Solving d-SAT via Backdoors to Small Treewidth

Fedor V. Fomin[1], Daniel Lokshtanov[1], Neeldhara Misra[2], M. S. Ramanujan[1], and Saket Saurabh[1,3]

[1]Department of Informatics, University of Bergen, Norway, fomin@ii.uib.no, daniello@uib.no
[2]Indian Institute of Science, Bangalore, neeldhara@csa.iisc.ernet.in
[3]Institute of Mathematical Sciences, India, saket@imsc.res.in

Abstract

A backdoor set of a CNF formula is a set of variables such that fixing the truth values of the variables from this set moves the formula into a polynomial-time decidable class. In this work we obtain several algorithmic results for solving d-SAT, by exploiting backdoors to d-CNF formulas whose incidence graphs have small treewidth.

For a CNF formula ϕ and integer t, a *strong backdoor set* to treewidth t is a set of variables such that *each* possible partial assignment τ to this set reduces ϕ to a formula whose incidence graph is of treewidth at most t. A *weak backdoor set* to treewidth t is a set of variables such that there is a partial assignment to this set that reduces ϕ to a *satisfiable* formula of treewidth at most t. Our main contribution is an algorithm that, given a d-CNF formula ϕ and an integer k, in time $2^{\mathcal{O}(k)}|\phi|$,

- either finds a satisfying assignment of ϕ, or
- reports correctly that ϕ is not satisfiable, or
- concludes correctly that ϕ has no weak or strong backdoor set to treewidth t of size at most k.

As a consequence of the above, we show that d-SAT parameterized by the size of a smallest weak/strong backdoor set to formulas of treewidth t, is fixed-parameter tractable. Prior to our work, such results were know only for the very special case of $t = 1$ (Gaspers and Szeider, ICALP 2012). Our result not only extends the previous work, it also improves the running time substantially. The running time of our algorithm is linear in the input size for every fixed k. Moreover, the exponential dependence on the parameter k is asymptotically optimal under Exponential Time Hypothesis (ETH).

One of our main technical contributions is a linear time "protrusion replacer" improving over a $\mathcal{O}(n \log^2 n)$-time procedure of Fomin et al. (FOCS 2012). The new deterministic linear time protrusion replacer has several applications in kernelization and parameterised algorithms.

1 Introduction

There is a mysterious disparity in the way the Boolean Satisfiability problem (also often referred to as Propositional Satisfiability and abbreviated as SAT) is perceived by different communities in Computer Science and Artificial Intelligence. From a theoretician's perspective, since SAT is NP-complete, the existence of an efficient algorithm for this problem is highly unexpected. Even worse, all currently known algorithms for SAT with n variables, in the worst case, do not perform significantly better than a trivial brute-force algorithm trying all possible 2^n assignments to the variables [5]. On the other hand, in practice, modern SAT solvers can solve instances with hundreds thousands of variables within seconds. According to Malik and Zhang [21], similar to mathematical programming tools or linear equation solvers, SAT solvers have matured to the point to be used in a wide range of application domains. Thus encoding a problem as an instance of SAT and then applying a SAT solver is a success story for many applications. Understanding the reasons for such a huge discrepancy between theory and practice is not only an intellectual challenge, it also can bring us closer to even faster SAT solvers.

The notion of backdoors to SAT was introduced by Williams et al. in [26] in an effort towards a rigorous understanding of the surprising performance of SAT solvers in practice. Roughly speaking, backdoors are small set of variables capturing the overall combinatorics of the SAT instance. The definition of backdoors is based on the notion of a polynomial time algorithm, a *sub-solver*, solving tractable instances of SAT. A sub-solver is an algorithm $\mathcal{A}$ which, given a formula ϕ, in polynomial time either rejects the input or correctly solves ϕ. Furthermore, given any partial assignment τ to the variables of ϕ, if $\mathcal{A}$ solves ϕ, then $\mathcal{A}$ also solves the reduced instance $\phi[\tau]$. For example, $\mathcal{A}$ can be an algorithm solving ϕ if it is an instance of 2-SAT and rejecting all other instances.

A nonempty subset B of the variables is a *weak backdoor* to ϕ for a sub-solver $\mathcal{A}$ if there exists an assignment τ to the variables in B such that $\mathcal{A}$ returns a satisfying assignment of the reduced instance $\phi[\tau]$. We say that B is a *strong backdoor* to ϕ for a sub-solver $\mathcal{A}$ if for each assignment τ to the variables in b, $\mathcal{A}$ solves $\phi[\tau]$, i.e. either returns a satisfying assignment or concludes unsatisfiability of $\phi[\tau]$. It appears that many instances in practice happen to have small "weak" or "strong" backdoors for different sub-solvers [20, 26]. There has been an extensive theoretical study of backdoors to different sub-solvers in the realm of parameterized complexity [23, 25, 16]. We refer to the surveys of Gaspers and Szeider [15] for more background.

One of the well-studied classes of SAT solvable in polynomial time is the class of "decomposable" formulas. In particular, the tree- (and its close relative branch-) width measures have been applied to satisfiability in the following way. If the treewidth of the incidence graph (the bipartite graph on the variables and clauses where a variable is adjacent to all the clauses containing it) does not exceed some constant, then SAT for such formulas can be decided in polynomial time [6, 11, 24].

Since the property of having an incidence graph with small treewidth makes SAT polynomial time solvable, it is very natural to ask about backdoors to a sub-solver on formulas of bounded treewidth. The study of such backdoors from the parameterized complexity perspective was initiated by Gaspers and Szeider in [14, 16]. In [14], Gaspers and Szeider study the problem of detecting a weak backdoor of size at most k to acyclic SAT, i.e, a weak backdoor to a sub-solver on formulas with incidence graphs of treewidth at most 1. They show that this problem is W[2]-hard in general but FPT on d-CNF formulas for fixed d, when parameterized by k. In [16] Gaspers and Szeider gave an FPT-approximation algorithm for strong backdoor set to treewidth at most t which either detects in time $f(k)n^3$, for some function f, a strong backdoor of size at most 2^k or reports that there is no strong backdoor of such size.

Let $\mathcal{W}_\eta$ be a class of formulas of treewidth at most η. Let us note that for a formula ϕ, the minimum sizes of weak and strong $\mathcal{W}_\eta$-backdoor sets can be very different. For a satisfiable formula the minimum size of a weak backdoor does not exceed the size of a strong one. However, this is not true for unsatisfiable formulas. For example, any unsatisfiable formula does not have a weak backdoor but it could have a small strong backdoor. In this work we give an FPT algorithm for d-SAT parameterized by the minimum of both sizes. Formally, our main result is the following.

Theorem 1. *There is an algorithm that takes as input a d-CNF formula ϕ and an integer k, runs in time $2^{\mathcal{O}(k)}|\phi|$ and*

- *either finds a satisfying assignment of ϕ, or*
- *reports correctly that ϕ is not satisfiable, or*
- *concludes correctly that ϕ has no weak or strong $\mathcal{W}_\eta$-backdoor set of size at most k.*

The main features of our result as well as the techniques are the following.

* It extends the tractability results for d-SAT in [14] to a significantly larger class of d-CNF formulas. Furthemore, although our algorithm for d-SAT does not rely on actually computing the entire backdoor sets, our methods show that a weak backdoor set to treewidth at most t can in fact be detected in FPT time.

* The running time of our algorithm is $2^{\mathcal{O}(k)}|\phi|$, that is, it has a single exponential dependence on the parameter and linear dependence on the input length $|\phi|$. It is also easy to show that unless the Exponential Time Hypothesis (ETH) fails, there is no $2^{o(k)}|\phi|^{\mathcal{O}(1)}$ solving d-SAT for every $d \geq 3$ [17]. Thus, our algorithm is asymptotically optimal.

* On the way to obtaining our algorithm we develop a new deterministic linear time protrusion replacer algorithm (we refer to Preliminaries for the definition of a protrusion replacer). Prior to our work the best deterministic protrusion replacer was of running time $\mathcal{O}(n \log^2 n)$ [7]. This improvement implies a speedup for many parameterized and kernelization algorithms based on protrusion replacements. In particular, due to this replacement, all kernelization algorithms obtained in [2, 8, 9, 12, 19] and parameterized algorithms from [7, 19] can be implemented to run in deterministic linear time.

At first glance, the problem of detecting a weak $\mathcal{W}_\eta$-backdoor set resembles the algorithmic graph problem of deleting at most k vertices such that the new graph is of treewidth at most t. However, as it was observed by Gaspers and Szeider in [14], already the problem of computing a weak backdoor set to acyclic d-SAT is very different from the seemingly related FEEDBACK VERTEX SET problem because while the size of the backdoor, k, can be very small, the treewidth of the incidence graph can be unbounded by any function of k. As a result, the techniques developed by a subset of the authors in [7] merely provide a starting point and need to be built upon in a problem specific way to detect backdoor sets. To further confirm this intuition, we show that under standard complexity theoretic assumptions, the problem of detecting a weak $\mathcal{W}_\eta$-backdoor does not admit a polynomial kernel. This separates the kernelization complexity of the two "related" problems because the vertex removal problem does in fact admit a polynomial kernel [7].

We briefly describe the ideas involved in our randomized algorithm, which are somewhat easier to explain. We say that a subset S of a graph G is a $\mathcal{W}_\eta$-modulator if the treewidth of $G - S$ is at most η. Our starting point is the observation that if X is a subset of variables that form a strong (or weak) $\mathcal{W}_\eta$-backdoor, then the set of their neighbours in the incidence graph, $N(X)$, is a $\mathcal{W}_\eta$-modulator. Note that $|N(X)|$ could be arbitrarily large compared to $|X|$. In particular, it is futile to attempt to look for a small $\mathcal{W}_\eta$-modulator among the clauses. We begin with a linear-time preprocessing procedure which ensures that for every $\mathcal{W}_\eta$-backdoor set X, the set $N(X)$ is incident with a large fraction of the edges in G. Therefore, if we pick an edge uniformly at random, the clause endpoint of the edge belongs to $N(X)$ with constant probability. Further, since the clauses are of constant size d, we have that a randomly chosen variable from the clause belongs to X with a constant probability $f(d, \eta)$.

When we are working with SAT using weak backdoors, the algorithm simply branches on the chosen variable x in the usual way — in one branch, we simplify the formula by setting x to 1, and in the other branch, we set x to 0. At this point, we recurse. However, when working with SAT using strong backdoors, it is not clear that this approach can be used as it is. Our algorithm solving SAT using weak backdoors uses the fact that a formula admits a weak backdoor if and only if it is satisfiable. On the other hand, in the case of strong backdoors, we are faced with *three* possible scenarios — that the formula does not admit a small strong backdoor, or that it does, and it is either satisfiable or not. Combining the varied recursive outputs appropriately is less obvious in this situation. The typical approach for solving SAT using strong backdoors involves first *finding* a strong backdoor using a search tree similar to the above. The set output by the recursive procedure is the union of all the recursively obtained solutions and thus its size can be proportional to the size of the search tree, often 2^k. Finally, SAT is solved in the standard way, which involves trying all possible truth assignments of the strong backdoor, and therefore, the overall expense incurred for solving SAT is $2^{2^k}|\phi|^{\mathcal{O}(1)}$. Under ETH, even if we are given a strong backdoor of size ℓ, we do not expect algorithms solving SAT in time $2^{o(\ell)}|\phi|^{\mathcal{O}(1)}$. Fortunately, it turns out that *detecting* backdoors is not a prerequisite to solving SAT. Indeed, in our algorithms, we sidestep the problem of detecting strong backdoors, and directly achieve a running time of $2^{\mathcal{O}(k)}|\phi|$, where k is the size of a smallest $\mathcal{W}_\eta$-strong backdoor to ϕ.

One of the main ingredients of our algorithm is the linear time preprocessing step which ensures that a large fraction of the edges are incident with the neighbors of every backdoor set. Towards this we give a new deterministic linear time "protrusion replacer" which has several applications in kernelization and parameterized algorithms. A protrusion is a subgraph that has constant treewidth and a constant-sized neighbourhood. Protrusions were employed in [2, 9] for obtaining meta-kernelization theorems for problems on sparse graphs like planar and H-minor-free graph. Our new protrusion replacer algorithm begins by enumerating all connected sets of size p with neighbourhood of size q. By a classical lemma of Bollobás [18], it can be shown that the number of such sets is at most $\binom{p+q}{p}$. However, for the purposes of developing the protrusion replacer, we use the enumeration algorithm proposed by Fomin and Villanger [10] in the context of designing exact algorithms for treewidth. Given these $n \cdot \binom{p+q}{p}$ sets, we carefully partition them into groups such that each of them form protrusions that are mutually internally disjoint (that is, while they may share their boundaries, their interiors do not overlap). We are also able to prove that these protrusions together account for a large fraction of the vertices appearing in any "collection of protrusions".

Having found these protrusions, we need an algorithm that can reduce protrusions, that is, remove these protrusions and replace them with smaller ones maintaining equivalence. We note that the known results about protrusion replacement cannot be used directly here. The existing machinery for replacing protrusions relies crucially on the notion of *finite integer index*. However, in our context, defining an appropriately equivalent notion applicable in the usual way seems rather difficult. Thus, we resort to the "finite state" style of making protrusion replacement. Also this is a more practical and arguably more direct line of attack. We consider a tree decomposition of the protrusion and analyze it to identify bags that are "equivalent", and then suggest a suitable reduction rule. The methods described here are similar in spirit to the ideas used in [13] for kernelization.

2 Preliminaries

2.1 CNF Formulas and Assignments

We consider propositional formulas in conjunctive normal form (CNF). We assume, without loss of generality, that the clauses do not contain a pair of complementary literals. For a CNF formula ϕ, we use $\mathsf{var}(\phi)$ and $\mathsf{cla}(\phi)$ to refer to the sets of variables and clauses in ϕ, respectively. We say that a variable x is positive (negative) in a clause C if $x \in C$ ($\overline{x} \in C$), and we write $\mathsf{var}(C)$ for the set of variables that are positive or negative in C, while we use $\mathsf{lit}(C)$ to denote the set of literals in C.

The *length* of a formula ϕ is given by $|\mathsf{var}(\phi)| + \sum_{C \in \mathsf{cla}(\phi)}(1 + |\mathsf{var}(C)|)$ and is denoted by $|\phi|$. A *truth assignment* τ is a mapping from a set of variables, denoted by $\mathsf{var}(\tau)$, to $\{0, 1\}$. A truth assignment τ *satisfies* a clause C if it sets at least one positive variable of C to 1 or at least one negative variable of C to 0. A truth assignment τ of $\mathsf{var}(\phi)$ satisfies the formula ϕ if it satisfies all clauses of ϕ. Given a CNF formula ϕ and a truth assignment τ, $\phi[\tau]$ denotes the *truth assignment reduct* of ϕ under τ, which is the CNF formula obtained from ϕ by first removing all clauses that are satisfied by τ and second removing from the remaining clauses all literals $x, \overline{x}$ with $x \in \mathsf{var}(\tau)$. For a formula ϕ and a subset of clauses $\mathcal{C} \subseteq \mathsf{cla}(\phi)$, we use $\phi \setminus \mathcal{C}$ to denote the formula ϕ with the clauses in $\mathcal{C}$ removed.

