Local Backbones

Ronald de Haan[1,⋆], Iyad Kanj[2], and Stefan Szeider[1,⋆]

[1] Institute of Information Systems, Vienna University of Technology, Vienna, Austria
[2] School of Computing, DePaul University, Chicago, IL

Abstract. A backbone of a propositional CNF formula is a variable whose truth value is the same in every truth assignment that satisfies the formula. The notion of backbones for CNF formulas has been studied in various contexts. In this paper, we introduce local variants of backbones, and study the computational complexity of detecting them. In particular, we consider k-backbones, which are backbones for sub-formulas consisting of at most k clauses, and iterative k-backbones, which are backbones that result after repeated instantiations of k-backbones. We determine the parameterized complexity of deciding whether a variable is a k-backbone or an iterative k-backbone for various restricted formula classes, including Horn, definite Horn, and Krom. We also present some first empirical results regarding backbones for CNF-Satisfiability (SAT). The empirical results we obtain show that a large fraction of the backbones of structured SAT instances are local, in contrast to random instances, which appear to have few local backbones.

1 Introduction

A *backbone* of a propositional formula φ is a variable whose truth value is the same for all satisfying assignments of φ. The term originates in computational physics [24], and the notion of backbones has been studied for SAT in various contexts. Backbones have also been considered in other contexts (e.g., knowledge compilation [5]) and for other combinatorial problems [25]. If a backbone and its truth value are known, then we can simplify the formula without changing its satisfiability, or the number of satisfying assignments. Therefore, it is desirable to have an efficient algorithm for detecting backbones. In general, however, the problem of identifying backbones is coNP-complete (this follows from the fact that a literal l is enforced by a formula φ if and only if $\varphi \wedge \neg l$ is unsatisfiable).

A variable can be a backbone because of *local properties* of the formula (such backbones we call *local backbones*). As an extreme example consider a CNF formula that contains a unit clause. In this case we know that the variable appearing in the unit clause is a backbone of the formula. More generally, we define the *order* of a backbone x of a CNF formula φ to be the cardinality of a smallest subset $\varphi' \subseteq \varphi$ such that x is a backbone of φ', and we refer to backbones of order $\leq k$ as *k-backbones*. Thus, unit clauses give rise to 1-backbones.

A natural generalization of k-backbones are variables whose truth values are enforced by repeatedly assigning k-backbones to their appropriate truth value and simplifying the formula according to this assignment. We call variables that are assigned

⋆ Supported by the European Research Council (ERC), project COMPLEX REASON, 239962.

M. Järvisalo and A. Van Gelder (Eds.): SAT 2013, LNCS 7962, pp. 377–393, 2013.
© Springer-Verlag Berlin Heidelberg 2013

by this iterative process *iterative k-backbones* (for a formal definition, see Section 2.1). For instance, iterative 1-backbones are exactly those variables whose truth values are enforced by unit propagation. The *iterative order* of a backbone x is the smallest k such that x is an iterative k-backbone.

Finding Local Backbones. For every constant k, we can clearly identify all k-backbones and iterative k-backbones of a CNF formula φ in polynomial time by simply going over all subsets of φ of size at most k (and iterating this process if necessary). However, if φ consists of m clauses, then this brute-force search requires us to consider at least m^k subsets, which is impractical already for small values of k. It would be desirable to have an algorithm that detects (iterative) k-backbones in time $f(k)||\varphi||^c$ where f is a function, $||\varphi||$ denotes the length of the formula, and c is a constant. An algorithm with such a running time would render the problem *fixed-parameter tractable* with respect to parameter k [7]. In this paper we study the question of whether the identification of (iterative) k-backbones of a CNF formula is fixed-parameter tractable or not, considering various restrictions on the CNF formula. We therefore define the following template for parameterized problems, where $\mathcal{C}$ is an arbitrary class of CNF formulas.

LOCAL-BACKBONE$[\mathcal{C}]$
Instance: a CNF formula $\varphi \in \mathcal{C}$, a variable x of φ, and an integer $k \geq 1$.
Parameter: The integer k.
Question: Is x a k-backbone of φ?

The problem ITERATIVE-LOCAL-BACKBONE is defined similarly. It is not hard to see that LOCAL-BACKBONE$[\mathcal{C}]$ is closely related to the problem of finding a small unsatisfiable subset of a CNF formula (this is proven below in Lemmas 1 and 2). More precisely, for every class $\mathcal{C}$, the problem LOCAL-BACKBONE$[\mathcal{C}]$ has the same parameterized complexity as the following problem, studied by Fellows et al. [10].

