows since x_i is the only interesting external killer of C_i.

Rule 7 (Killing Same Cycles). *If there are vertices y and z and at least $2^{k-1} + 1$ $\mathcal{C}'$-cycles such that both y and z are interesting external killers of each of these $\mathcal{C}'$-cycles, then set $S := \{y, z\}$.*

We will show that at least one of y and z is in any strong FOREST-BDS $B \subseteq \mathsf{var}'(F)$ of F of size k. Suppose otherwise and consider a strong FOREST-BDS $B \subseteq \mathsf{var}'(F) \setminus \{y, z\}$ of F of size k. Consider the $\mathcal{C}'$-cycles for which y and z are interesting external killers and the set U of all variables u_i, v_i of each such $\mathcal{C}'$-cycle C_i as defined above. Note that $|U| \geq 2^k + 2$. We iteratively define a truth assignment τ to $B = \{b_1, \ldots, b_k\}$. Initially, all vertices in U are unmarked. At iteration i, let U_i and $\overline{U}_i$ denote the set of unmarked vertices from U that are incident with positive and negative edges to b_i, respectively. Set $\tau(b_i) = 1$ if $|\overline{U}_i| \geq |U_i|$, and set $\tau(b_i) = 0$ otherwise. If $\tau(b_i) = 1$, then mark all vertices in U_i, otherwise mark all vertices in $\overline{U}_i$. In the end, $F[\tau]$ contains at least $\lceil (2^k + 2)/2^k \rceil = 2$ variables from U, which form a cycle with y and z in $\mathsf{inc}(F[\tau])$. This cycle is a contradiction to B being a strong FOREST-BDS of F.

Rule 8 (Killing Many Cycles). *If there is a $y \in \mathsf{var}'(F)$ such that y is an interesting external killer of at least $k \cdot 2^{k-1} + 1$ $\mathcal{C}'$-cycles, then set $S := \{y\}$.*

As the previous rule is not applicable, every vertex $z \neq y$ is an interesting external killer for at most 2^{k-1} of these $\mathcal{C}'$-cycles. Thus, no set of interesting external killers of these $\mathcal{C}'$-cycles of size at most k excludes y. It follows that y is in any strong FOREST-BDS $B \subseteq \mathsf{var}'(F)$ of F of size k.

Rule 9 (Too Many Cycles). *Set $S := \emptyset$.*

If none of Rules 6–8 applies, then F has no strong FOREST-BDS $B \subseteq \mathsf{var}'(F)$ of F of size k. Indeed every vertex is an interesting external killer for at most $k \cdot 2^{k-1}$ $\mathcal{C}'$-cycles, but the number of $\mathcal{C}'$-cycles is $\mathsf{ext\text{-}cycles}(k) = \mathsf{cycles}(k) - k = k^2 \cdot 2^{k-1} + 1$. $\qquad\square$

The following lemma summarizes the construction of the set S^*.

Lemma 7. *There is an FPT algorithm that, given a CNF formula F, a positive integer parameter k, and $\mathsf{cycles}(k)$ vertex-disjoint cycles of $\mathsf{inc}(G)$, computes a set S^* of $O(k^{2k} 2^{k^2 - k})$ variables from $\mathsf{var}(F)$ such that every strong FOREST-BDS of F of size at most k includes a variable from S^*.*

Proof. The algorithm starts with $S^* = \emptyset$. For each choice $\mathfrak{C}$ among the $\binom{\mathsf{cycles}(k)}{k}$ cycles to be killed externally, the algorithm executes one of the described rules. It computes a set S such that every strong FOREST-BDS of F of size at most k respecting $\mathfrak{C}$ contains a variable from S. We set S^* to be the union of all S that are returned over all choices of cycles to be killed externally. As any strong FOREST-BDS respects at least one such choice, F has a strong FOREST-BDS of size at most k containing at least one variable from S^* if F has a strong FOREST-BDS of size at most k.

It remains to bound the size of S^*. The largest S are returned by Rule 8 and have size 2. As $\binom{\mathsf{cycles}(k)}{k} \leq (k^2 2^{k-1} + k + 1)^k$, the lemma follows. $\qquad\square$

This can now be used in an FPT-approximation algorithm for STRONG FOREST-BDS DETECTION. From this algorithm, it follows that SAT and #SAT, parameterized by the size of a smallest strong FOREST-BDS, are fixed-parameter tractable.