The *incidence graph* of a CNF formula ϕ, $\mathsf{inc}(\phi)$, is the bipartite graph whose vertices are the variables and clauses of ϕ, and where vertices corresponding to a variable x and a clause C are adjacent if and only if $x \in \mathsf{var}(C)$. Further, an edge between a vertex corresponding to $x \in \mathsf{var}(\phi)$ and $C \in \mathsf{cla}(\phi)$ has the label $+$ if $x \in \mathsf{lit}(C)$ and is labeled $-$ if $\overline{x} \in \mathsf{lit}(C)$.

We refer to the class of two-edge colored bipartite graphs as *SAT incidence graphs*, or incidence graphs for short. Typically, we use $(G = (X, C), E, \ell)$ to denote an incidence graph, where $\ell : E \to \{+, -\}$. The formula $\psi(G)$ is defined over the variable set $\{x_v \mid v \in X\}$, with a clause C_u for every $u \in C$. Further, the clause C_u consists of the literals:

$$\{x_v \mid (x_v, u) \in E \text{ and } \ell(x_v, u) = +\} \cup \{\overline{x_v} \mid (x_v, u) \in E \text{ and } \ell(x_v, u) = -\}.$$

For an incidence graph G, we abuse notation and use $\mathsf{var}(G)$ to refer to the vertices of G that correspond to variables in $\psi(G)$, and $\mathsf{cla}(G)$ to refer to the vertices of G that correspond to clauses in $\psi(G)$. Also, for a vertex subset $X \subseteq V(G)$, we continue to use the notations $\mathsf{var}(X)$ and $\mathsf{cla}(X)$ to refer to the sets $\mathsf{var}(G) \cap X$ and $\mathsf{cla}(G) \cap X$, respectively.

We say that G is an incidence graph of order d if G is an incidence graph where the maximum degree among the vertices in $\mathsf{cla}(G)$ is bounded by d. Note that these graphs correspond to d-CNF formulas (where a d-CNF formula involves clauses of length at most d).

Let $\mathcal{B}$ denote a fixed class of formulas under consideration. A *weak $\mathcal{B}$-backdoor set* of a CNF formula ϕ is a set B of variables such that there is a truth assignment τ of the variables in B such that the formula $\phi[\tau]$ is satisfiable and $\phi[\tau] \in \mathcal{B}$. Such an assignment is called a *witness assignment* for the weak backdoor. A *strong $\mathcal{B}$-backdoor set* of F is a set B of variables such that for each truth assignment τ of the variables in B, the formula $\phi[\tau]$ is in $\mathcal{B}$.

We let $\mathcal{K}_\eta$ denote the class of formulas ϕ for which $\mathsf{inc}(F)$ excludes the $(\eta \times \eta)$ grid as a minor (c.f. Subsection 2.5 for the relevant definitions), and let $\mathsf{wb}_g(\phi, \eta)$ (respectively, $\mathsf{sb}_g(\phi, \eta)$) denote the smallest possible size of a weak (respectively, strong) $\mathcal{K}_\eta$ backdoor. Also, we let $\mathcal{W}_\eta$ denote the class of formulas ϕ for which $\mathsf{inc}(F)$ has treewidth at most η (c.f. Subsection 2.3 for the relevant definition), and let $\mathsf{wb}_{tw}(\phi, \eta)$ (respectively, $\mathsf{sb}_{tw}(\phi, \eta)$) denote the smallest possible size of a weak (respectively, strong) $\mathcal{W}_\eta$ backdoor. Note that $\mathsf{wb}_g(\phi, \eta) \leq \mathsf{wb}_{tw}(\phi, \eta)$, and $\mathsf{sb}_g(\phi, \eta) \leq \mathsf{sb}_{tw}(\phi, \eta)$, since every backdoor to $\mathcal{W}_t$ is also a backdoor to $\mathcal{K}_t$ (although the converse is not necessarily true).

The *satisfiability* problem (SAT) of a CNF formula ϕ is to decide whether F has a satisfying truth assignment.

2.2 Parameterized algorithms and kernels.

A parameterized problem Π is a subset of $\Gamma^* \times \mathbb{N}$ for some finite alphabet Γ. An instance of a parameterized problem consists of (x, k), where k is called the parameter. We assume that k is *given in unary* and hence $k \leq |x|$. A central notion in parameterized complexity is *fixed parameter tractability (FPT)* which means, for a given instance (x, k), solvability in time $f(k) \cdot p(|x|)$, where f is an arbitrary function of k and p is a polynomial in the input size. The notion of *kernelization* is formally defined as follows.

Definition 1. [**Kernelization**] *Let* $\Pi \subseteq \Gamma^* \times \mathbb{N}$ *be a parameterized problem and* g *be a computable function. We say that* Π admits a kernel of size g *if there exists an algorithm* K*, called* kernelization algorithm*, or, in short, a* kernelization*, that given* $(x, k) \in \Gamma^* \times \mathbb{N}$*, outputs, in time polynomial in* $|x| + k$*, a pair* $(x', k') \in \Gamma^* \times \mathbb{N}$ *such that*

(a) $(x, k) \in \Pi$ *if and only if* $(x', k') \in \Pi$*, and*

(b) $\max\{|x'|, k'\} \leq g(k)$.

When $g(k) = k^{\mathcal{O}(1)}$ *or* $g(k) = \mathcal{O}(k)$ *then we say that* Π admits a polynomial *or* linear kernel *respectively. If additionally* $k' \leq k$ *we say that the kernel is* strict.

A generalization of the notion of kernelization as described above is the following.

Definition 2. [**Polynomial Compression**] *A* polynomial compression *of a parameterized language* $Q \subseteq \Sigma^* \times \mathbb{N}$ *into an unparameterized language* $R \subseteq \Sigma^*$ *is an algorithm that takes as input an instance* $(x, k) \in \Sigma^* \times \mathbb{N}$*, works in time polynomial in* $|x| + k$*, and returns a string* y *such that:*

1. $|y| \leq p(k)$ *for some polynomial* $p(\cdot)$*, and*
2. $y \in R$ *if and only if* $(x, k) \in Q$.

In case $|\Sigma| = 2$*, the polynomial* $p(\cdot)$ *will be called the* bitsize *of the compression.*

To show the non-existence of polynomial compression algorithms, we first need the following definitions.

Definition 3. [**Polynomial Equivalence Relation**] *An equivalence relation* $\mathcal{R}$ *on the set* Σ^* *is called a* polynomial equivalence relation *if the following conditions are satisfied:*

1. *There exists an algorithm that, given strings* $x, y \in \Sigma^*$*, resolves whether* $x \equiv_{\mathcal{R}} y$ *in time polynomial in* $|x| + |y|$.
2. *Relation* $\mathcal{R}$ *restricted to the set* $\Sigma^{\leq n}$ *has at most* $p(n)$ *equivalence classes, for some polynomial* $p(\cdot)$.

Definition 4. [**Cross Composition**] *Let* $L \subseteq \Sigma^*$ *be an unparameterized language and* $Q \subseteq \Sigma^* \times \mathbb{N}$ *be a parameterized language. We say that* L cross-composes *into* Q *if there exists a polynomial equivalence relation* $\mathcal{R}$ *and an algorithm* $\mathcal{A}$*, called the* cross-composition*, satisfying the following conditions. The algorithm* $\mathcal{A}$ *takes on input a sequence of strings* $x_1, x_2, \ldots, x_t \in \Sigma^*$ *that are equivalent with respect to* $\mathcal{R}$*, runs in time polynomial in* $\sum_{i=1}^{t} |x_i|$*, and outputs one instance* $(y, k) \in \Sigma^* \times \mathbb{N}$ *such that:*

It turns out that if a problem admits a cross-composition algorithm for some polynomial equivalence relation, then it does not admit a polynomial compression unless NP $\subseteq$ coNP/ poly.

Theorem. *Assume that an* NP*-hard language* L *cross-composes to a parameterized language* Q*. Then* Q *does not admit a polynomial compression, unless* NP $\subseteq$ coNP/ poly*.*

2.3 Treewidth.

Let G be a graph. A *tree decomposition* of G is a pair $(T, \mathcal{X} = \{X_t\}_{t \in V(T)})$ where T is a tree and $\mathcal{X}$ is a collection of subsets of $V(G)$ such that:

- $\forall e = uv \in E(G), \exists t \in V(T) : \{u, v\} \subseteq X_t$ and
- $\forall v \in V(G),\ T[\{t \mid v \in X_t\}]$ is a non-empty connected subtree of T.

We call the vertices of T *nodes* and the sets in $\mathcal{X}$ *bags* of the tree decomposition $(T, \mathcal{X})$. 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.

A *nice tree decomposition* is a pair $(T, \mathcal{X})$ where $(T, \mathcal{X})$ is a tree decomposition such that T is a rooted tree and the following conditions are satisfied:

- Every node of the tree T has at most two children;
- if a node t has two children t_1 and t_2, then $X_t = X_{t_1} = X_{t_2}$; and
- if a node t has one child t_1, then either $|X_t| = |X_{t_1}| + 1$ and $X_{t_1} \subset X_t$ (in this case we call t_1 *insert node*) or $|X_t| = |X_{t_1}| - 1$ and $X_t \subset X_{t_1}$ (in this case we call t_1 *insert node*).

It is possible to transform a given tree decomposition $(T, \mathcal{X})$ into a nice tree decomposition $(T', \mathcal{X}')$ in time $O(|V| + |E|)$ [3]. A treewidth-t modulator for a graph G is a subset X such that the treewidth of $G \setminus X$ is at most t.

2.4 t-Boundaried graphs and Gluing.

A t-boundaried graph is a graph G and a set $B \subset V(G)$ of size at most t with each vertex $v \in B$ having a label $\ell_G(v) \in \{1, \ldots, t\}$. Each vertex in B has a unique label. We refer to B as the *boundary* of G. For a t-boundaried G the function $\delta(G)$ returns the boundary of G. When it is clear from the context, will often use the notation (G, B) to refer to a t-boundaried graph G with boundary B.

Let $G = (X \cup C, E, \ell)$ be a t-boundaried incidence graph with boundary $B = \{b_1, \ldots, b_t\} \subseteq (X \cup C)$, where b_i is the vertex in the boundary whose label is i. We use $\vartheta(G)$ to denote the characteristic of (G, B), which is the t-length word over $\{0, 1\}$ given by:

$$\vartheta(G)[i] = \begin{cases} 0 & \text{if } b_i \in X, \\ 1 & \text{if}, b_i \in C. \end{cases}$$

We say that two t-boundaried incidence graphs G_1 and G_2 have the same *boundary type* if $\vartheta(G_1) = \vartheta(G_2)$. Incidence graphs with the same boundary type can be "glued" together to result in a new incidence graph, which we denote by $G_1 \oplus G_2$. The gluing operation takes the disjoint union of G_1 and G_2 and identifies the vertices of $\delta(G_1)$ and $\delta(G_2)$ with the same label. If there are vertices $u_1,\ v_1 \in \delta(G_1)$ and $u_2,\ v_2 \in \delta(G_2)$ such that $\ell_{G_1}(u_1) = \ell_{G_2}(u_2)$ and $\ell_{G_1}(v_1) = \ell_{G_2}(v_2)$ then G has vertices u formed by unifying u_1 and u_2 and v formed by unifying v_1 and v_2. The new vertices u and v are adjacent if $u_1v_1 \in E(G_1)$ or $u_2v_2 \in E(G_2)$. Note that the graph G is also an incidence graph.

The *boundaried gluing operation* $\oplus_\delta$ is similar to the normal gluing operation, but results in a t-boundaried graph rather than a graph. Specifically $G_1 \oplus_\delta G_2$ results in a t-boundaried graph where the graph is $G = G_1 \oplus G_2$ and a vertex is in the boundary of G if it was in the boundary of G_1 or G_2. Vertices in the boundary of G keep their label from G_1 or G_2. Both for gluing and boundaried gluing we will refer to $G_1 \oplus G_2$ or $G_1 \oplus_\delta G_2$ as the *sum* of G_1 and G_2, and G_1 and G_2 are the *terms* of the sum.

Now, let (G_1, B_1) and (G_2, B_2) be two t-boundaried incidence graphs of order d. Let b_i^h denote the vertex in B_h with label i, for $h \in \{1, 2\}$. We say that G_1 and G_2 are *friendly* if they have the same boundary type, and further, for every $i \in [t]$ such that $b_i^1 \in \mathsf{cla}(B_1)$ and $b_i^2 \in \mathsf{cla}(B_2)$, we have that $d_{G_1}(b_i^1) + d_{G_2}(b_i^2) \leq d$. Notice that when two friendly t-boundaried incidence graphs of order d are glued together, the resulting graph is also an incidence graph of order d.

2.5 Minors

Given an edge $e = xy$ of a graph G, the graph G/e is obtained from G by contracting the edge e, that is, the endpoints x and y are replaced by a new vertex v_{xy} which is adjacent to the old neighbors of x and y (except from x and y). A graph H obtained by a sequence of edge-contractions is said to be a *contraction* of G. We denote it by $H \leq_c G$. A graph H is a *minor* of a graph G if H is the contraction of some subgraph of G and we denote it by $H \leq_m G$. We say that a graph G is *H-minor-free* when it does not contain H as a minor. We also say that a graph class $\mathcal{G}$ is *H-minor-free* (or, excludes H as a minor) when all its members are H-minor-free. It is well-known [22] that if $H \leq_m G$ then $tw(H) \leq tw(G)$. We will also use the fact that every graph of treewidth at least η^{100} contains the $(\eta \times \eta)$ grid as a minor [4]. We also use $\boxplus_\eta$ to denote the $(\eta \times \eta)$ grid.

Definition 5. *Let G_1 and G_2 be two graphs, and let t be a fixed positive integer. For $i \in \{1, 2\}$, let f_{G_i} be a function that associates with every vertex of $V(G_i)$ some subset of $[t]$. The image of a vertex $v \in G_i$ under f_{G_i} is called the label of that vertex. We say that that G_1 is* label-wise isomorphic *to G_2, and denote it by $G_1 \cong_t G_2$, if there is an map $h : V(G_1) \to V(G_2)$ such that (a) h is one to one and onto; (b) $(u, v) \in E(G_1)$ if and only if $(h(u), h(v)) \in E(G_2)$ and (c) $f_{G_1}(v) = f_{G_2}(h(v))$. We call h a* label-preserving isomorphism.

Notice that the first two conditions of Definition 5 simply indicate that G_1 and G_2 are isomorphic. Now, let G be a t-boundaried graph, that is, G has t distinguished vertices, uniquely labeled from 1 to t. Given a t-boundaried graph G, we definition a canonical labeling function $\mu_G : V(G) \to 2^{[t]}$. The function μ_G maps every distinguished vertex v with label $\ell \in [t]$ to the set $\{\ell\}$, that is $\mu_G(v) = \{\ell\}$, and for all vertices $v \in (V(G) \setminus \partial(G))$ we have that $\mu_G(v) = \emptyset$.

Next we definition a notion of labeled edge contraction. Let H be a graph together with a function $f_H : V(H) \to 2^{[t]}$ and $(u, v) \in E(H)$. Furthermore, let H' be the graph obtained from H by identifying the vertices u and v into w_{uv}, removing all the parallel edges and removing all the loops. Then by *labeled edge contraction* of an edge (u, v) of a graph H, we mean obtaining a graph H' with the label function $f_{H'} : V(H') \to 2^{[t]}$. For $x \in V(H') \cap V(H)$ we have that $f_{H'}(x) = f_H(x)$ and for w_{uv} we definition $f_{H'}(w_{uv}) = f_H(u) \cup f_H(v)$. Now we introduce a notion of labeled minors of a t-boundaried graph.