SMALL-UNSATISFIABLE-SUBSET$[\mathcal{C}]$
Instance: a CNF formula $\varphi \in \mathcal{C}$, and an integer $k \geq 1$.
Parameter: The integer k.
Question: Is there an unsatisfiable subset $\varphi' \subseteq \varphi$ consisting of at most k clauses?

This problem is of relevance also for classes $\mathcal{C}$ for which the satisfiability is decidable in polynomial time. For instance, given an inconsistent knowledge base in terms of an unsatisfiable set of Horn clauses, one might want to detect the cause for the inconsistency in terms of a small unsatisfiable subset.

Results. We draw a detailed parameterized complexity map of the considered problems LOCAL-BACKBONE$[\mathcal{C}]$, ITERATIVE-LOCAL-BACKBONE$[\mathcal{C}]$, and SMALL-UNSATISFIABLE-SUBSET$[\mathcal{C}]$, for various classes $\mathcal{C}$. Table 1 provides an overview of our complexity results (FPT indicates that the problem is fixed-parameter tractable, W[1]-hardness indicates strong evidence that the problem is not fixed-parameter tractable; see Section 2.2 for details). It is interesting to observe that the non-iterative problems tend to be at least as hard as the iterative problems. Somewhat surprising is the W[1]-hardness of LOCAL-BACKBONE[KROM] and SMALL-UNSATISFIABLE-SUBSET[KROM] (which also implies the NP-hardness of the unparameterized versions of these problems). On the one hand, this seems to contrast with the fact that a shortest tree-like resolution

Table 1. Map of parameterized complexity results. (The classes $\mathcal{C}$ of formulas are defined in Section 2.1).

$\mathcal{C}$	LOCAL-BACKBONE[$\mathcal{C}$]		ITERATIVE-LOCAL-BACKBONE[$\mathcal{C}$]	
CNF	W[1]-c	(Thm 2)	W[1]-h	(Cor 3)
DEFHORN	W[1]-c	(Thm 2)	P	(Thm 7)
NUHORN	W[1]-c	(Thm 3)	W[1]-h	(Cor 3)
KROM	W[1]-c	(Thm 4)	P	(Thm 8)
VO_d	FPT	(Thm 5)	FPT	(Thm 6)

refutation of an unsatisfiable Krom formula can be found in polynomial time [4]. On the other hand, this is in line with the result that deciding whether a CNF formula can be refuted within k resolution steps (parameterized by k) is W[1]-complete [10]. The polynomial time solvability of finding iterative local backbones in Krom and definite Horn formulas is also interesting, especially in the light of the intractability of the corresponding problems of finding (non-iterative) local backbones.

We also provide some first empirical results on the distribution of local backbones in some benchmark SAT instances. We consider structured instances and random instances. For the structured instances that we consider we observe that a large fraction of the backbones are of relatively small iterative order. In contrast, the backbones of the random instances that we consider are of large iterative order. The results suggest that the distribution of the iterative order of backbones might be an indicator for a hidden structure in SAT instances.

Related Work. The notion of backbones has initially been studied in the context of optimization problems in computational physics [24]. The notion has later been applied to several combinatorial problems [25], including SAT. The relation between backbones and the difficulty of finding a solution for SAT has been studied by Kilby et al. [18], by Parkes [22] and by Slaney and Walsh [25]. The complexity of finding backbones has been studied theoretically by Kilby et al. [18]. The notion of backbones has also been used for improving SAT solving algorithms by Dubois and Dequen [8] and by Hertli et al. [14]. The problem of identifying unsatisfiable subsets of size at most k has been considered by Fellows et al. [10], who proved that this problem is W[1]-complete. Furthermore, they showed by the same reduction that finding a k-step resolution refutation for a given formula is W[1]-complete as well. Related notions of locally enforced literals have also been studied, including a notion of generalized unit-refutation [13,19].

Full Version. Because of space constraints some proofs have been omitted or shortened. Detailed proofs can be found in the full version, available at arxiv.org/abs/1304.5479.