Theorem 6. *There is an* FPT *algorithm, which, for a CNF formula F and a positive integer parameter k, either concludes that F has no strong* FOREST-*BDS of size at most k or concludes that F has a strong* FOREST-*BDS of size at most 2^k.*

Proof. If $k \leq 1$, our algorithm solves the problem exactly in polynomial time. Otherwise, it invokes the algorithm from Theorem 2 to either find a set of at least $\mathsf{cycles}(k)$ vertex-disjoint cycles, or a feedback vertex set of G of size at most $\mathsf{fvs}(k)$.

In case it finds a feedback vertex set of G of size at most $\mathsf{fvs}(k)$, it uses Lemma 5 to compute a strong FOREST-BDS of F of size k if one exists, and it returns the answer.

In case it finds a set of at least $\mathsf{cycles}(k)$ vertex-disjoint cycles, it executes the procedure from Lemma 7 to find a set S^* of $O(k^{2k}2^{k^2-k})$ variables such that any strong FOREST-BDS of F of size at most k contains at least one variable from S^*. The algorithm considers all possibilities that the Backdoor set contains every $x \in S^*$; there are $O(k^{2k}2^{k^2-k})$ choices for x. For each such choice, recurse on $F[x = 1]$ and $F[x = 0]$ with parameter $k - 1$. If, for any $x \in S^*$, both recursive calls return strong FOREST-BDSs B and B', then return $B \cup B' \cup \{x\}$, otherwise, return NO. As $2^k - 1 = 2 \cdot (2^{k-1} - 1) + 1$, the solution size is upper bounded by $2^k - 1$. On the other hand, if at least one recursive call returns NO for every $x \in S^*$, then F has no strong FOREST-BDS of size at most k. $\square$

6 Conclusion

To identify large tractable subproblems the deletion of feedback vertex sets has been used in several other contexts, where instantiations of a smaller number of variables could already lead to acyclic subproblems. Examples include Nonmonotonic Reasoning [24], Bayesian inference [3, 32], and QBF satisfiability [5]. We believe that elements from our algorithms and proofs could be used in the design of parameterized, moderately exponential, and approximation algorithms for FOREST-BDS DETECTION problems and related problems such as finding a backdoor tree [36] of minimum height or with a minimum number of leaves, for SAT and problems in the above-mentioned contexts. Indeed, a similar approach has very recently been used to design an FPT-approximation algorithm for the detection of strong backdoor sets with respect to the base class of nested CNF formulas [22].

References

[1] Michael Alekhnovich and Alexander A. Razborov. Satisfiability, branch-width and Tseitin tautologies. In *Proc. of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2002)*, pages 593–603, 2002.

[2] Fahiem Bacchus, Shannon Dalmao, and Toniann Pitassi. Algorithms and complexity results for #SAT and Bayesian inference. In *Proc. of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2003)*, pages 340–351, 2003.

[3] Ann Becker, Reuven Bar-Yehuda, and Dan Geiger. Randomized algorithms for the loop cutset problem. *J. Artificial Intelligence Res.*, 12:219–234, 2000.

[4] Hans L. Bodlaender. On disjoint cycles. *International Journal of Foundations of Computer Science*, 5(1):59–68, 1994.

[5] Hubie Chen. Quantified constraint satisfaction and bounded treewidth. In *Proc. of the 16th Eureopean Conference on Artificial Intelligence (ECAI 2004)*, pages 161–165. IOS Press, 2004.

[6] Jianer Chen, Fedor V. Fomin, Yang Liu, Songjian Lu, and Yngve Villanger. Improved algorithms for feedback vertex set problems. *Journal of Computer and System Sciences*, 74(7):1188–1198, 2008.

[7] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences*, 72(8):1346–1367, 2006.

[8] Jianer Chen, Yang Liu, Songjian Lu, Barry O'Sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *Journal of the ACM*, 55(5):Art. 21, 19, 2008.

[9] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proc. of the 3rd Annual ACM Symposium on Theory of Computing (STOC 1971)*, pages 151–158, 1971.

[10] Bruno Courcelle. Graph rewriting: an algebraic and logic approach. In *Handbook of theoretical computer science, Vol. B*, pages 193–242. Elsevier, 1990.

[11] Rina Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[12] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, New York, 1999.

[13] Paul Erdős and Lajos Pósa. On independent circuits contained in a graph. *Canadian Journal of Mathematics*, 17:347–352, 1965.

[14] Ronald Fagin. Degrees of acyclicity for hypergraphs and relational database schemes. *Journal of the ACM*, 30(3):514–550, 1983.