Definition 6. *Let H be a graph together with a function $f : V(H) \to 2^{[t]}$ and G be a t-boundaried graph with canonical labeling function μ_G. A graph H is called a labeled minor of G, if we can obtain a labeled isomorphic copy of H from G by performing edge deletion and labeled edge contraction.*

Remark 1. *We note that the notion of a label-preserving isomorphism for graphs depends only on the labeling function, and is oblivious to the boundary. In particular, if G and H are two labeled t-boundaried graphs that are label-wise isomorphic, a label preserving isomorphism is not required to necessarily map the boundary vertices of G to boundary vertices of H.*

Finally, we definition the notion of *h-folios* and equivalence on t-boundaried graphs.

Definition 7. *A h-folio of a t-boundaried graph G is the set $\mathcal{M}_h(G)$ of all t-labeled minors of G on at most h vertices.*

2.6 Protrusions and Protrusion Replacement

For a graph G and $S \subseteq V(G)$, we define $\partial_G(S)$ as the set of vertices in S that have a neighbor in $V(G) \setminus S$. For a set $S \subseteq V(G)$ the *neighbourhood* of S is $N_G(S) = \partial_G(V(G) \setminus S)$. When it is clear from the context, we omit the subscripts. A *r-protrusion* in a graph G is a set $X \subseteq V$ such that $|\partial(X)| \leq r$ and $\mathsf{tw}(G[X]) \leq r$. Further, a *(r, s)-protrusion* is a set $X \subseteq V$ such that $|\partial(X)| \leq r$ and $\mathsf{tw}(G[X]) \leq s$. If G is a graph containing a r-protrusion X and X' is a r-boundaried graph, the act of *replacing* X by X' means replacing G by $G^{\partial(X)}_{V(G)\setminus X} \oplus X'$.

Let G be the incidence graph of a formula F. A *variable r-protrusion* in G is a r-protrusion such that all the vertices in $\partial(X)$ correspond to variables of F.

A *protrusion replacer* for a parameterized graph problem Π is a family of algorithms, with one algorithm for every constant r. The r'th algorithm has the following specifications. There exists a constant r' (which depends on r) such that given an instance (G, k) and an r-protrusion X in G of size at least r', the algorithm runs in time $O(|X|)$ and outputs an instance (G', k') such that $(G', k') \in \Pi$ if and only if $(G, k) \in \Pi$, $k' \leq k$ and G' is obtained from G by replacing X by a r-boundaried graph X' with less than r' vertices. Observe that since X has at least r' vertices and X' has less than r' vertices this implies that $|V(G')| < |V(G)|$.

3 Algorithms for d-SAT using $\mathcal{K}_\eta$-Backdoors

This section begins with the algorithms for solving SAT, which are described assuming the existence of algorithms that we call *reducers*. Subsequently, we unravel the construction of the reducers. The following is a more detailed overview.

The Algorithms In the first subsection, we present our algorithms for d-SAT assuming the existence of certain preprocessing subroutines, referred to as *reducers*. We begin by presenting a linear time randomized algorithm for d-SAT parameterized by the size of a weak $\mathcal{K}_\eta$-backdoor set. We then give a deterministic version of this algorithm while still managing to achieve the optimal asymptotic dependence on the parameter and the formula size. Following this, we present our algorithm for d-SAT parameterized by the size of a strong $\mathcal{K}_\eta$-backdoor set. We conclude this subsection with a proof of the fixed parameter tractability of computing weak $\mathcal{W}_\eta$-backdoor sets.

The Reducers In the second subsection, we give a description of the reducers assuming the ability to both find and replace protrusions in linear time. The algorithm that we use to detect protrusions has to find sufficiently many protrusions in linear time and will be developed in a more general form in the next section.

The Replacers In the final subsection, we give algorithms that replace protrusions while preserving satisfiability as well as the existence of small backdoor sets. As known theorems based on the notion of Finite Integer Index are not applicable to this problem, we have to develop problem specific protrusion replacers.

The (Improved) Fast Protrusion Replacers In Section 4, we present our linear time algorithm to detect protrusions that cover a sufficiently part of a given graph. This algorithm is developed in a very general setting and can be directly invoked to improve several existing kernelization as well as FPT results.

3.1 The Algorithms

The basis of the randomized (and subsequently, the deterministic) algorithms is the fact that the reducers ensure that the vertices corresponding to backdoors are always incident with a large fraction of the edges in the incident graph. This property is formalized by the following definition.

Definition 8. *Let G be a graph and let $S \subseteq V(G)$. Also, let $0 < \rho < 1$. We call S a ρ-cover for G if $\sum_{v \in S} d(v) \geq \rho \sum_{v \in V(G)} d(v)$. Let ϕ be a d-CNF formula and $S \subseteq \mathsf{var}(\phi)$. We call S a ρ-cover for ϕ if $N_{\mathsf{inc}(\phi)}[S]$ is a ρ-cover for the graph $\mathsf{inc}(\phi)$.*

Next, we formalize the properties of the algorithms that we refer to as reducers.

Definition 9. *Let $\eta \geq 1$ and $0 < \rho < 1$ be constants and $\mathcal{Q}$ a class of d-CNF formulas. A $(\mathsf{wb}, \mathcal{Q}, \rho)$-reducer is a pair of algorithms $(\mathcal{A}, \mathcal{A}')$ such that $\mathcal{A}$ takes as input a d-CNF formula ϕ and returns a d-CNF formula ϕ' and $\mathcal{A}'$ takes as input a truth assignment τ' to ϕ' and returns a truth assignment τ to ϕ such that*

- $|\phi'| \leq |\phi|$.
- *ϕ has a weak $\mathcal{Q}$-backdoor set of size at most k if and only if ϕ' has a weak $\mathcal{Q}$-backdoor set of size at most k for every $0 \leq k \leq |\mathsf{var}(\phi)|$.*
- *every set of variables which forms a weak $\mathcal{Q}$-backdoor set for the formula ϕ' is a ρ-cover of ϕ'.*
- *if τ' is a satisfying assignment for ϕ' then τ is a satisfying assignment for ϕ.*

A $(\mathsf{sb}, \mathcal{Q}, \rho)$-reducer is defined analogously with respect to strong $\mathcal{Q}$-backdoor sets along with the additional property that the formula ϕ' computed by $\mathcal{A}$ is explicitly required to be equivalent to ϕ.

3.1.1 d-SAT parameterized by weak $\mathcal{K}_\eta$-backdoor sets

We now turn to the descriptions of the algorithms. We first present an accessible description of a randomized algorithm for d-SAT when parameterized by the size of weak $\mathcal{K}_\eta$-backdoor sets. Subsequently, we show that there is also a single-exponential deterministic algorithm.

Lemma 1. *Let $(\mathcal{A}_1, \mathcal{A}_1')$ be a $(\mathsf{wb}, \mathcal{K}_\eta, \rho)$-reducer. Then, Algorithm* **Randomized-FPT-SAT-Weak** *(Figure 1) on input ϕ and an integer k, runs in time $\mathcal{O}(k(|\phi| + T_{\mathcal{A}_1}(|\phi|) + T_{\mathcal{A}_1'}(|\phi|)))$. Furthermore,*

- *If ϕ has a weak $\mathcal{K}_\eta$-backdoor set of size at most k then with probability at least $(\frac{\rho}{2d})^k$, the algorithm computes a satisfying assignment of ϕ.*
- *Correctly concludes that ϕ has no weak $\mathcal{K}_\eta$-backdoor set of size at most k otherwise.*

Randomized-FPT-SAT-weak(ϕ,k)

$\phi_0 := \phi$, $i := 0$

While ($\mathsf{inc}(\phi_i)$ has treewidth more than η^{100}) proceed as follows:

1. If $k \leq 0$ return that ϕ_i does not have a k-sized weak $\mathcal{K}_\eta$-backdoor set .
2. Execute algorithm $\mathcal{A}_1$ on ϕ_i and obtain an equivalent formula ϕ'_i.
3. Pick an edge $e \in E(\mathsf{inc}(\phi'_i))$ uniformly at random. Let x_i be the variable endpoint of e.
4. Select $\alpha_i \in \{0, 1\}$ uniformly at random.
5. Set $\phi_{i+1} := \phi'_i[x_i = \alpha_i]$, $k := k - 1$, $i := i + 1$.

Solve satisfiability of ϕ_i using a bounded-treewidth sub-solver. If unsatisfiable, simply return the answer. If satisfiable, compute a satisfying assignment of ϕ_i and recover a satisfying assignment for ϕ using $\mathcal{A}'_1$.

Figure 1: Algorithm **Randomized-FPT-SAT-Weak**

Proof. Since each iteration of this algorithm is dominated by the time required to run the $(\mathsf{wb}, \mathcal{K}_\eta, \rho)$-reducer on input ϕ and there are at most k possible iterations, the bound on the running time of the algorithm follows. It is clear that if ϕ has no weak $\mathcal{K}_\eta$-backdoor set of size at most k, then the algorithm correctly concludes that there is not such backdoor set. Therefore, we only need to consider the case when a smallest weak $\mathcal{K}_\eta$-backdoor set for ϕ, say S, has size at most k. Observe that in this case, ϕ is satisfiable, and a satisfying assignment may be obtained by using a bounded-treewidth sub-solver (note that $\mathsf{inc}(\phi)$ does not have $\mathcal{K}_\eta$ as a minor and hence its treewidth is bounded by at most η^{100}). We now prove the following claim regarding a run of the algorithm on the input (ϕ, k).

Claim 1. *For each $0 \leq i < k$, with probability at least $(\frac{\rho}{2d})^{i+1}$ the following events occur.*

1. $\phi_{i+1} \equiv \phi$,

2. ϕ_{i+1} has a weak $\mathcal{K}_\eta$-backdoor set of size at most $k - (i + 1)$.

Proof. The proof is by induction on i. Consider the base case when $i = 0$. Let S_0 be a smallest weak $\mathcal{K}_\eta$-backdoor set for ϕ'_0. Since S_0 is a ρ-cover for ϕ'_0, we have that S_0 is a $\frac{\rho}{d}$-cover for $\mathsf{inc}(\phi'_0)$. Therefore, the probability of choosing an edge incident on S_0 is at least $\frac{\rho}{d}$ which is also a lower bound on the probability that $x_0 \in S_0$. Furthermore, algorithm $\mathcal{A}_1$ by definition guarantees that ϕ'_0 is equivalent to ϕ_0 and that $|S_0| \leq k$. Therefore, let $\tau^*_0 : \mathsf{var}(\phi) \to \{0, 1\}$ be a satisfying assignment of ϕ'_0 such that $\phi'_0[\tau^*_0|_{S_0}]$ is in $\mathcal{K}_\eta$. Observe that $\phi'_0[\tau^*_0|_{x_0}]$ is also satisfiable and furthermore, $S_0 \setminus \{x_0\}$ is a weak $\mathcal{K}_\eta$-backdoor set for $\phi'_0[\tau^*_0|_{x_0}]$. Finally, the probability that $\phi_1 = \phi'_0[\tau^*_0|_{x_0}]$ is the probability that $\alpha_0 = \tau^*|_{x_0}$, which is at least $\frac{1}{2}$. Therefore, we conclude that with probability at least $\frac{\rho}{2d}$, $\phi_1 \equiv \phi_0$ and ϕ_1 has a weak $\mathcal{K}_\eta$-backdoor set of size at most $k - 1$.

We now move on to the induction step for $i \geq 1$. Algorithm $\mathcal{A}_1$ by definition guarantees that ϕ'_i is equivalent to ϕ_i. Furthermore, by the induction hypothesis, with probability at least $(\frac{\rho}{2d}^{i+1}$, we have that $\phi_i \equiv \phi$, implying that ϕ_i is satisfiable. Therefore, ϕ'_i is satisfiable. Also, we have

Algorithm 1: SolveWB((ϕ, k))

Input: (ϕ, k), where ϕ is a d-CNF formula, and k is a positive integer.
Output: FAIL if ϕ has no weak $\mathcal{K}_\eta$-backdoor set of size at most k, a satisfying assignment τ^* otherwise.

if $k = 0$ *and* $\phi \notin \mathcal{K}_\eta$*:* **then**
  return FAIL
if $k = 0$ *and* $\phi \in \mathcal{K}_\eta$*:* **then**
  Solve SAT for ϕ in polynomial time.
  if *ϕ has a satisfying assignment τ^*:* **then**
    return τ^*
  if *ϕ is not satisfiable:* **then**
    return FAIL
if $k > 0$ **then**
  $\phi' := \mathcal{A}(\phi)$
  Compute the buckets $B_1, \ldots, B_{\lceil \log n \rceil}$ for ϕ'
  for *each big bucket B_i* **do**
    for *each subset $S \subseteq B_i$ such that $|S| \geq \ell \cdot |B_i|$* **do**
      for *each assignment $\tau : S \to \{0, 1\}$* **do**
        $\mathcal{R}$:=SolveWB($\phi'[\tau], k - |S|$)
        if *$\mathcal{R}$ is not* FAIL **then return** $\tau^* \cup \tau$
      end
    end
  end
  return FAIL

that ϕ_i has a weak $\mathcal{K}_\eta$-backdoor set of size at most $k - i$. Therefore, algorithm $\mathcal{A}_1$ guarantees that ϕ' has a weak $\mathcal{K}_\eta$-backdoor set, say S_i of size at most $k - i$. Let $\tau_i^* : \mathsf{var}(\phi_i) \to \{0, 1\}$ be a satisfying assignment of ϕ_i' such that $\phi_i'[\tau_i^*|_{S_i}] \in \mathcal{K}_\eta$. Since S_i is a $\frac{\rho}{d}$-cover of $\mathsf{inc}(\phi_i')$, the probability that $x_i \in S_i$ is at least $\frac{1}{\rho d}$. Also $\phi_i'[\tau_i^*|_{x_i}]$ is satisfiable and $S_i \setminus \{x_i\}$ is a weak $\mathcal{K}_\eta$-backdoor set for $\phi_i'[\tau_i^*|_{x_i}]$. Since $\alpha_i = \tau_i^*|_{x_i}$ with probability at least $\frac{1}{2}$, we conclude that $\phi_{i+1} = \phi_i'[\tau_i^*|_{x_i}]$ and therefore both events occur with probability at least $(\frac{\rho}{2d})^{i+1}$. This completes the proof of the claim. ☐

Given the above claim, it follows that with probability at least $(\frac{\rho}{2d})^k$, for some $\ell \leq k$, the formula $\phi_\ell \in \mathcal{K}_\eta$ and $\phi_\ell \equiv \phi$. Since ϕ is satisfiable, so is ϕ_ℓ and this is correctly detected by a bounded-treewidth sub-solver. Finally, we can compute the satisfying assignment for ϕ by starting with a satisfying assignment for ϕ_ℓ and applying the algorithm $\mathcal{A}_1'$ iteratively to the formulas $\phi_\ell, \phi_{\ell-1}, \ldots, \phi_1$. This completes the proof of the lemma. ☐

We now give a deterministic version of the above algorithm. Our branching strategy is based on the intuition that a subset of vertices that covers a constant fraction of all the edges in G must contain sufficiently many vertices of high degree. Equivalently, a set of variables that form a ρ-cover must contain some variables that occur with a substantial frequency among the clauses of ϕ. We use a partition of the variables according to frequency that formalizes this intuition, which is based on the definitions in [7]. Indeed, our branching algorithm is exactly along the same lines; however, we present the details here for the sake of completeness.