2 Preliminaries

2.1 CNF Formulas, Unsatisfiable Subsets and Local Backbones

A *literal* is a propositional variable x or a negated variable $\neg x$. The *complement* $\overline{x}$ of a positive literal x is $\neg x$, and the complement $\overline{\neg x}$ of a negative literal $\neg x$ is x. A *clause*

is a finite set of literals, not containing a complementary pair $x, \neg x$. A *unit clause* is a clause of size 1. We let $\perp$ denote the empty clause. A *formula* in conjunctive normal form (or CNF formula) is a finite set of clauses. We define the *length* $||\varphi||$ of a formula φ to be $\sum_{c \in \varphi} |c|$; the number of clauses of φ is denoted by $|\varphi|$. A formula φ is a k-CNF formula if the size of each of its clauses is at most k. A 2-CNF formula is also called a Krom formula. A clause is a *Horn clause* if it contains at most one positive literal. A Horn clause containing exactly one positive literal is a *definite Horn clause*. Formulas containing only Horn clauses are called *Horn formulas*. *Definite Horn formulas* are defined analogously. We denote the class of all Krom formulas by KROM, the class of all Horn formulas by HORN and the class of all definite Horn formulas by DEFHORN. We let NUHORN denote the class of Horn formulas not containing unit clauses (such formulas are always satisfiable). Let d be an integer. The class of CNF formulas such that each variable occurs at most d times is denoted by VO_d.

For a CNF-formula φ, the set $\text{Var}(\varphi)$ denotes the set of all variables x such that some clause of φ contains x or $\neg x$; the set $\text{Lit}(\varphi)$ denotes the set of all literals l such that some clause of φ contains l or $\overline{l}$. A formula φ is *satisfiable* if there exists an assignment $\tau : \text{Var}(\varphi) \to \{0,1\}$ such that every clause $c \in \varphi$ contains some variable x with $\tau(x) = 1$ or some negated variable $\neg x$ with $\tau(x) = 0$ (we say that such an assigment τ satisfies φ); otherwise, φ is *unsatisfiable*. φ is *minimally unsatisfiable* if φ is unsatisfiable and every proper subset of φ is satisfiable. It is well-known that any minimal unsatisfiable CNF formula has more clauses than variables (this is known as Tarsi's Lemma [1,20]). For two formulas φ, ψ, whenever all assignments satisfying φ also satisfy ψ, we write $\varphi \models \psi$. The reduct $\varphi|_L$ of a formula φ with respect to a set of literals $L \subseteq \text{Lit}(\varphi)$ is the set of clauses of φ that do not contain any $l \in L$ with all occurrences of $\overline{l}$ for all $l \in L$ removed. For singletons $L = \{l\}$, we also write $\varphi|_l$. We say that a class $\mathcal{C}$ of formulas is *closed under variable instantiation* if for every $\varphi \in \mathcal{C}$ and every $l \in \text{Lit}(\varphi)$ we have that $\varphi|_l \in \mathcal{C}$. For an integer k, a variable x is a *k-backbone* of φ, if there exists a $\varphi' \subseteq \varphi$ such that $|\varphi'| \leq k$ and either $\varphi' \models x$ or $\varphi' \models \neg x$. A variable x is a *backbone* of a formula φ if it is a $|\varphi|$-backbone. Note that the definition of the backbone of a formula φ that is used in some of the literature includes all literals $l \in \text{Lit}(\varphi)$ such that $\varphi \models l$. For an integer k, a variable x is an *iterative k-backbone* of φ if either (i) x is a k-backbone of φ, or (ii) there exists $y \in \text{Var}(\varphi)$ such that y is a k-backbone of φ, and for some $l \in \{y, \neg y\}$, $\varphi \models l$ and x is an iterative k-backbone of $\varphi|_l$.

For a Krom formula φ, we let $\text{impl}(\varphi)$ be the *implication graph* (V, E) of φ, where $V = \{\, x, \neg x : x \in \text{Var}(\varphi) \,\}$ and $E = \{\, (\overline{a}, b), (\overline{b}, a) : \{a, b\} \in \varphi \,\}$. We say that a path p in this graph *uses a clause* $\{a, b\}$ of φ if either one of the edges $(\overline{a}, b)$ and $(\overline{b}, a)$ occurs in p; we say that p *doubly uses* this clause if both edges occur in p.

2.2 Parameterized Complexity

Here we introduce the relevant concepts of parameterized complexity theory. For more details, we refer to text books on the topic [7,11,21]. An instance of a parameterized problem is a pair (I, k) where I is the main part of the instance, and k is the parameter. A parameterized problem is *fixed-parameter tractable* if instances (I, k) can be solved by a deterministic algorithm that runs in time $f(k)|I|^c$, where f is a computable function

of k, and c is a constant (algorithms running within such time bounds are called *fpt-algorithms*). If $c = 1$, we say the problem is *fixed-parameter linear*. FPT denotes the class of all fixed-parameter tractable problems. Using fixed-parameter tractability, many problems that are classified as intractable in the classical setting can be shown to be tractable for small values of the parameter.