[15] E. Fischer, J. A. Makowsky, and E. R. Ravve. Counting truth assignments of formulas of bounded tree-width or clique-width. *Discr. Appl. Math.*, 156(4):511–529, 2008.

[16] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series*. Springer Verlag, Berlin, 2006.

[17] Fedor V. Fomin, Serge Gaspers, Artem V. Pyatkin, and Igor Razgon. On the minimum feedback vertex set problem: exact and enumeration algorithms. *Algorithmica*, 52(2):293–307, 2008.

[18] John Franco and John Martin. A history of satisfiabilty. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, chapter 1, pages 3–97. IOS Press, 2009.

[19] Robert Ganian, Petr Hliněný, and Jan Obdržálek. Better algorithms for satisfiability problems for formulas of bounded rank-width. In Kamal Lodaya and Meena Mahajan, editors, *Proc. of the 30th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010)*, volume 8 of *LIPIcs*, pages 73–83. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.

[20] Michael R. Garey and David R. Johnson. *Computers and Intractability*. W. H. Freeman and Company, New York, San Francisco, 1979.

[21] Serge Gaspers and Stefan Szeider. Backdoors to satisfaction. arXiv CoRR 1110.6387, 2011.

[22] Serge Gaspers and Stefan Szeider. Strong backdoors to nested satisfiability. arXiv CoRR 1202.4331, 2012.

[23] Carla P. Gomes, Henry Kautz, Ashish Sabharwal, and Bart Selman. Satisfiability solvers. In *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, pages 89–134. Elsevier, 2008.

[24] Georg Gottlob, Francesco Scarcello, and Martha Sideri. Fixed-parameter complexity in AI and non-monotonic reasoning. *Artificial Intelligence*, 138(1-2):55–86, 2002.

[25] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

[26] Mamadou Moustapha Kanté and Michaël Rao. **F**-rank-width of (edge-colored) graphs. In *Proceedings of the 4th International Conference on Algebraic Informatics (CAI 2011)*, volume 6742 of *Lecture Notes in Computer Science*, pages 158–173. Springer, 2011.

[27] Leonid Levin. Universal sequential search problems. *Problems of Information Transmission*, 9(3):265–266, 1973.

[28] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and its Applications. Oxford University Press, Oxford, 2006.

[29] Naomi Nishimura, Prabhakar Ragde, and Stefan Szeider. Detecting backdoor sets with respect to Horn and binary clauses. In *Proc. of the 7th International Conference on Theory and Applications of Satisfiability Testing (SAT 2004)*, pages 96–103, 2004.

[30] Naomi Nishimura, Prabhakar Ragde, and Stefan Szeider. Solving #SAT using vertex covers. *Acta Informatica*, 44(7-8):509–523, 2007.

[31] Sebastian Ordyniak, Daniël Paulusma, and Stefan Szeider. Satisfiability of acyclic and almost acyclic CNF formulas. In Kamal Lodaya and Meena Mahajan, editors, *Proc. of the 30th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010)*, volume 8 of *LIPIcs*, pages 84–95. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.

[32] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. The Morgan Kaufmann Series in Representation and Reasoning. Morgan Kaufmann, San Mateo, CA, 1988.

[33] Neil Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986.

[34] Neil Robertson and P. D. Seymour. Graph minors. V. Excluding a planar graph. *J. Combin. Theory Ser. B*, 41(1):92–114, 1986.

[35] Dan Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1-2):273–302, 1996.

[36] Marko Samer and Stefan Szeider. Backdoor trees. In *Proceedings of the 23rd Conference on Artificial Intelligence (AAAI 2008)*, pages 363–368. AAAI Press, 2008.

[37] Marko Samer and Stefan Szeider. Algorithms for propositional model counting. *J. Discrete Algorithms*, 8(1):50–64, 2010.

[38] Stefan Szeider. On fixed-parameter tractable parameterizations of SAT. In Enrico Giunchiglia and Armando Tacchella, editors, *Proc. of the 6th International Conference on Theory and Applications of Satisfiability (SAT 2003)*, volume 2919 of *LNCS*, pages 188–202. Springer Verlag, 2004.

[39] Stefan Szeider. Backdoor sets for DLL subsolvers. *Journal of Automated Reasoning*, 35(1-3):73–88, 2005.

[40] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.

[41] Ryan Williams, Carla Gomes, and Bart Selman. Backdoors to typical case complexity. In Georg Gottlob and Toby Walsh, editors, *Proc. of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 1173–1178. Morgan Kaufmann, 2003.