Lemma 2. *Let $(\mathcal{A}, \mathcal{A}')$ be a $(\mathsf{wb}, \mathcal{K}_\eta, \rho)$-reducer. Then, there is a deterministic algorithm that takes as input ϕ and an integer k, runs in time $2^{\mathcal{O}(k)}(|\phi| + T_\mathcal{A}(|\phi|) + T_{\mathcal{A}'}(|\phi|))$ and*

- *either finds a satisfying assignment of ϕ, or*
- *concludes correctly that ϕ has no weak $\mathcal{K}_\eta$-backdoor set of size at most k.*

Proof. We first execute the algorithm $\mathcal{A}$ on input ϕ (see Algorithm 1) to obtain a formula ϕ' such that ϕ has a weak $\mathcal{K}_\eta$-backdoor set of size at most k if and only if ϕ' has a weak $\mathcal{K}_\eta$-backdoor set of size at most k and furthermore, any $S \subseteq \mathsf{var}(\phi')$ which is a weak $\mathcal{K}_\eta$-backdoor set is a ρ-cover of ϕ' for some constant $\rho < 1$. Let $G = \mathsf{inc}(\phi')$. The branching strategy is based on a partition the variables of ϕ' into sets, called *buckets*, which are defined as follows. For each $i \geq 1$, we let:

$$B_i = \left\{ v \in V(G) | \frac{n}{2^i} < d(v) \leq \frac{n}{2^{i-1}} \right\}.$$

Fix constants μ and ℓ such that $\frac{(4\ell+3\mu)}{2} < \frac{\rho}{d}$ and let X be a fixed smallest weak $\mathcal{K}_\eta$-backdoor set. We call a bucket B_i *big* if $|B_i| > i\mu$ and we call it *good* if $|B_i \cap N_G[X]| \geq \ell |B_i|$. We compute the buckets, and for each big bucket B_i, for every subset S of B_i of size at least $\ell|B_i|$, for every partial truth assignment τ to the variables in S, we recurse on the instance $(\phi'[\tau], k - |S|)$. We return that ϕ is satisfiable if for some bucket B_i and some subset S and some assignment τ, the recursion $(\phi'[\tau], k - |S|)$ returned it is satisfiable and we return that ϕ has no weak $\mathcal{K}_\eta$-backdoor set of size at most k otherwise. We now turn to the proof of correctness and analysis of running time.

Claim 2. *There is a bucket which is both big and good.*

Proof. Since X is a $\frac{\rho}{d}$-cover in $\mathsf{inc}(\phi')$, we have that $\sum_{v \in X} d(v) \geq \frac{\rho}{d} \cdot 2m$, where $m = |E(G)|$. If there were no buckets which are good as well as big, then we have the following. For the sake of contradiction, assume that ϕ does not have a bucket that is both big and good. Then, we have the following.

$$\sum_{v \in X} d(v) = \sum_{i=1}^{\log n} \sum_{v \in B_i \cap X} d(v)$$

$$= \sum_{\{i | B_i \text{is not good}\}} \sum_{v \in B_i \cap X} d(v) + \sum_{\{i | B_i \text{is not big}\}} \sum_{v \in B_i \cap X} d(v)$$

$$\leq \ell \cdot 4m + \sum_{\{i | B_i \text{is not big}\}} i\mu \cdot \left(\frac{n}{2^i} \right)$$

$$\leq \ell \cdot 4m + 3\mu n = 2dm \frac{4\ell + 3\mu}{2} < \frac{2m\rho}{d},$$

which contradicts that X is a $\frac{\rho}{d}$-cover. □

The correctness of the algorithm follows from the above claim and the exhaustiveness of the branching. We now analyze the running time. Suppose for the sake of analysis that all buckets are big, and let a_i be the size of bucket i. Then we have that

$$T(k) \leq \sum_{i=1}^{\log n} \binom{a_i}{k} T(k - \ell a_i) \leq \sum_{i=1}^{\log n} 2^{a_i} T(k - \ell a_i)$$

Assuming $T(k) = x^k$, substitute recursively to get:

$$T(k) \leq \sum_{i=1}^{\log n} 2^{a_i} x^{(k - \ell a_i)} \leq x^k \sum_{i=1}^{\log n} \left(\frac{2}{x^\ell} \right)^{a_i}$$

Algorithm 2: SolveSB$((\phi, k))$

Input: (ϕ, k), where ϕ is a d-CNF formula, and k is a positive integer.
Output: FAIL if ϕ has no strong $\mathcal{K}_\eta$-backdoors of size at most k, otherwise; YES if ϕ has a satisfying assignment, and NO if ϕ is not satisfiable.
Remark: See Figure 2 for the definition of $\curlywedge$ and $\curlyvee$.

if $k = 0$ *and* $\phi \notin \mathcal{K}_\eta$*:* **then**
    return FAIL
if $k = 0$ *and* $\phi \in \mathcal{K}_\eta$*:* **then**
    Solve SAT for ϕ in polynomial time.
    if ϕ *is satisfiable:* **then**
        return YES
    if ϕ *is not satisfiable:* **then**
        return NO
if $k > 0$ **then**
    Let $(\mathcal{A}, \mathcal{A}')$ be the $(\mathsf{sb}, \mathcal{K}_\eta, \rho)$ reducer as given by Lemma 8.
    Let ϕ^* denote the output of $\mathcal{A}(\phi)$.
    Compute the buckets $B_1, \ldots, B_{\lceil \log n \rceil}$ for ϕ^*.
    for *each big bucket* B_i **do**
        Let $\mathcal{S} := \{S \mid S \subseteq B_i \text{ and } |S| \geq \ell \cdot |B_i|\}$.
        for $S \in \mathcal{S}$ **do**
            Let $S := \{z_1, \ldots, z_b\}$
            Let $z[S] := \curlywedge_{\tau \in 2^S}$ **solveSB**$((\phi^*[\tau], k - b))$.
        end
        return $\curlyvee_S(z[S])$
    end

If $\frac{2}{x^\ell} < 1$ then each term of the sum is maximized when the exponent is as small as possible. We will choose x (based on ℓ) such that $\frac{2}{x^\ell} < 1$ holds. Since $a_i \geq \mu i$ for any big bucket we have that

$$T(k) \leq x^k \sum_{i=1}^{\log n} \left(\frac{2}{x^d}\right)^{\mu i}$$

The sum above is a geometric series and converges to a value that is at most 1 for $x = c$, for a suitably small choice of c depending only on ℓ and μ, which depended only on η. This bounds the running time by c^k. Further, if not all buckets are big the sum above should only be done over the big buckets, yielding the same result. □

3.1.2 d-SAT parameterized by strong $\mathcal{K}_\eta$-backdoor sets

We now introduce Algorithm 2, which is a deterministic algorithm that either determines the satisfiability of the input formula, or declares that the input formula has no strong $\mathcal{K}_\eta$-backdoor of size at most k. The overall branching strategy is rather similar to Algorithm 1, however, the manner in which the outputs of the recursive subroutines are merged is more intricate in this case, and there are subtle differences that will be apparent in the proof of correctness.

Lemma 3. *Let $(\mathcal{A}, \mathcal{A}')$ be a $(\mathsf{sb}, \mathcal{K}_\eta, \rho)$-reducer.Then, there is a deterministic algorithm that takes as input ϕ and an integer k, runs in time $2^{\mathcal{O}(k)}(|\phi| + T_{\mathcal{A}}(|\phi|) + T_{\mathcal{A}'}(|\phi|))$ and*

- *either finds a satisfying assignment of ϕ, or*

$\curlywedge$	Yes	No	Fail
Yes	Yes	Yes	Yes
No	Yes	No	Fail
Fail	Yes	Fail	Fail

$\curlyvee$	Yes	No	Fail
Yes	Yes	Yes	Yes
No	Yes	No	No
Fail	Yes	No	Fail

Figure 2: The functions $\curlyvee$ and $\curlywedge$.

- *reports correctly that ϕ is not satisfiable, or*
- *concludes correctly that ϕ has no $\mathcal{K}_\eta$ strong backdoor set of size at most k.*

Proof. As with the proof of Lemma 2, we begin by running the algorithm $\mathcal{A}$ on ϕ to ensure that ϕ is an equivalent instance where every strong backdoor set of size at most k is a ρ-cover for some constant ρ. We then classify the variables into different buckets according to their degree, and we will have, as before, that there is a bucket that is both large and also contributes a constant fraction of its vertices to the ρ-cover.

We remark that the analysis of the running time of the algorithm is identical to the analysis in Lemma 2. Therefore, we now focus on the proof of correctness for Algorithm 2. Note that the algorithm has three possible outputs; namely Yes, No, and Fail. We claim that if the algorithm reports Fail, then ϕ has no $\mathcal{K}_\eta$ strong backdoor set of size at most k. On the other hand, if the algorithm returns Yes (respectively, No), then ϕ has a satisfying assignment (respectively, is not satisfiable). We proceed by induction on k. In the base case, when $k = 0$, if $\phi \in \mathcal{K}_\eta$, then a small-treewidth sub-solver for SAT will correctly determine the satisfiability of ϕ, so the correctness of these outputs follow. On the other hand, if $\phi \notin \mathcal{K}_\eta$, then there is (by definition) no strong $\mathcal{K}_\eta$-backdoor of size k, and accordingly, the output is Fail.

Our induction hypothesis is that the output of the algorithm on (ϕ, ℓ) is correct for all values of $\ell \leq k$. Now, consider the behavior of the algorithm on $(\phi, k+1)$. By the correctness of the replacer algorithm, the formula ϕ^* has a strong $\mathcal{K}_\eta$-backdoor of size at most $(k+1)$ if and only if ϕ has a strong $\mathcal{K}_\eta$-backdoor of size at most k. The algorithm then proceeds to examine all subsets of size at most $(k+1)$ of the big buckets. We have the following cases.

- **SolveSB**$((\phi, k))$ = Yes. Observe that **solveSB**$((\phi, k+1))$ returns Yes if, and only if, there is a subset $S \in \mathcal{S}$ for which $z[S]$ was Yes. This in turn is true if, and only if, there is an assignment $\tau \in 2^S$ to the variables in S for which SolveSB$((\phi^*[\tau], k-b))$ returns Yes. Inductively, this implies that $\phi^*[\tau]$ is in fact a satisfiable formula. Let τ' then be a satisfying assignment for $\phi^*[\tau]$. Note that $\tau^*(x)$, given by:

$$\tau^*(x) = \begin{cases} \tau(x) & \text{if } x \in S, \\ \tau'(x) & \text{if}, x \notin S, \end{cases}$$

  is a satisfying assignment for ϕ^*. By the equivalence of ϕ and ϕ^* with respect to satisfiability (as guaranteed by the reducer) we know that ϕ is also satisfiable, concluding the proof.

- **SolveSB**$((\phi, k))$ = No. In this case, **solveSB**$((\phi, k+1))$ returns Yes if, and only if, there is a subset $S \in \mathcal{S}$ for which $z[S]$ was No. This in turn is true if, and only if, there for *every* assignment $\tau \in 2^S$, SolveSB$((\phi^*[\tau], k-b))$ returns No. Since we have, inductively, that $\phi^*[\tau]$ is not satisfiable for any assignment to the variables in S, we know that ϕ^* is also not satisfiable. Indeed, suppose to the contrary that ϕ^* does admit a satisfying assignment τ^*. Then the formula $\phi^*[\tau^*\restriction_S]$ would be satisfiable as well, which contradicts the induction hypothesis. The correctness again follows from the equivalence of ϕ and ϕ^* with respect to satisfiability.

– **SolveSB**$((\phi, k))$ = FAIL. Here, we have that **solveSB**$((\phi, k+1))$ returns FAIL if, and only if, for all subsets $S \in \mathcal{S}$, $z[S]$ is FAIL. In other words, for every subset $S \in \mathcal{S}$, there is an assignment τ_S to the variables of S for which SolveSB$((\phi^*[\tau_S], k+1-|S|))$ is FAIL. By the induction hypothesis, we have that the formulas $(\phi^*[\tau_S]$ do not admit a strong backdoors of size at most $k - |S|$ for every $S \in \mathcal{S}$.

  Suppose, for the sake of contradiction, ϕ^* does admit a strong $\mathcal{K}_\eta$-backdoor of size at most $k+1$. Since ϕ^* is the output of a $(\mathsf{wb}, \mathcal{K}_\eta, \rho)$-reducer, we know every $\mathcal{K}_\eta$-backdoor in ϕ^* is a ρ-cover, and therefore intersects a large fraction of one of the sets in $\mathcal{S}$. In particular, let S^* be a strong backdoor of size at most $(k+1)$ such that its intersection with B_i is at least $\ell|B_i|$. Then $S' := S^* \cap B_i \in \mathcal{S}$, and for any $\tau \in 2^{S'}$, we have that $\phi^*[\tau]$ does admit a strong backdoor of size at most $k+1-|S'|$, indeed, $S' \cap \mathsf{var}(\phi^*[\tau])$ would be such a strong backdoor. However, by the discussion above, there exists an assignment τ' to S' for which SolveSB$((\phi^*[\tau', k+1-|S'|))$ returns FAIL, contradicting the induction hypothesis. The correctness for $(\phi, k+1)$ follows from the equivalence of ϕ and ϕ^* with respect to having strong backdoors of size at most $(k+1)$, as guaranteed by the reducer.

Observe that the case analysis above is exhaustive, and establishes the correctness of Algorithm 2. □

We conclude at this point by observing that combining Lemmas 2 and 3 along with Lemma 8 (proved in the next subsection) gives us two algorithms – one for d-SAT parameterized by the size of a smallest weak $\mathcal{K}_t$-backdoor and one for d-SAT parameterized by the size of a smallest strong $\mathcal{K}_t$-backdoor. For any input (ϕ, k), we can in fact run both algorithms on the same input, giving us Theorem 1.

3.1.3 Fixed Parameterized Tractability of computing weak $\mathcal{W}_\eta$-backdoor sets

We also obtain the following fixed parameter tractability result (assuming appropriate reducers) for the problem of deciding if a given formula has a weak $\mathcal{W}_\eta$ backdoor set of size at most k.

Theorem 2. *There is an algorithm that, given a d-CNF formula ϕ and an integer k, runs in time $2^{\mathcal{O}(k)}|\phi|$ either returns a set of at most k variables which form a $\mathcal{W}_\eta$ weak backdoor set or correctly concludes that such a set does not exist.*

Proof. To prove the theorem, we repurpose the algorithm of Lemma 1, where instead of just fixing a random assignment to a randomly chosen variable, we now also add this variable to the $\mathcal{W}_\eta$ weak backdoor set. Therefore, it suffices to give a linear time $(\mathsf{wb}, \mathcal{W}_\eta, \rho)$-reducer, which is described formally in the next subsection. This gives a randomized $2^{\mathcal{O}(k)}|\phi|$ algorithm to detect weak $\mathcal{W}_\eta$-backdoor sets of size at most k. This algorithm can be derandomized identical to the way the algorithm of Lemma 1 is derandomized in Lemma 2. The correctness and running time bounds of this algorithm follow along the same lines as those for the algorithm of Lemma 2. □

In the next subsection, we describe the reducers whose existence was assumed in the above lemmas and show that they run in linear time.