Parameterized complexity also offers a *completeness theory*, similar to the theory of NP-completeness. This allows the accumulation of strong theoretical evidence that a parameterized problem is not fixed-parameter tractable. Hardness for parameterized complexity classes is based on fpt-reductions, which are many-one reductions where the parameter of one problem maps into the parameter for the other. More specifically, a parameterized problem L is fpt-reducible to another parameterized problem L' (denoted $L \leq_{\text{fpt}} L'$) if there is a mapping R from instances of L to instances of L' such that (i) $(I, k) \in L$ if and only if $(I', k') = R(I, k) \in L'$, (ii) $k' \leq g(k)$ for a computable function g, and (iii) R can be computed in time $O(f(k)|I|^c)$ for a computable function f and a constant c.

Central to the completeness theory is the hierarchy FPT $\subseteq$ W[1] $\subseteq$ W[2] $\subseteq \cdots \subseteq$ para-NP. Each intractability class W[t] contains all parameterized problems that can be reduced to a certain parameterized satisfiability problem under fpt-reductions. The intractability class para-NP includes all parameterized problems that can be solved by a nondeterministic fpt-algorithm. Fixed-parameter tractability of any problem hard for any of these intractability classes would imply that the Exponential Time Hypothesis fails [11,16] (i.e., the existence of a $2^{o(n)}$ algorithm for n-variable 3SAT).

3 Local Backbones and Small Unsatisfiable Subsets

The straightforward reductions in the proofs of the following two lemmas, illustrate the close connection between LOCAL-BACKBONE and SMALL-UNSATISFIABLE-SUBSET.

Lemma 1. SMALL-UNSATISFIABLE-SUBSET $\leq_{\text{fpt}}$ LOCAL-BACKBONE.

Proof. Let (φ, k) be an instance of SMALL-UNSATISFIABLE-SUBSET. We construct an instance (φ', z, k) of LOCAL-BACKBONE, by letting $\varphi' = \{\, c \cup \{z\} : c \in \varphi \,\}$ for some $z \notin \text{Var}(\varphi)$. We claim that $(\varphi, k) \in$ SMALL-UNSATISFIABLE-SUBSET if and only if $(\varphi', z, k) \in$ LOCAL-BACKBONE. A complete proof of this claim can be found in the full version of the paper. □

Lemma 2. LOCAL-BACKBONE $\leq_{\text{fpt}}$ SMALL-UNSATISFIABLE-SUBSET.

Proof. Let (φ, z, k) be an instance of LOCAL-BACKBONE. We construct an instance (ψ, k) of SMALL-UNSATISFIABLE-SUBSET. For every variable $x \in \text{Var}(\varphi)$ we take two copies x_1, x_2. For $i \in \{1, 2\}$ we let φ_i be a copy of φ using the variables x_i. Now we define $\psi = \varphi_1|_{z_1} \cup \varphi_2|_{\neg z_2}$. In other words, ψ is the union of two disjoint copies of the reducts of φ with respect to z and $\neg z$. We claim that $(\varphi, z, k) \in$ LOCAL-BACKBONE if and only if $(\psi, k) \in$ SMALL-UNSATISFIABLE-SUBSET. A complete proof of this claim can be found in the full version of the paper. □

Theorem 1. LOCAL-BACKBONE *is* W[1]*-complete.*

Proof. Since SMALL-UNSATISFIABLE-SUBSET is W[1]-complete [10], the result follows from Lemmas 1 and 2. □

4 Local Backbones of Horn Formulas

Restricting the problem of finding backbones in arbitrary formulas to Horn formulas reduces the classical complexity from co-NP-completeness to polynomial time solvability. It is a natural question whether the parameterized complexity of finding local backbones decreases in a similar way when the problem is restricted to Horn formulas. We will show that this is not the case. In order to do so, we define the parameterized problem SHORT-HYPERPATH, show that it is W[1]-hard, and then provide fpt-reductions from SHORT-HYPERPATH.

For a Horn formula φ and $s, t \in \mathrm{Var}(\varphi)$, we say that a subformula $\varphi' \subseteq \varphi$ is a *hyperpath* from s to t if (i) $t = s$ or (ii) $c = \{x_1, \ldots, x_n, t\} \in \varphi'$ and $\varphi' \backslash c$ is a hyperpath from s to x_i for each $1 \leq i \leq n$. If $|\varphi| \leq k$ then φ is called a *k-hyperpath*. The parameterized problem SHORT-HYPERPATH takes as input a Horn formula φ, two variables $s, t \in \mathrm{Var}(\varphi)$ and an integer k. The problem is parameterized by k. The question is whether there exists a k-hyperpath from s to t. For a more detailed discussion on the relation between (backward) hyperpaths in hypergraphs and hyperpaths as defined above, we refer to a survey article by Gallo et al. [12].