3.2 Reducers for d-SAT

We begin by introducing a *satisfiability preserving* version of protrusion replacers for d-SAT. More formally, for $\mathcal{J} \in \{\mathsf{wb}, \mathsf{sb}\}$, a class of formulas $\mathcal{Q}$ and a constant r, we have the following definition of a $(\mathcal{J}, \mathcal{Q}, r)$-protrusion replacer.

Definition 10. *A* (wb, $\mathcal{Q}, r$)*-protrusion replacer is an algorithm that, given a d-CNF formula* ϕ *and a* r*-protrusion* X *in* $\mathsf{inc}(\phi)$*, runs in time* $\mathcal{O}(|X|)$ *and outputs a formula* ϕ *such that* $|\phi'| \leq |\phi|$, $\phi' \equiv \phi$*, and* ϕ *has a weak* $\mathcal{Q}$*-backdoor set of size at most* k *if and only if* ϕ' *has a weak* $\mathcal{Q}$*-backdoor set of size at most* k *for every* $0 \leq k \leq |\mathsf{var}(\phi)|$*. A* ($\mathsf{sb}, \mathcal{Q}, r$)*-protrusion replacer is defined analogously for strong backdoor sets.*

The following lemma, proved in the following subsection gives protrusion replacers that preserve both satisfiability as well as backdoor sets.

Lemma 4. *For every* η*, there is a constant* r *and algorithms* $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3$*, where* $\mathcal{R}_1$ *is a* ($\mathsf{wb}, \mathcal{K}_\eta, r$)*-protrusion replacer,* $\mathcal{R}_2$ *is a* ($\mathsf{sb}, \mathcal{K}_\eta, r$)*-protrusion replacer and* $\mathcal{R}_3$ *is a* ($\mathsf{wb}, \mathcal{W}_\eta, r$)*-protrusion replacer respectively.*

The statement of the following lemma is analogous to the general protrusion replacer theorem for graph problems proved in the next section. However, since we are dealing with CNF formulas and need to preserve satisfiability as well as backdoor sets, this replacer theorem cannot be directly applied to this problem and hence we require a "SAT" version of this theorem for our purposes here.

Lemma 5. *For a fixed* r*, let* $\mathcal{R}$ *be a* ($x, \mathcal{Q}, r$)*-protrusion replacer (*$x \in \{\mathsf{wb}, \mathsf{sb}\}$*) that replaces* r*-protrusions of size at least* r'*. Let* s *and* β *be constants such that* $s \geq r' \cdot 2^r$ *and* $r \geq 3(\beta + 1)$*. Given a formula* ϕ *as input, there is an algorithm that runs in time* $\mathcal{O}(|\phi|)$ *and produces an equivalent formula* ϕ' *with* $|\phi'| \leq |\phi|$*. If additionally* $\mathsf{inc}(\phi)$ *has a* (α, β)*-protrusion decomposition such that* $\alpha \leq \frac{|\phi|}{244s}$*, then we have that* $|\phi'| \leq (1 - \delta)|\phi|$ *for some constant* δ*.*

For the rest of this subsection, we assume the above lemmas with their proofs deferred to later parts of the paper. We now recall results from [2] relating ρ-covers and protrusions.

Lemma 6. [7] *Let* G *be a graph and* $S \subseteq V(G)$ *such that* $tw(G - S) \leq \eta$*. For any constant* $s > 1$*, let* $\rho = \frac{1}{488s(\eta+1)+2}$*. If* S *is not a* ρ*-cover, then* $|N[S]| \leq \frac{n}{(244s)}$*.*

We also need the notion of a protrusion decomposition defined in [2] where it was shown that if a graph G has a set X such that $\mathsf{tw}(G - X) \leq b$, for some fixed b, then it admits a protrusion decomposition for an appropriate value of the parameters.

Definition 11. [**Protrusion Decomposition**][**2**] *A graph* G *has an* (α, β, η)-protrusion decomposition *if* $V(G)$ *has a partition* $\mathcal{P} = \{R_0, R_1, \ldots, R_t\}$ *where*

- $\max\{t, |R_0|\} \leq \alpha$*,*
- *each* $N_G[R_i]$*,* $i \in \{1, \ldots, t\}$ *is a* (β, η)*-protrusion of* G*, and*
- *for all* $i > 1$*,* $N[R_i] \subseteq R_0$*.*

We call the sets $R_i^+ = N_G[R_i]$*,* $i \in \{1, \ldots, t\}$ protrusions *of* $\mathcal{P}$*. By an* (α, β)*-protrusion decomposition, we mean a* (α, β, β)*-protrusion decomposition.*

Lemma 7. [2] *Let* G *be a graph and* $S \subseteq V(G)$ *such that* $tw(G - S) \leq \eta$*. Then,* G *has a* ($4|N[S]|(\eta + 1), 2(\eta + 1), \eta$)*-protrusion decomposition*

We now prove the main lemma of this subsection – a description of the reducers.

Lemma 8. *For every* η*, there is a positive* $\rho < 1$ *and algorithms* $\mathcal{A}_1, \mathcal{A}'_1, \mathcal{A}_2, \mathcal{A}'_2$ *such that* ($\mathcal{A}_1, \mathcal{A}'_1$) *is a* ($\mathsf{wb}, \mathcal{K}_\eta, \rho$)*-reducer and* ($\mathcal{A}_2, \mathcal{A}'_2$) *is a* ($\mathsf{sb}, \mathcal{K}_\eta, \rho$)*-reducer.*

Proof. Let $\nabla_1 : \mathbb{N} \to \mathbb{N}$ and $\nabla_2 : \mathbb{N} \to \mathbb{N}$ be such that (r, η)-protrusions of size at least $\nabla_1(r)$ ($\nabla_2(r)$) can be reduced while preserving equivalence and weak $\mathcal{K}_\eta$-backdoor sets (respectively strong $\mathcal{K}_\eta$-backdoor sets). Let ϕ be a d-CNF formula, $G = \mathsf{inc}(\phi)$ and $n = |V(G)|$. Let $S \subseteq \mathsf{var}(\phi)$ be any weak (strong) $\mathcal{K}_\eta$-backdoor set for ϕ.

- Fix a such that $a > 4(\eta+1)8^{(2\eta+3)}\nabla_c(3(2\eta+3))$ where $c = 1$ for the case of weak backdoors and $c = 2$ for the case of strong backdoors.
- Set $\rho = \frac{1}{488a(\eta+1)+2}$.
- By Lemma 6, we have that if S is not already a ρ-cover, then $|N_G^2[S]| \le \frac{n}{(244a)}$.
- By Lemma 7, we have that G has a $(4|N_G^2[S]|(\eta+1), 2(\eta+1), \eta)$-protrusion decomposition. Setting $\alpha = 4|N_G^2[S](\eta+1)$, $s = \frac{a}{4(\eta+1)}$, $\beta = 2(\eta+1)$ and using the fact that $|N_G^2[S]| \le \frac{n}{(244a)}$ gives us the existence of a (α, β)-protrusion decomposition of G such that $\alpha \le \frac{n}{244s}$.
- Since $\alpha \le \frac{n}{244s}$, an application of the algorithm of Lemma 5 for $r = 3(\beta + 1)$ along with the appropriate protrusion replacer (Lemma 4) on ϕ results in a formula ϕ'. Then, we know that $\phi \equiv \phi'$, ϕ has a weak (strong) $\mathcal{K}_\eta$-backdoor set of size at most k if and only if ϕ' has a weak (strong) $\mathcal{K}_\eta$-backdoor set of size at most k for every $0 \le k \le |\mathsf{var}(\phi')|$. Furthermore, $|\phi'| \le |\phi|(1-\delta)$ for a constant $0 < \delta < 1$.

Combining the above, the algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ are easily described as follows. Combine the algorithm of Lemma 5 with the appropriate protrusion replacer to obtain a formula ϕ' and if $|\mathsf{inc}(\phi')| < |\mathsf{inc}(\phi)|(1-\delta)$ then repeat this process with ϕ' as input. Otherwise stop and output ϕ'.

The correctness of the algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ follows from the correctness of Lemma 6, Lemma 7 and Lemma 5, while the linear running time follows from the fact that the time required by the algorithm is bounded by a geometric series. It only remains to describe the algorithms $\mathcal{A}'_1$ and $\mathcal{A}'_2$. The descriptions as well as correctness of these algorithms are part of the proofs of Lemma 12 and Lemma 14 which are deferred to the next subsection. This completes the proof of this lemma. □

Finally, recall that we also needed to assume a $(\mathsf{wb}, \mathcal{W}_\eta, \rho)$-reducer in the proof Theorem 2. Observe that in order to describe a $(\mathsf{wb}, \mathcal{W}_\eta, \rho)$-reducer the descriptions in Lemma 8 can be replicated almost identically with the only changes appearing in the selection of the function ∇ which provides a lower bound on the size of replaceable protrusions and then in the selection of the appropriate protrusion replacer given by Lemma 4.

3.3 Protrusion Replacement

In this section we describe procedures that, given the incidence graph $(G = (X, C), E, \ell)$ of a d-CNF formula ϕ and protrusion of large enough size in G, obtains an equivalent instance that is strictly smaller than the original. The equivalence is both with respect to maintaining the satisfiability of $\psi(G)$, and preserving the presence of a strong or weak $\mathcal{K}_\eta$-backdoor (or a strong/weak $\mathcal{W}_\eta$-backdoor). We remark here that these methods described here similar in spirit to the ideas used in [13].

3.3.1 Weak $\mathcal{K}_\eta$ Backdoors

We begin by introducing some definitions. A *h-state* of a t-boundaried incidence graph G with boundary Z is a tuple $(P, Q, \tau, w, \mathfrak{H})$, where:

- P is a subset of $\mathsf{var}(Z)$,
- Q is a subset of $\mathsf{cla}(Z)$,
- τ is a truth assignment of $\mathsf{var}(Z)$, that is, $\tau : \mathsf{var}(Z) \to \{0,1\}$,
- $w \in [t]$ is a positive integer, and
- $\mathfrak{H}$ is a collection of labeled graphs on at most h vertices, where the labels are from $[t]$.

For a t-boundaried incidence graph G with boundary Z, let $\mathfrak{S}((G,Z),h)$ denote the set of all possible h-states of (G,Z). The *realizability function* of G with respect to h, denoted by Λ_h, maps pairs $((G,Z),\omega)$, to $\{0,1\}$, where $\omega \in \mathfrak{S}(G,h)$. This function is defined as follows. Let $\omega = (P,Q,\tau,w,\mathfrak{H}) \in \mathfrak{S}(G,h)$. We have that $\Lambda_h(\omega) = 1$ if, and only if, there is a subset $S^* \subseteq \mathsf{var}(G)$ of size at most w, and a truth assignment τ^* on $\mathsf{var}(G)$, such that the following holds:

- $S^* \cap Z = P$
- τ^* is a satisfying assignment for $\psi(G) \setminus Q$,
- $\tau^*\restriction_Z = \tau$, that is, for every $v \in Z$, $\tau^*(v) = \tau(v)$,
- $\mathfrak{H} = \mathcal{M}_h(H)$, where H is the incidence graph of $\psi(G)[\tau^*\restriction_{S^*}]$.

Consider two t-boundaried incidence graphs (G_1,Z_1) and (G_2,Z_2) that have the same boundary type. For $i \in [t]$ and $h \in \{1,2\}$, we use b_i^h to denote the vertex of Z_h with label i. Let $\jmath : Z_1 \to Z_2$ be defined as $\jmath(b_i^1) = b_i^2$. For any $Z \subseteq Z_1$, we let $\jmath(Z) := \bigcup_{v \in Z} \jmath(v)$. Further, for every h-state $\omega = (P,Q,\tau,w,\mathfrak{H}) \in \mathfrak{S}(G_1,h)$, we define $\omega' := (\jmath(P),\jmath(Q),\tau',w,\mathfrak{H}) \in \mathfrak{S}(G_2,h)$, where $\tau' : Z_2 \to \{0,1\}$ and $\tau'(b_i^2) := \tau(b_i^1)$.

We say that two t-boundaried incidence graphs (G_1,Z_1) and (G_2,Z_2) *equivalent with respect to h-states* if they have the same boundary type, and for every $\omega = (P,Q,\tau_P,w,\mathfrak{H}) \in \mathfrak{S}(G_1,h)$, we have that $\Lambda_h(\omega) = \Lambda_h(\omega^*)$.

Let G be an incidence graph of order d. Let $Y \subseteq V(G)$ be a (ζ,t)-protrusion in G with boundary Z, and let $(T,\mathcal{B} = \{B_v\}_{v \in V(T)})$ be a tree decomposition of $G[Y]$. For a node $v \in V(T)$, let $D_v := \{u \mid u \text{ is a descendant of } v \text{ in } T\}$. We note that we consider v to be a descendant of itself. We let $H_t := G[\bigcup_{u \in D_v} B_u]$, and $H_t^* := H_t \setminus B_t$.

For a bag B_v and a vertex $b \in B_v$, we let $d_v^{\uparrow}[b]$ denote the degree of b in $G \setminus H_v$ and we let $d_v^{\downarrow}[b]$ denote the degree of b in H_v. For two nodes $u,v \in T$, we say that H_u and H_v have the *same clause degree sequence* if the t-boundaried graphs (H_u,B_u) and (H_v,B_v) have the same boundary type, and for every vertex $b \in \mathsf{cla}(B_u)$, we have that $d_v^{\downarrow}[b] = d_u^{\downarrow}[\jmath(b)]$, and $d_v^{\uparrow}[b] = d_u^{\uparrow}[\jmath(b)]$.

We are now finally ready to describe the central notion that in the prelude to protrusion replacement — namely, redundancy. For a fixed h, we say that two bags B_u and B_v in the tree decomposition of Y are *h-redundant* if the corresponding t-boundaried graphs (H_u,B_u) and (H_v,B_v) have the same boundary type, the same clause degree sequence, are also equivalent with respect to h-states, and the graphs induced on the boundaries B_u and B_v are isomorphic.

We now give a brief informal overview of the procedure that we will follow from here. Let $h := t^2$. Suppose we find h-redundant bags H_u and H_v such that u is an ancestor of v. Then, consider the graph obtained by removing H_u^* and gluing on (H_v,B_v) to $(G \setminus H_u^*, B_u)$. It turns out that the resulting graph, has a weak-$\mathcal{K}_t$ backdoor of size at most k if, and only if, G has a weak-$\mathcal{K}_t$ backdoor of size at most k. Further, if u and v were introduce bags, then the number of vertices in the graph obtained as a result of the replacement that we described is strictly smaller than $|V(G)|$. After establishing the correctness of this reduction, we only need to show that there is a constant r' such that any protrusion on at least r' vertices indeed admits redundant introduce bags B_u and B_v in its tree decomposition, where u is an ancestor of v, which will complete our argument for being able to replace large protrusions.

Towards formulating a reduction rule and proving its correctness, we begin with the following lemma.

Lemma 9. *Let $\eta \in \mathbb{Z}^+$ be a constant, and let (G_1, Z_1) and (G_2, Z_2) be two t-boundaried incidence graphs that are equivalent with respect to h-states, where $h = \eta^2$. Further, suppose G_1 and G_2 do not admit the $\boxplus_\eta$ grid as a labeled minor. Then, for any t-boundaried graph (G_3, Z_3) that has the same boundary type as G_1 and G_2, and any positive integer k, we have that $\psi(G_1 \oplus G_3)$ has a weak $\mathcal{K}_\eta$-backdoor of size at most k if, and only if, $\psi(G_1 \oplus G_3)$ has a weak $\mathcal{K}_\eta$-backdoor of size at most k.*

Proof. In the forward direction, suppose $\psi(G_1 \oplus G_3)$ has a weak $\mathcal{K}_\eta$-backdoor, S^*, of size at most k, and let $\tau : \mathsf{var}(G_1 \oplus G_3) \to \{0, 1\}$ be the witness assignment for S^*. Let S_1, S and S_3 denote the intersection of S^* with $G_1 \setminus Z_1$, Z_1, and $G_3 \setminus Z_3$, respectively. Let $\mathfrak{T}$ denote the collection of labeled minors of size at most η^2 that can be obtained from $\mathsf{inc}(\psi(G_1)[\tau\restriction_{(S \cup S_1)}])$. Let Q be the set of clauses of $\mathsf{inc}(\psi(G_1)[\tau\restriction_{(S \cup S_1)}])$ that are not satisfied by $\tau\restriction_{V(G_1)}$.

We first note that $|S| + |S_1| \leq t$. Recall that G_1 does not contain any $\boxplus_\eta$ grid as a labeled minor. Therefore, $S^* \cap (G_3 \setminus Z_3) \cup Z_3$ is a weak $\mathcal{K}_\eta$-backdoor if S^* is a weak $\mathcal{K}_\eta$-backdoor (with the same witness assignments). Thus, $(S, Q, \tau\restriction_{Z_1}, |S| + |S_1|, \mathfrak{T}) \in \mathfrak{S}(G_1, Z_1)$. Further, observe that:

$$\Lambda_h((G_1, Z_1), (S, Q, \tau\restriction_{Z_1}, |S| + |S_1|, \mathfrak{T})) = 1$$

Let $\jmath$ be the function that maps boundary vertices of Z_1 of label i to boundary vertices of Z_2 of label i, and this function is extended to subsets of boundary vertices in the usual way. Also, for $h \in \{1, 2\}$, if $b_i^h \in Z_h$ is the vertex of Z_h with label i, then define:

$$(\tau\restriction_{Z_1})'(b_i^2) := (\tau\restriction_{Z_1})(b_i^1).$$

Since G_1 and G_2 are equivalent with respect to h-states, we have that:

$$\Lambda_h((G_2, Z_2), (\jmath(S), \jmath(Q), (\tau\restriction_{Z_1})', |S + S_1|, \mathfrak{T})) = 1,$$

By the definition of Λ_h, this implies that there is a subset S_2^* of $\mathsf{var}(G_2)$ of size at most $|S + S_1|$, and a truth assignment τ_2 on $\mathsf{var}(G_2)$, such that the following holds:

- $S^* \cap Z_2 = \jmath(S)$
- τ_2 is a satisfying assignment for $\psi(G_2) \setminus \jmath(Q)$,
- $\tau_2\restriction_{Z_2} = (\tau\restriction_{Z_1})'$,
- $\mathfrak{T} = \mathcal{M}_h(H)$, where H is the incidence graph of $\psi(G_2)[\tau\restriction_{\jmath(S)}]$.

Now we propose $T^* := S_3 \cup S_2^*$ as weak $\mathcal{K}_\eta$-backdoor for $G_2 \oplus G_3$, with the witness assignment τ_2^* given by:

$$\tau_2^*(v) = \begin{cases} \tau_2(v) & \text{if } v \in \mathsf{var}(G_2), \\ \tau(v) & \text{if, } v \in \mathsf{var}(G_3). \end{cases}$$

Note that the τ_2^* is well-defined because the assignments τ and τ_2 agree on the boundary.

Clearly, $|T^*| = |S_3| + |S_2^*| \leq |S_3| + |S_1| + |S| = |S|$, so T^* has the desired size. Next, we claim that τ_2^* as defined above satisfies all the clauses of $\psi(G_2 \oplus G_3)$. Indeed, consider a clause $C \in \mathsf{cla}(\psi(G_2 \oplus G_3))$.

- If C is a clause in $\psi(G_2 \oplus G_3 \setminus G_2)$, then $\mathsf{var}(C) \subseteq \mathsf{var}(G_3)$, and evidently $\tau_2^*(v) = \tau(v)$ for all $v \in C$. If this clause is not satisfied by τ_2^*, then it is also not satisfied by τ in $\psi(G_1 \oplus G_3)$, which is a contradiction.
- If C is a clause in $\psi(G_2) \setminus \jmath(Q)$, then it is satisfied by $\tau_2(v)$ by definition.
- Finally, let C be a clause in $\jmath(Q)$. Consider the clause C in $\psi(G_1 \oplus G_2)$, and let $v \in \mathsf{lit}(C)$ be such that $\tau(v) = 1$. Note that the v must belong to $\mathsf{var}(G_3) \setminus \mathsf{var}(G_1)$ — indeed, we know that every variable in $C \cap \mathsf{var}(G_1)$ evaluates to false under the assignment τ (by the definition of Q). Since the assignment τ_2^* as defined above agrees with the assignment of τ to v, the clause C is satisfied under τ_2^* as well.

Finally, we need to show that $H := \mathsf{inc}(\psi(G_2 \oplus G_3)[\tau_2^* \restriction_{T^*}])$ does not contain the $\boxplus_\eta$ as a minor. For the sake of contradiction, assume that H does contain a subgraph H^* that is a minor model of $\boxplus_t$. Let the components of H^* witnessing the vertices of $\boxplus_\eta$ be $\{C_{i,j} \mid 1 \leq i, j \leq \eta\}$. Consider the labeled graph $H_2^* := H^* \cap G_2$, where the identified vertices are simply given by $H^* \cap Z_2$. Further, let $H^\dagger$ be the labeled graph obtained from H_2^* by performing labeled contractions of the edges in $C_{i,j} \cap H_2^*$ for all $1 \leq i, j \leq \eta$. Note that $H^\dagger \in \mathfrak{T}$. However, now consider that $H^\dagger$ is also a labeled minor of $(\mathsf{inc}(\psi(G_1)[\tau \restriction_{S_1 \cup S}]), Z)$, by the definition of Λ_h. Therefore, the graph $(G_3, Z_3) \oplus H^\dagger$ is a minor of $\mathsf{inc}(\psi(G_1 \oplus G_3)[\tau \restriction_{S_1 \cup S}])$.

However, now contracting the edges in $C_{i,j} \cap G_3$, we have that $\boxplus_t$ is a minor of $\mathsf{inc}(\psi(G_1 \oplus G_3)[\tau \restriction_{S^*}])$. Note that the edges we are contracting here are available in the graph $\mathsf{inc}(\psi(G_1 \oplus G_3)[\tau \restriction_{S^*}])$, since the solution T^* and S^* are designed to agree completely on G_3, including the boundary (they agree on the boundary because of the by the definition of T^* and the fact that G_1 and G_2 have identical h-states). Note that this contradicts the assumption that S^* was a weak $\mathcal{K}_\eta$-backdoor for $(G_1 \oplus G_3)$. The argument in the reverse is symmetric. ☐

We now have the following theorem, which will lead us to formalizing the procedure of protrusion replacement.

Lemma 10. *Let G be an incidence graph of order d, and let Y be a (t, η)-protrusion. Let $(T, \mathcal{B} = \{B_v\}_{v \in V(T)})$ be a nice tree decomposition of $G[Y]$. If T admits two (η^2)-redundant introduce bags B_u and B_v, where u is an ancestor of v in T then:*

- *The graph $H := (G \setminus H_u^*, B_u) \oplus (H_v, B_v)$ is also an incidence graph of order d,*
- *$|V(H)| < |V(G)|$, and*
- *The instance $(\psi(H), k)$ has a weak $\mathcal{K}_\eta$-backdoor if, and only if, the instance $(\psi(G), k)$ has a weak $\mathcal{K}_\eta$-backdoor.*

Proof. Consider the t-boundaried graphs (H_u, B_v) and (H_v, B_v). Note that the graphs (H_u, B_u) and $(G \setminus H_u^*, B_u)$ are friendly by definition. Since the B_u and B_v are (η^2)-redundant, they have the same clause degree sequence. Therefore, it is immediate that (H_v, B_v) and $(G \setminus H_u^*, B_u)$ are also friendly, and the first claim follows.

Since B_u and B_v are both introduce bags in T, and u is an ancestor of v, we have that $H_u \setminus H_v$ is non-empty. Recall that H denotes $(G \setminus H_u^*, B_u) \oplus (H_v, B_v)$, the graph obtained as a result of the proposed replacement. Note that:

$$|V(H)| = |V(H_v)| + |V(G \setminus H_u)| = |V(H_v)| + |V(G)| - |V(H_u)| < |V(G)|.$$

For the last statement, let S be a $\mathcal{K}_\eta$ -backdoor of $\psi(G)$, of size at most k. Consider the t-boundaried graphs (H_u, B_u), (H_v, B_v). Because these graphs are η^2-redundant, they are equivalent with respect to η^2-states. Now, apply Lemma 9 with $(G_3, Z_3) := (G \setminus H_u^*, B_u)$, and $(G_1, Z_1) := (H_u, B_u)$ and $(G_2, Z_2) := (H_u, B_u)$ to obtain the statement of the theorem. □

Next, we show that a sufficiently large protrusion always has redundant bags as required by Theorem 10.

Lemma 11. *Let G be an incidence graph of order d, let Y be a (t, η)-protrusion. There is a constant c depending only on d, η and t such that if $|Y| > c$, then a nice tree decomposition of $G[Y]$ admits two η^2-redundant bags B_u and B_v, such that u is an ancestor of v.*

Proof. Let $h := \eta^2$. The proof relies on the fact that the set of h-states of the t-boundaried graph $(G[Y], Z)$ (where $Z := \partial(Y)$) is bounded by a function of t and η alone, and therefore, the number of possible realizability functions is also bounded by t and η. In this argument, we make no attempt to optimize the constants, and often the bounds are not the best possible.

Recall that a h-state is a tuple $(P, Q, \tau, \omega, \mathfrak{H})$, where P and Q are subsets of $\mathsf{var}(Z)$ and $\mathsf{cla}(Z)$, respectively, τ is a truth assignment of Z, $\omega \in [t]$, and $\mathfrak{H}$ is a collection of labeled graphs on at most h vertices. Let us denote the number of possible h-states of a t-boundaried graph (G, Z) by $\lambda_1(h, t)$. Note that:

$$\lambda_1(h, t) \leq 2^{|\mathsf{var}(Z)|} \cdot 2^{|\mathsf{cla}(Z)|} \cdot 2^{|Z|} \cdot t \cdot \iota(h, t) \leq 2^{2t} \cdot t \cdot \iota(h, t),$$

where $\iota(h, t)$ is the number of all possible labeled graphs on at most h vertices with labels from the set $[t]$. Note that $|Z| = t$.

Recall that incidence graphs are two-edge colored graphs where every edge gets exactly one color. Let $\lambda_2(n)$ denote the number of all possible two-edge colored graphs on n vertices where the vertices have labels from $[n]$. Since the set of two-edge colored graphs is isomorphic to the set of words of length $\binom{n}{2}$ over the alphabet $\{0, 1, 2\}$ (indicating the absence of an edge, the presence of an edge with color 1 and the presence of an edge with color 2, respectively), we have that $\lambda_2(t) \leq t! 3^{\binom{t}{2}}$.

Let $\lambda_3(t)$ denote the number of possible clause degree sequences of a set of t vertices of G. Since G is an incidence graph of order d, the total degree of any clause vertex is at most d. Therefore, $\lambda_3(t) \leq d^{2t}$, since a clause degree sequence is characterized by an ordered collection of t pairs of numbers, where each number is at most d.

Let $\lambda_4(t)$ denote the number of possible boundary types of a t-boundaried incidence graph. Clearly, $\lambda_4(t) \leq 2^t$. Now, let:

$$c^* := 2^{\lambda_1(h,t)} \cdot \lambda_2(h, t) \cdot \lambda_3(t) \cdot (\lambda_4(t) + 1),$$

and let $c := 2^{c^*}$. Now consider a (t, η)-protrusion Y on at least c vertices, and let T be a nice tree decomposition of $G[Y]$. Note that T has at least c introduce nodes. Further, since T is a binary tree, and a binary tree on $2^n - 1$ vertices has at least one root-to-leaf path of length at least n, we have that T admits a root-to-leaf path, say P, of length at least c^*. Now, since the number of realizability functions of $G[Y]$ with respect to h is at most $2^{\lambda_1(h,t)}$, we know by an averaging argument that there are at least $\lambda_2(h, t) \cdot \lambda_3(t) \cdot (\lambda_4(t) + 1)$ bags on P that are equivalent with respect to h-states. Further, at least $\lambda_3(t) \cdot (\lambda_4(t) + 1)$ of these bags induce graphs that have a labeled isomorphism between them. By averaging again, we have that at least $(\lambda_4(t) + 1)$ of these bags have the same clause degree sequence, and finally, we see that at least two of these bags must also share their boundary type. These bags are h-redundant, as desired. This concludes the proof of the lemma. $\square$

We now describe the overall replacement algorithm. Given an (t, η)-protrusion Y of size at least c and at most $2c$, where c is as given by Lemma 11, we begin by computing a nice tree decomposition T of $G[Y]$. Walking bottom-up through this tree decomposition, for every node $v \in T$, we compute the realizability function for the graph (H_v, B_v), by brute force. By Lemma 11, we will inevitably find two bags B_u and B_v that are η^2-redundant. We then return the graph $(G \setminus H_u^*, H_u) \oplus (H_v, B_v)$ as the reduced, equivalent instance. The correctness of this reduction follows from Lemma 10.

We note that along with the realizability function, we can also compute a realizability witness table which stores, for every $\omega \in \mathfrak{S}((G, Z), h)$ such that $\Lambda_h(\omega) = 1$, the subset $S^* \subseteq \mathsf{var}(G)$ and a truth assignment τ^* on $\mathsf{var}(G)$ that are witnesses to $\Lambda_h(\omega) = 1$. Note that for a constant-sized protrusion, the size of the realizability witness table is also a constant. We may store, for every protrusion that is replaced, its realizability witness table with only a constant space overhead per replacement. When backtracking through the replacements, the table will enable us to recover assignments made to variables that were deleted as a result of the replacement. Further, if we are given a satisfying assignment τ of the instance obtained after the replacement, say H, then the we can obtain a satisfying assignment for the original instance G as follows. We look up the entry in the realizability witness table corresponding to the tuple $(\mathsf{var}(Z), \emptyset, \tau\restriction_Z, |\mathsf{var}(Z)|, \mathfrak{H})$, where $\mathfrak{H}$ is the collection of all labelled minors of size at most h that can be obtained from $\mathsf{inc}(\psi(G)[\tau\restriction_Z])$. It can be checked that if τ was a satisfying assignment for $\psi(H)$, then the table entry with this index will contain a satisfying assignment for $\psi(G)$. We summarize these observations in our next lemma, where we use $c(d, \eta, t)$ to denote the constant c given by Lemma 11.