For the hardness proof of SHORT-HYPERPATH, we reduce from the W[1]-complete problem MULTICOLORED-CLIQUE [9]. The MULTICOLORED-CLIQUE problem takes as input a graph G, some integer k, and a proper k-coloring c of the vertices of G. The problem is parameterized by k. The question is whether there is a properly colored k-clique in G.

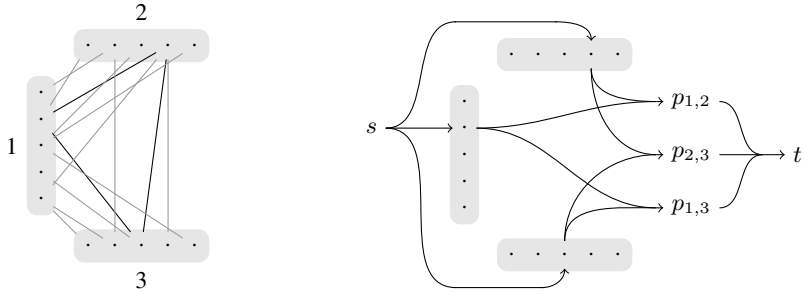
Lemma 3. SHORT-HYPERPATH *is* W[1]*-hard, even for instances* (φ, s, t, k) *where* $\varphi \in$ 3CNF.

Proof. We give a reduction from MULTICOLORED-CLIQUE. Let (G, k, c) be an instance of MULTICOLORED-CLIQUE, where $G = (V, E)$ and $V_1, \ldots, V_k$ are the equivalence classes of V induced by the k-coloring c. We construct an instance (φ, s, t, k') of SHORT-HYPERPATH, where $k' = k + \binom{k}{2} + 1$ and

$$\begin{aligned}
\mathrm{Var}(\varphi) &= \{s, t\} \cup V \cup \{\, p_{i,j} : 1 \leq i < j \leq k \,\};\\
\varphi &= \varphi_V \cup \varphi_p \cup \varphi_t;\\
\varphi_V &= \{\, \{\neg s, v\} : v \in V \,\};\\
\varphi_p &= \{\, \{\neg v_i, \neg v_j, p_{i,j}\} : 1 \leq i < j \leq k, v_i \in V_i, v_j \in V_j, \{v_i, v_j\} \in E \,\};\\
\varphi_t &= \{\{\, \neg p_{i,j} : 1 \leq i < j \leq k \,\} \cup \{t\}\}.
\end{aligned}$$

This construction is illustrated for an example with $k = 3$ in Figure 1. We claim that $(G, k, c) \in$ MULTICOLORED-CLIQUE if and only if $(\varphi, s, t, k') \in$ SHORT-HYPERPATH. A complete proof of this claim can be found in the full version of the paper.

To see that clauses of size at most 3 in the hyperpath suffice, we slightly adapt the reduction. The only clause we need to change is the single clause $e \in \varphi_t$. This clause e is of the form $\{\neg p_1, \ldots, \neg p_m, t\}$, for $m = \binom{k}{2}$. We introduce new variables $v_1, \ldots, v_m$ and replace e by the $m+1$ many clauses $\{\neg p_1, v_1\}$, $\{\neg v_{i-1}, \neg p_i, v_i\}$ for all $1 < i \leq m$ and $\{\neg v_m, t\}$. Clearly, the resulting Horn formula only has clauses of size at most 3. This adapted reduction works with the exact same line of reasoning as the reduction described above, with the only change that $k' = k + 2\binom{k}{2} + 1$.



(a) A 3-partite graph G with a clique (in black)

(b) The B-hyperpath in H of size $k' = 3 + \binom{3}{2} + 1$ from s to t corresponding to the clique

Fig. 1. Illustration of the reduction in the proof of Lemma 3 for the case of a 3-colored clique

Note that even the slightly stronger claim holds that G has a properly colored k-clique if and only if there exists a (subset) minimal k'-B-hyperpath $\varphi' \subseteq \varphi$ for which we have $|\varphi'| = k'$. □

We are now in a position to prove the W[1]-hardness of LOCAL-BACKBONE[HORN]. In fact, we show that finding local backbones is already W[1]-hard for definite Horn formulas with a single unit clause. We also show that this hardness crucially depends on allowing unit clauses in the formula, since for definite Horn formulas without unit clauses the problem is trivial. In fact, the complexity jumps to W[1]-hardness already when allowing a single unit clause.

Theorem 2. LOCAL-BACKBONE[DEFHORN ∩ 3CNF] *is* W[1]*-hard, already for instances* (φ, x, k) *where* φ *has at most one unit clause.*

Proof. We show W[1]-hardness by reducing from SHORT-HYPERPATH. Let (φ, s, t, k) be an instance of SHORT-HYPERPATH. We can assume that $\varphi \in$ 3CNF. We construct an instance (ψ_φ, t, k') of LOCAL-BACKBONE. Here $k' = k + 1$. For each $\varphi' \subseteq \varphi$ we define a formula $\psi_{\varphi'}$, by letting $\text{Var}(\psi_{\varphi'}) = \text{Var}(\varphi')$ and $\psi_{\varphi'} = \{\{s\}\} \cup \varphi'$. Clearly $\psi_\varphi \in$ DEFHORN ∩ 3CNF and ψ_φ has only a single unit clause. We claim that $(\psi_\varphi, t, k') \in$ LOCAL-BACKBONE if and only if $(\varphi, s, t, k) \in$ SHORT-HYPERPATH. A complete proof of this claim can be found in the full version of the paper. □

Also, restricting the problem to Horn formulas without unit clauses unfortunately does not yield fixed-parameter tractability.

Theorem 3. LOCAL-BACKBONE[NUHORN ∩ 3CNF] *is* W[1]*-hard.*

Proof. We show the W[1]-hardness of LOCAL-BACKBONE[NUHORN ∩ 3CNF] by reducing from SHORT-HYPERPATH. Let (φ, s, t, k) be an instance of SHORT-HYPERPATH. We can assume without loss of generality that $\varphi \in$ 3CNF, and that each clause in which t occurs positively is of size 3. We construct an instance (ψ_φ, x_s, k) of LOCAL-BACKBONE. For each $\varphi' \subseteq \varphi$ we define a formula $\psi_{\varphi'}$.

$$\begin{aligned}\psi_{\varphi'} = &\{\{\neg x_a, \neg x_b, x_c\} : \{\neg a, \neg b, c\} \in \varphi', c \neq t\} \cup \\ &\{\{\neg x_a, \neg x_b\} : \{\neg a, \neg b, t\} \in \varphi'\} \\ &\{\{\neg x_a, x_b\} : \{\neg a, b\} \in \varphi'\}\end{aligned}$$

Clearly we have that $\psi_\varphi \in$ HORN∩3CNF and that ψ_φ has no unit clauses. We claim that $(\psi_\varphi, x_s, k) \in$ LOCAL-BACKBONE if and only if $(\varphi, s, t, k) \in$ SHORT-HYPERPATH. A complete proof of this claim can be found in the full version of the paper. □

5 Local Backbones of Krom Formulas

We have seen that finding local backbones is already fixed-parameter intractable for Horn formulas, for which finding backbones is tractable. We show that Krom formulas have a similar property: even though finding backbones in Krom formulas is tractable, finding local backbones is fixed-parameter intractable.

Theorem 4. LOCAL-BACKBONE[KROM] *is* W[1]*-hard.*

Proof. We reduce from MULTICOLORED-CLIQUE. Let (G, k, c) be an instance of MULTICOLORED-CLIQUE, where $G = (V, E)$ and $V_1, \ldots, V_k$ are the equivalence classes of V induced by the k-coloring c. We construct an instance (φ, x, k') of LOCAL-BACKBONE[KROM]. Intuitively, we introduce a gadget for each V_i (see Figure 2a) and additionally a gadget for each pair (i, j) for $1 \leq i < j \leq k$ (see Figure 2b), and sequentially link these gadgets together (see Figure 2c). In the definition of (φ, x, k') for each $1 \leq i \leq k$ we define a formula φ_i^{guess} that corresponds to the gadget for V_i, and for each $1 \leq i < j \leq k$ we define a formula $\varphi_{i,j}^{\text{check}}$ that corresponds to the gadget for the pair (i, j). In the construction of φ we use variables $\sigma_{i,v}^j$ and $\tau_{i,v}^j$ that are used to encode the choice of vertex v in V_i for the clique, and that are used to verify whether v and the choice for V_j are connected. We let $x = g_1$ and $k' = k(2k+1) + 3\binom{k}{2} + 2$ and we define