Lemma 12. *Let G incidence graph of order d, let Y be a (t, η)-protrusion. If Y has at least $c(d, \eta, t)$ vertices, it is possible to compute a strictly smaller instance H such that G has a weak $\mathcal{K}_\eta$-backdoor of size at most k if, and only if, H has a weak $\mathcal{K}_\eta$-backdoor of size at most k. Further, given a weak $\mathcal{K}_\eta$-backdoor for $\psi(H)$, it is possible to compute a weak $\mathcal{K}_\eta$-backdoor for $\psi(G)$ in constant time. In the case when $\psi(G)$ is satisfiable, given a satisfying assignment for H, it is possible to compute a satisfying assignment for G in constant time.*

We observe that Lemma 12 holds even if we would like to have equivalence with respect to having a $\mathcal{W}_\eta$-backdoor. By the results of Robertson and Seymour, it is well-known that the class of graphs in $\mathcal{W}_\eta$ is characterized by a finite set of forbidden minors. In other words, there is a finite set of graphs $\mathcal{O}_\eta$ such that a graph belongs to $\mathcal{W}_t$ if, and only if, it does not contain any graph from $\mathcal{O}_\eta$ as a minor. Therefore, instead of using $\boxplus_t$ as the forbidden subgraph in the arguments before, we use the set of graphs in $\mathcal{O}_\eta$. The graphs in $\mathcal{O}_\eta$ can be computed in constant time for all values of η by the results of [1]. Therefore, we also have the following:

Lemma 13. *Let G incidence graph of order d, let Y be a (t, η)-protrusion. If Y has at least $c(d, \eta, t)$ vertices, it is possible to compute a strictly smaller instance H such that G has a weak $\mathcal{W}_\eta$-backdoor of size at most k if, and only if, H has a weak $\mathcal{W}_\eta$-backdoor of size at most k. Further, given a weak $\mathcal{W}_\eta$-backdoor for $\psi(H)$, it is possible to compute a weak $\mathcal{W}_\eta$-backdoor for $\psi(G)$ in constant time. In the case when $\psi(G)$ is satisfiable, given a satisfying assignment for H, it is possible to compute a satisfying assignment for G in constant time.*

Our preference for $\boxplus_\eta$ instead of $\mathcal{O}_\eta$ is due to the fact that it is easier to implement replacement without having to compute $\mathcal{O}_\eta$, and we achieve very similar results — the only difference is that when we solve SAT for bounded treewidth formulas, the treewidth is guaranteed to be at most η^{100} instead of being at most η.

3.3.2 Strong $\mathcal{K}_\eta$-Backdoors

For replacing protrusions while maintaining the equivalence with respect to the existence of strong $\mathcal{K}_\eta$-Backdoors, we define a slightly different notion of h-states. For ease of discussion, we continue to use the terms *h-state* and *realizability functions with respect to h*. We note that these terms take on different meanings depending on the problem. In particular, for the purpose of this discussion, a *h-state* of a t-boundaried incidence graph G with boundary Z is a tuple $(P, w, \mathfrak{H})$, where:

- P is a subset of $\mathsf{var}(Z)$,
- $w \in [t]$ is a positive integer, and
- $\{\mathfrak{H}_\tau \mid , \tau : S^* \to \{0,1\}\}$, where each $\mathfrak{H}_\tau$ is a collection of labeled graphs on at most h vertices, where the labels are from $[t]$,
- $\{b_\tau \mid \tau \in 2^P\}$, where each $b_\tau \in \{0,1\}$.

For a t-boundaried incidence graph G with boundary Z, let $\mathfrak{S}((G,Z),h)$ denote the set of all possible h-states of (G,Z). The *realizability function* of G with respect to h, denoted by Λ_h, maps pairs $((G,Z),\omega)$, to $\{0,1\}$, where $\omega \in \mathfrak{S}(G,h)$. This function is defined as follows. Let $\omega = (P, w, \{\mathfrak{H}_\tau \mid , \tau : S^* \to \{0,1\}\})$. We have that $\Lambda_h(\omega) = 1$ if, and only if, there is a subset $S^* \subseteq \mathsf{var}(G)$ of size at most w such that the following holds:

- $S^* \cap Z = P$,
- for every assignment $\tau : S^* \to \{0,1\}$, the collection of labeled graphs that can be obtained as labeled minors of $\mathsf{inc}(\psi(G)[\tau])$ is given by $\mathfrak{H}_\tau$.
- for every assignment $\tau : S^* \to \{0,1\}$, the formula $\psi(G)[\tau]$ is satisfiable if, and only if, $b_\tau = 1$.

Given the notion h-states and realizability functions as above, we have a natural notion of equivalence with respect to h-states, as before, where two instances are equivalent if they have the same realizability functions. We now note that analogs of Lemmas 9,10, and 11, can be shown in the context of strong $\mathcal{K}_\eta$-backdoors. The proofs are along very similar lines. Let $c'(d,\eta,t)$ denote an appropriate function as obtained by an analog of 11.

Additionally, we may define a realizability witness table which stores, for every $\omega \in \mathfrak{S}((G,Z),h)$ such that $\Lambda_h(\omega) = 1$, a witness strong $\mathcal{K}_\eta$-backdoor $S^* \subseteq \mathsf{var}(G)$ and for every truth assignment τ to $S^* \cap Z$ for which $b_\tau = 1$, a truth assignment $\tau^\dagger$ on $\mathsf{var}(G)$ such that $\tau^\dagger$ satisfies $\psi(G)$. These constitute witnesses to $\Lambda_h(\omega) = 1$. We will then have the following:

Lemma 14. *Let G incidence graph of order d, let Y be a (t,η)-protrusion. If Y has at least $c'(d,\eta,t)$ vertices, it is possible to compute a strictly smaller instance H such that G has a strong $\mathcal{K}_\eta$-backdoor of size at most k if, and only if, H has a strong $\mathcal{K}_\eta$-backdoor of size at most k. Further, given a strong $\mathcal{K}_\eta$-backdoor for $\psi(H)$ it is possible to compute a strong $\mathcal{K}_\eta$-backdoor for $\psi(G)$ in constant time. In the case when $\psi(G)$ is satisfiable, given a satisfying assignment for H, it is possible to compute a satisfying assignment for G in constant time.*

We remark that the natural analogue of the lemma above for strong $\mathcal{W}_\eta$-backdoors also holds, and the proof uses the obstruction set $\mathcal{O}_\eta$ as described before in the context of weak backdoors. We conclude this subsection by pointing out that Lemma 4 used in the description of reducers in the previous subsection follows by putting together Lemma 12, Lemma 13 and Lemma 14. Furthemore, the algorithms $\mathcal{A}_1'$ and $\mathcal{A}_2'$ needed to recover satisfying assignments in the proof of Lemma 8 follow from Lemma 12 and Lemma 13.

4 Fast Protrusion Replacement

In this section, we present our linear time algorithm to detect protrusions that cover a sufficiently part of a given graph. Using this algorithm, we prove our Linear Time Protrusion Replacement Theorem. Although the main motivation behind designing this algorithm is to achieve reducers that run in linear time for d-SAT, this theorem is developed in a much more general setting so as to facilitate “black-box” applications and can also be invoked directly to improve several existing kernelization as well as FPT results. We begin by recalling the notions of protrusion covers.

Definition 12. *A (a, b, r, η)-**protrusion cover** in a graph G is a collection $\mathcal{Z} = Z_1, \ldots, Z_q$ of sets such that*

- *for every i, $N[Z_i]$ is a (r, η)-protrusion in G*
- *for every i, $a \leq |Z_i| \leq b$*
- *for every $i \neq j$, $Z_i \cap Z_j = \emptyset$ and $N[Z_i] \cap Z_j = \emptyset$.*

The size of $\mathcal{Z}$ is denoted by $|\mathcal{Z}|$.

Note that the protrusions in a (a, b, r, η)-protrusion cover are not necessarily connected. However, the following lemma shows that we may make this assumption at a cost of decreasing the lower bound on the sizes of the protrusions.

Lemma 15. *Let G be a graph with a $(s, 6s, r, \eta)$-protrusion cover $\mathcal{Z}$. Then, G has a $(\frac{s}{2^r}, 6s, r, \eta)$-protrusion cover $\mathcal{Z}'$ of size at least $|\mathcal{Z}|$ such that for every $Z \in \mathcal{Z}'$, the connected components of $G[Z]$ have the same neighbourhood.*

Proof. Let $\mathcal{Z}_1, \ldots, \mathcal{Z}_p$ be the partition of the sets in $\mathcal{Z}$ according to their neighborhood. Furthermore, for each $\mathcal{Z}_i$, let $\mathcal{P}_i$ denote a subset of $\mathcal{Z}_i$ such that for every $Z \in \mathcal{Z}_i$, there is a set $P \in \mathcal{P}_i$ such that $|P| \geq \frac{s}{2^r}$ and the connected components of $G[P]$ have the same neighborhood. Since each $Z \in \mathcal{Z}_i$ has size at least s and neighbourhood at most r, such a P exists for every Z and therefore $\mathcal{P}_i$ exists for every $\mathcal{Z}_i$. Observe that the set $\mathcal{P} = \{\mathcal{P}_i | i \in [p]\}$ is indeed a $(\frac{s}{2^r}, 6s, r, \eta)$-protrusion cover satisfying the conditions in the statement of the lemma. □

Clearly, it is very desirable to be able to compute protrusion covers of large size, which then allows us to reduce the size of instances by a significant amount. Our next algorithm achieves this – in linear time, it computes a protrusion cover which “approximates” any protrusion cover in the graph with certain parameters. We use the following algorithm for enumerating small connected components with a small neighborhood.

Proposition 1. [10] *Let G be a graph and let $v \in V(G)$, $p, q \in [|V(G)|]$. The number of sets S containing v such that $G[S]$ is connected, $|S| \leq p$, and $|N(S)| \leq q$ is at most $\binom{p+q}{p}$ and given v, they can be enumerated in constant time for fixed p and q.*

Lemma 16. *For every r and s where $s > 2^r$, there is an algorithm that runs in time $\mathcal{O}(m+n)$ and computes a $(\frac{s}{2^r}, 7s, r, \eta)$-protrusion cover. Furthermore, if G has a $(\frac{s}{2^r}, 6s, r, \eta)$-protrusion cover $\mathcal{Z}$ such that for every $Z \in \mathcal{Z}'$, the connected components of $G[Z]$ have the same neighbourhood then the computed $(\frac{s}{2^r}, 7s, r, \eta)$-protrusion cover $\mathcal{Z}'$ has size at least $\delta|\mathcal{Z}|$, where δ is a constant depending only on r and s.*

Proof. For every vertex $v \in V(G)$, we use Proposition 1 to enumerate the family $\mathcal{S}_v = \{S \subseteq V(G) | v \in S, |S| \leq 6s, |N(S)| \leq r, G[S]$ is connected $\}$. Since enumerating the family $\mathcal{S}_v$ for each vertex takes constant time, the sets $\mathcal{S}_v$ for all vertices can be computed in time $\mathcal{O}(n)$. For every $v \in V(G)$, we discard the sets $S \in \mathcal{S}_v$ such that $tw(G[S]) > t$. Since we can use the algorithm of Bodlaender[3] to compute the treewidth of each $S \in \mathcal{S}_v$ in constant time, the discarding process taken over the sets S_v for all $v \in V(G)$ can be done linear time. Let $S^* = \bigcup_{v \in V(G)} \mathcal{S}_v$. We now group the sets in S^* according to their neighbourhood. More precisely, we compute a partition of S^* where sets with the same neighbourhood are in the same class of the partition. This can be done as follows (see for example [7]). Fix an ordering of the vertex set of G and sort the neighbor lists of each set in S^*. Following this, in r stable bucket sorts, we can sort the sets in S^* based on their 'first' neighbor first, then the 'second' and so on. This procedure takes time $\mathcal{O}(nr)$ since $|S^*| = \mathcal{O}(n)$ and each set in S^* has a boundary of size at most r. Let $\mathcal{X}_1, \ldots, \mathcal{X}_q$ be the partition of S^* obtained as described above. Observe that for any $i \in [q]$, for any $S_1, S_2 \in \mathcal{X}_i$, $S_1 \cap S_2 = \emptyset$. This follows from the fact that $G[S_1]$ and $G[S_2]$ are both connected. More precisely, if $S_1 \cap S_2 \neq \emptyset$ then S_1 must have a neighbor in S_2, contradicting the fact that they lie in the same class of the partition.

We now club together certain sets in each class of the partition together as follows. For each $\mathcal{X}_i$ do the following. As long as $\sum_{S \in \mathcal{X}_i} |S| \geq \frac{s}{2^r}$, select a minimal collection $\mathcal{X}'_i \subseteq \mathcal{X}_i$ such that $\sum_{S \in \mathcal{X}'_i} |S| \geq \frac{s}{2^r}$ and add it to the set $\mathcal{Y}$ and remove the sets in this collection from $\mathcal{X}_i$. We repeat this step as long as possible. Observe that each time we add a minimal collection $\mathcal{X}'_i \subseteq \mathcal{X}_i$ to $\mathcal{Y}$, it must be the case that $\frac{s}{2^r} \leq \sum_{S \in \mathcal{X}'_i} |S| \leq 6s + \frac{s}{2^r} \leq 7a$. Let $\mathcal{Y}_1, \ldots, \mathcal{Y}_q$ be the collections added to $\mathcal{Y}$ in this way where we know that for every $i \in [q]$, the sum of the sizes of the sets in $\mathcal{Y}_i$ is at least $\frac{s}{2^r}$ and at most $7s$.

Before describing the subsequent steps of the algorithm, we prove a bound on the size of the set $\mathcal{Y}$ assuming the existence of a $(\frac{s}{2^r}, 6s, r, \eta)$-protrusion cover $\mathcal{Z}$. Let $\mathcal{Z}_1, \ldots, \mathcal{Z}_p$ be the partition of the sets in $\mathcal{Z}$ according to their neighbourhood. Observe that for each $i \in [p]$, $\mathcal{Z}_i$ is a subset of $\mathcal{X}_j$ for some unique j, denoted by $\sigma(i)$. We will show that for $\mathcal{Z}_i$, the number of collections contributed by $\mathcal{X}_{\sigma(i)}$ to the set $\mathcal{Y}$ is a constant fraction of the number of sets in $\mathcal{Z}_i$.

More precisely, let V_i be the set of vertices in the union of the sets in $\mathcal{Z}_i$. Since every set in $\mathcal{Z}_i$ has size at least $\frac{s}{2^r}$, the size of each set $\mathcal{Z}_i$ is at most $\frac{|V_i| 2^r}{s}$. Furthermore, since every collection contributed by $\mathcal{X}_{\sigma(i)}$ to $\mathcal{Y}$ covers at most $7s$ vertices, the number of collections contributed by $\mathcal{X}_{\sigma(i)}$ to $\mathcal{Y}$ is at least $\lfloor \frac{|V_i|}{7s} \rfloor$. If $|V_i| \leq 8s$, then since $\mathcal{X}_{\sigma(i)}$ contributes at least one collection to $\mathcal{Y}$, we conclude that the number of collections contributed by $\mathcal{X}_{\sigma(i)}$ to $\mathcal{Y}$ is at least a $\frac{1}{8 \cdot 2^r} = \frac{1}{2^{r+3}}$ fraction of the size of $\mathcal{Z}_i$. On the other hand, if $|V_i| > 8s$, then since $\mathcal{X}_{\sigma(i)}$ contributes at least $\frac{|V_i|}{7s} - 1$ collections to $\mathcal{Y}$, we infer that the number of collections contributed by $\mathcal{X}_{\sigma(i)}$ to $\mathcal{Y}$ is at least a $\frac{1}{2^{r+3}}$ fraction of the size of $\mathcal{Z}_i$. Having concluded that for any $(\frac{s}{2^r}, 6s, r, \eta)$-protrusion cover $\mathcal{Z}$, we have that $|\mathcal{Y}| \geq \omega \cdot |\mathcal{Z}|$ where $\omega = \frac{1}{2^{r+3}}$, we now move to the final step of obtaining the required protrusion cover $\mathcal{Z}'$.