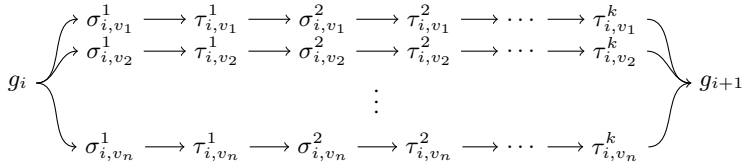
$$\begin{aligned}\text{Var}(\varphi) = {} & \{g_1, \ldots, g_{k+1}\} \cup \{c_{1,1}, c_{1,2} \ldots, c_{k-1,k}, c_{k,k+1}\} \cup \\ & \{\, \sigma_{i,v}^j, \tau_{i,v}^j : 1 \leq i \leq k, 1 \leq j \leq k, v \in V_i \,\}, \text{ and} \\ \varphi = {} & \bigcup_{1 \leq i \leq k} \varphi_i^{\text{guess}} \cup \bigcup_{1 \leq i < j \leq k} \varphi_{i,j}^{\text{check}} \cup \{\{\neg g_{k+1}, c_{1,1}\}, \{\neg c_{k,k+1}, \neg g_1\}\}.\end{aligned}$$

For each $1 \leq i \leq k$, we define φ_i^{guess}, where $V_i = \{v_1, \ldots, v_n\}$, by letting

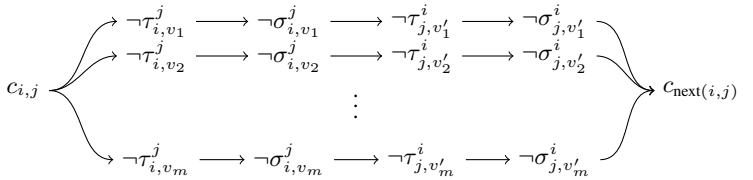
$$\begin{aligned}\varphi_i^{\text{guess}} = {} & \{\, \{\neg g_i, \sigma_{i,v_l}^1\} : 1 \leq l \leq n \,\} \cup \\ & \{\, \{\neg \sigma_{i,v_l}^j, \tau_{i,v_l}^j\} : 1 \leq j \leq k, 1 \leq l \leq n \,\} \cup \\ & \{\, \{\neg \tau_{i,v_l}^j, \sigma_{i,v_l}^{j+1}\} : 1 \leq j < k, 1 \leq l \leq n \,\} \cup \\ & \{\, \{\neg \tau_{i,v_l}^k, g_{i+1}\} : 1 \leq l \leq n \,\}.\end{aligned}$$

Similarly, for each $1 \leq i < j \leq k$ we define the subformula $\varphi_{i,j}^{\text{check}}$ as follows. Here we let $E \cap (V_i \times V_j) = \{(v_1, v_1'), \ldots, (v_m, v_m')\}$. Also, we define the function next by letting $\text{next}(i, j) = (i, j+1)$ if $j \neq k$ and $\text{next}(i, j) = (i+1, i+2)$ if $j = k$.

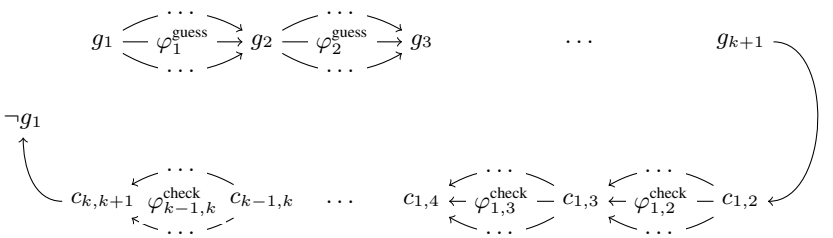
$$\begin{aligned}\varphi_{i,j}^{\text{check}} = {} & \{\, \{\neg c_{i,j}, \neg \tau_{i,v_l}^j\} : 1 \leq l \leq m \,\} \cup \\ & \{\, \{\tau_{i,v_l}^j, \neg \sigma_{i,v_l}^j\}, \{\sigma_{i,v_l}^j, \neg \tau_{j,v_l'}^i\}, \{\tau_{j,v_l'}^i, \neg \sigma_{j,v_l'}^i\} : 1 \leq l \leq m \,\} \cup \\ & \{\, \{\sigma_{j,v_l'}^i, c_{\text{next}(i,j)}\} : 1 \leq l \leq m \,\}.\end{aligned}$$



(a) Gadget φ_i^{guess} for partition $V_i = \{v_1, \ldots, v_n\}$



(b) Gadget $\varphi_{i,j}^{\text{check}}$ for partitions i and j and $E \cap (V_i \times V_j) = \{(v_1, v_1'), \ldots, (v_m, v_m')\}$