Let H be a graph with vertex set corresponding to the sets in $\mathcal{Y}$ and edge set defined as follows. There is an edge between vertices u and v in H if the corresponding sets $\mathcal{Y}_u$ and $\mathcal{Y}_v$ are such that for some $Y_u \in \mathcal{Y}_u$ and $Y_v \in \mathcal{Y}_v$, $Y_u \cap Y_v \neq \emptyset$ or Y_u has an edge to Y_v. We observe that the maximum degree $\Delta(H)$ of the graph H is bounded by a constant depending only on s and r. Indeed, for each $\mathcal{Y}_u$, the number of sets in $\mathcal{S}^*$ that intersect a set $Y \in \mathcal{Y}_u$ is bounded by $|Y| \cdot \binom{6s+r}{r}$. Furthermore, the number of sets in $\mathcal{S}^*$ that intersect $N(Y)$ is bounded by $r \cdot \binom{6s+r}{r}$. Finally, the number of vertices contained in the union of the sets in $\mathcal{Y}_u$ for any $u \in V(H)$ is bounded by $7s$ and each $\mathcal{Y}_u$ is a union of sets in $\mathcal{S}^*$. Therefore, the degree of any vertex in H is bounded by $\lambda = (7s + r) \cdot \binom{6s+r}{r}$. Therefore, we can compute in time $\mathcal{O}(V(H) + E(H))$ an independent set in H of size at least $\frac{|V(H)|}{\lambda+1}$. We set $\mathcal{Z}'$ to be the collection of sets corresponding to the vertices in this independent set. It is clear that $\mathcal{Z}'$ indeed is a $(\frac{s}{2^r}, 7s, r, \eta)$-protrusion cover of G. Therefore, it only remains to prove the required lower bound on the size of $\mathcal{Z}'$. Set $\delta = \frac{\omega}{(\lambda+1)}$. Observe that the size of $\mathcal{Z}'$ is at least a $\frac{1}{\lambda+1}$ fraction of $|V(H)|$, which in turn is at least $\omega|\mathcal{Z}|$. This completes the proof of the lemma. □

In particular, the above lemma implies that if G has a protrusion-cover $\mathcal{Z}$ such that a constant fraction of vertices of G appear in distinct reducible sets in $\mathcal{Z}$, then we can reduce G by a constant fraction of its vertices in linear time by computing a large enough approximate protrusion cover and then invoking the appropriate protrusion replacer, leading to a linear time algorithm for protrusion replacement. We now state and prove our theorem formally. Before we state the theorem, we recall the following lemma from [7] relating protrusion decompositions and protrusion covers with certain size guarantees.

Lemma 17. [7] *Let G be a graph with a (α, β, η)-protrusion decomposition. Then, for all $s > \beta$, G has a $(s, 6s, 3(\beta+1), \eta)$-protrusion cover of size at least $\frac{n}{122s} - \alpha$.*

Theorem 3. *(Linear Time Protrusion Replacement Theorem) Let Π be a problem that has a protrusion replacer which replaces r protrusions of size at least r' for some fixed r. Let s and β be constants such that $s \geq r' \cdot 2^r$ and $r \geq 3(\beta+1)$. Given an instance (G, k) as input, there is an algorithm that runs in time $\mathcal{O}(m+n)$ and produces an equivalent instance (G', k') with $|V(G')| \leq |V(G)|$ and $k' \leq k$. If additionally G has a (α, β)-protrusion decomposition such that $\alpha \leq \frac{n}{244s}$, then we have that $|V(G')| \leq (1-\delta)|V(G)|$ for some constant δ.*

Proof. We first run the algorithm of Lemma 16 with the parameters r and s to compute a $(\frac{s}{2^r}, 7s, r, r)$-protrusion cover $\mathcal{Z}$. Since $\frac{s}{2^r} > r'$, each set in $\mathcal{Z}$ is a reducible protrusion and therefore we invoke the protrusion replacer to reduce all protrusions in $\mathcal{Z}$ to get an equivalent instance (G', k'), where $|V(G')| \leq |V(G)| - |\mathcal{Z}|$. We now claim that if G has a (α, β)-protrusion decomposition such that $\alpha \leq \frac{n}{244s}$, then we have that $|\mathcal{Z}| \geq \delta V(G)$ for some constant δ, implying that $|V(G')| \leq (1-\delta)|V(G)|$.

By Lemma 17 we know that if G has a (α, β)-protrusion decomposition then for all $s > \beta$, G has a $(s, 6s, 3(\beta+1), r)$-protrusion cover of size at least $\frac{n}{122s} - \alpha \leq \frac{n}{244s}$. Furthermore, by Lemma 15, we know that G has a $(\frac{s}{2^r}, 6s, r, r)$-protrusion cover $\mathcal{Z}'$ of size at least $\frac{n}{244s}$ such that for every $Z \in \mathcal{Z}'$, the connected components of $G[Z]$ have the same neighbourhood. However, in this case Lemma 16 guarantees that the computed protrusion cover $\mathcal{Z}$ has size at least $\delta'|\mathcal{Z}'|$ for some constant δ'. Therefore, setting $\delta = \frac{\delta'}{244s}$ completes the proof of the theorem. □

The proof of Lemma 5 (required for the description of reducers) is identical to the proof of the above theorem, except that when replacing protrusions in the incidence graph of a formula, we need to use the satisfiability preserving protrusion replacers described in the previous section.

5 Kernel lower bounds

In this section, we show that the problem of finding a weak $\mathcal{W}_\eta$-backdoor of size at most k, when parameterized by k, does not admit a compression algorithm unless NP $\subseteq$ coNP/ poly, even when we restrict ourselves to 4-CNF formulas. We refer the reader to the preliminaries for the terminology that is used in the proof.

Lemma 18. *Unless $NP \subseteq coNP/Poly$, there is no polynomial compression for the problem of determining if a 4-CNF formula admits a weak $\mathcal{W}_\eta$-backdoor of size at most k, when parameterized by k.*

Proof. The proof is by cross-composition from 3-CNF-SAT. We define our polynomial equivalence relations as follows: two CNF formulas are equivalent if they have the same number of variables and the same number of clauses. Given 3-CNF formulas $\phi_1, \phi_2, \ldots, \phi_t$ over the variable set $\{x_1, \ldots, x_n\}$, we construct the following formula. Let ψ_i denote the formula obtained by inserting a new variable α_i into all the clauses of ϕ_i, that is, if $C_1^i, \ldots, C_m^i$ are the clauses of ϕ_i, then we have:

$$\psi_i := \bigwedge_{j=1}^{m} (C_j^i \cup \{\alpha_i\})$$

Let ψ_i^* denote the 3-CNF equivalent of ψ_i. Further, for new variables $\alpha_1, \ldots, \alpha_t$, let W be the clause $(\bar{\alpha}_1 \vee \bar{\alpha}_2 \vee \ldots, \vee \bar{\alpha}_t)$ and let ψ_0 be the 3-CNF equivalent of W, that is,

$$\psi_0 := (\bar{\alpha}_1 \vee \bar{\alpha}_2 \vee y_1) \wedge (\bar{y}_1 \vee \bar{\alpha}_3 \vee y_2) \wedge \cdots \wedge (\bar{\alpha}_t \vee \bar{y}_{t-2})$$

For each x_i, $1 \leq i \leq n$, insert the following clauses:

$$\lambda_i := (x_i \vee z_i^1) \wedge (\bar{x}_i \vee z_i^1) \wedge (x_i \vee z_i^2) \wedge (\bar{x}_i \vee z_i^2) \wedge \cdots \wedge (x_i \vee z_i^{n+1}) \wedge (\bar{x}_i \vee z_i^{n+1})$$

The composed formula is given by

$$\psi := \left\{\wedge_{j=0}^{r} \psi_j^*\right\} \bigwedge \left\{\wedge_{j=1}^{n} \lambda_j\right\}$$

We now claim that ψ has a weak backdoor set of size at most n if, and only if, there is an $1 \leq i \leq t$ such that ϕ_i is satisfiable.

Forward Direction. Let i be such that ϕ_i is satisfiable. Let τ be a satisfying assignment for ϕ_i. We claim that $\{x_1, \ldots, x_n\}$ is an acyclic weak backdoor set with witness assignment τ.

Consider $\psi[\tau]$. Set $\alpha_i = 0$ and $\alpha_j = 1$ for every $j \neq i$. This satisfies W and all unit clauses involving an α_j for $j \neq i$. Finally, setting $z_i = 1$ for every i gives the satisfying assignment for ϕ. It is easy to see that $\psi[\tau]$ is indeed a forest. In particular, the auxiliary y-variables degenerate into paths where the clause vertices are adjacent to vertices corresponding to α_i's. The α-vertices, in turn, are potentially adjacent to several leaf vertices (coming from the formula the clauses of the ϕ_i that were not satisfied by the assignment τ on the x_i's). It can be verified that these components together constitute an acyclic graph.

Reverse Direction. Observe that any acyclic weak backdoor set must contain all the x_i's, because of the λ_i's. Consider any witness assignment τ for ψ. Note that any satisfying assignment for ψ cannot set all the α_i's to true. In particular, therefore, τ sets at least one of the α_i's to false. Let α_j be such that $\tau(\alpha_j) = 0$. Now consider $\phi' := \psi[\tau\restriction_{\alpha_1,\ldots,\alpha_t}]$, and note that all the clauses from ϕ_j are present in ϕ'. Therefore, the assignment of τ restricted to the x_i's must be a satisfying assignment for ϕ_j, which concludes the argument. □

References

[1] I. Adler, M. Grohe, and S. Kreutzer, *Computing excluded minors*, in Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2008), ACM-SIAM, 2008, pp. 641–650.

[2] H. Bodlaender, F. V. Fomin, D. Lokshtanov, E. Penninkx, S. Saurabh, and D. M. Thilikos, *(Meta) Kernelization*, in Proceedings of the 50th Annual Symposium on Foundations of Computer Science (FOCS), IEEE, 2009, pp. 629–638.

[3] H. L. Bodlaender, *A linear-time algorithm for finding tree-decompositions of small treewidth*, SIAM J. Comput., 25 (1996), pp. 1305–1317.

[4] C. Chekuri and J. Chuzhoy, *Polynomial bounds for the grid-minor theorem*, in STOC, 2014, pp. 60–69.

[5] E. Dantsin and E. A. Hirsch, *Worst-case upper bounds*, in Handbook of Satisfiability, A. Biere, M. Heule, H. van Maaren, and T. Walsh, eds., vol. 185 of Frontiers in Artificial Intelligence and Applications, IOS Press, 2009, pp. 403–424.

[6] R. Dechter and J. Pearl, *Tree clustering for constraint networks.*, Artif. Intell., (1989), pp. 353–366.

[7] F. V. Fomin, D. Lokshtanov, N. Misra, and S. Saurabh, *Planar F-deletion: Approximation, kernelization and optimal FPT algorithms*, in Proceedings of the 53rd Annual Symposium on Foundations of Computer Science (FOCS), IEEE, 2012, pp. 470–479.

[8] F. V. Fomin, D. Lokshtanov, and S. Saurabh, *Bidimensionality and geometric graphs*, in Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2012, pp. 1563–1575.

[9] F. V. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos, *Bidimensionality and kernels*, in Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2010, pp. 503–510.

[10] F. V. Fomin and Y. Villanger, *Treewidth computation and extremal combinatorics*, Combinatorica, 32 (2012), pp. 289–308.

[11] E. C. Freuder, *A sufficient condition for backtrack-bounded search*, J. ACM, 32 (1985), pp. 755–761.

[12] J. Gajarský, P. Hlinený, J. Obdrzálek, S. Ordyniak, F. Reidl, P. Rossmanith, F. S. Villaamil, and S. Sikdar, *Kernelization using structural parameters on sparse graph classes*, in Proceedings of the 21st Annual European Symposium on Algorithms (ESA), vol. 8125 of Lecture Notes in Computer Science, Springer, 2013, pp. 529–540.

[13] V. Garnero, C. Paul, I. Sau, and D. M. Thilikos, *Explicit Linear Kernels via Dynamic Programming*, in STACS, 2014, pp. 312–324.

[14] S. Gaspers and S. Szeider, *Backdoors to acyclic SAT*, in Proceedings of the 39th International Colloquium on Automata, Languages, and Programming (ICALP), vol. 7391 of Lecture Notes in Computer Science, Springer, 2012, pp. 363–374.

[15] ——, *Backdoors to satisfaction*, in The Multivariate Algorithmic Revolution and Beyond, 2012, pp. 287–317.

[16] ——, *Strong backdoors to bounded treewidth SAT*, in 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society, 2013, pp. 489–498.

[17] R. Impagliazzo and R. Paturi, *The complexity of k-sat*, in COCO '99: Proceedings of the Fourteenth Annual IEEE Conference on Computational Complexity, Washington, DC, USA, 1999, IEEE Computer Society, p. 237.

[18] S. Jukna, *Extremal combinatorics With applications in computer science*, Springer-Verlag, Berlin, 2001.

[19] E. J. Kim, A. Langer, C. Paul, F. Reidl, P. Rossmanith, I. Sau, and S. Sikdar, *Linear kernels and single-exponential algorithms via protrusion decompositions*, in Proceedings of the 40th International Colloquium of Automata, Languages and Programming (ICALP), vol. 7965 of Lecture Notes in Computer Science, Springer, 2013, pp. 613–624.

[20] Z. Li and P. van Beek, *Finding small backdoors in SAT instances*, in Proceedings of the 24th Canadian Conference Advances in Artificial Intelligence (Canadian AI), vol. 6657 of Lecture Notes in Computer Science, Springer, 2011, pp. 269–280.

[21] S. Malik and L. Zhang, *Boolean satisfiability from theoretical hardness to practical success*, Commun. ACM, 52 (2009), pp. 76–82.

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

[23] M. Samer and S. Szeider, *Backdoor sets of quantified boolean formulas*, in Proceedings of SAT 2007, Tenth International Conference on Theory and Applications of Satisfiability Testing, May 28-31, 2007, Lisbon, Portugal,, J. Marques-Silva and K. A. Sakallah, eds., vol. 4501 of Lecture Notes in Computer Science, 2007, pp. 230Âŋ–243.

[24] ——, *Constraint satisfaction with bounded treewidth revisited*, J. Comput. Syst. Sci., 76 (2010), pp. 103–114.

[25] S. Szeider, *Matched formulas and backdoor sets*, in Proceedings of SAT 2007, Tenth International Conference on Theory and Applications of Satisfiability Testing, May 28-31, 2007, Lisbon, Portugal,, J. Marques-Silva and K. A. Sakallah, eds., vol. 4501 of Lecture Notes in Computer Science, 2007, pp. 94–99.

[26] R. Williams, C. Gomes, and B. Selman, *Backdoors to typical case complexity*, in Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, IJCAI 2003, G. Gottlob and T. Walsh, eds., Morgan Kaufmann, 2003, pp. 1173–1178.