(c) Linking the gadgets together in φ

Fig. 2. Gadgets for the reduction in the proof of Theorem 4

Intuitively, this reduction works as follows. Note that since g_1 occurs only negatively in φ we know that g_1 can only be a k'-backbone of φ if there exists a path from g_1 to $\neg g_1$ in $\text{impl}(\varphi)$ that uses at most k' clauses. A path of length $2k+1$ through $\text{impl}(\varphi_i^{\text{guess}})$ corresponds to guessing a vertex in the equivalence class V_i. Additionally, a path of length 5 through $\text{impl}(\varphi_{i,j}^{\text{check}})$ corresponds to verifying whether there is an edge in the graph that is in $V_i \times V_j$. So clearly, there exists a path of length $k(2k+1) + 5\binom{k}{2} + 2$ in $\text{impl}(\varphi)$ from g_1 to $\neg g_1$. However, many clauses in $\text{impl}(\varphi_{i,j}^{\text{check}})$ can be doubly used clauses, already used before in paths through $\text{impl}(\varphi_i^{\text{guess}})$. Concretely, there exists a path through $\text{impl}(\varphi_{i,j}^{\text{check}})$ that uses only 3 clauses that have not yet been used in paths through $\text{impl}(\varphi_i^{\text{guess}})$ if and only if the paths through $\text{impl}(\varphi_i^{\text{guess}})$ and $\text{impl}(\varphi_j^{\text{guess}})$ have selected vertices $v_i \in V_i$ and $v_j \in V_j$ such that $(v_i, v_j) \in E$. In other words, there exists a path in $\text{impl}(\varphi)$ from g_1 to $\neg g_1$ that uses k' clauses if and only if G has a properly colored k-clique.

In the full version of the paper, we formally prove that G has a properly colored k-clique if and only if g_1 is a k'-backbone of φ. □

We would like to point out that all complexity results for the various restrictions of LOCAL-BACKBONE also hold for SMALL-UNSATISFIABLE-SUBSET under the corresponding restrictions. This is because the reduction in the proof of Lemma 2 works for all classes of formulas that are closed under variable instantiations. For instance, the reduction in the proof of Lemma 2 together with Theorem 3 tells us that SMALL-UNSATISFIABLE-SUBSET[HORN ∩ 3CNF] is W[1]-hard. This does not follow from the reduction that Fellows et al. [10] use to prove the W[1]-hardness of SMALL-UNSATISFIABLE-SUBSET. In particular, the following previously unstated results hold.

Corollary 1. SMALL-UNSATISFIABLE-SUBSET[$\mathcal{C}$] *is* W[1]*-hard for each* $\mathcal{C} \in \{$DEFHORN ∩ 3CNF, NUHORN ∩ 3CNF, KROM$\}$.

In fact, these fixed-parameter intractability results for SMALL-UNSATISFIABLE-SUBSET give us the following NP-hardness results. Interestingly, for the case of KROM formulas this result contrasts with the known result that finding minimal resolution refutations for KROM formulas can be done in polynomial time [3,4].

Corollary 2. *Let* $\mathcal{C} \in \{$KROM, 3CNF ∩ DEFHORN, 3CNF ∩ NUHORN$\}$. *Given a formula* $\varphi \in \mathcal{C}$ *and an integer* k*, deciding whether* φ *contains an unsatisfiable subset of size* $\leq k$ *is* NP*-hard.*

6 Local Backbones of Formulas with Bounded Variable Occurrence

When considering the restriction of LOCAL-BACKBONE to formulas where variables occur a bounded number of times, we get a fixed-parameter tractability result at last. This fixed-parameter tractability result is closely related to the result that SMALL-UNSATISFIABLE-SUBSET is fixed-parameter tractable for instances restricted to classes of formulas that have locally bounded treewidth [10]. Fellows et al. used a meta theorem to prove this. We give a direct algorithm to solve SMALL-UNSATISFIABLE-SUBSET[VO$_d$] in fixed-parameter linear time.

Let (φ, k) be an instance of SMALL-UNSATISFIABLE-SUBSET[VO$_d$]. The following procedure decides whether there exists an unsatisfiable subset $\varphi' \subseteq \varphi$ of size at most k, and computes such a subset if it exists. We let $\varphi^\star = \{\, c \in \varphi : |c| < k \,\}$. It suffices to consider subsets of $\varphi^\star$, since any unsatisfiable subset $\varphi' \subseteq \varphi$ contains a minimally unsatisfiable subset $\varphi'' \subseteq \varphi'$, and by Tarsi's Lemma we know that φ'' contains only clauses of size smaller than k.

Without loss of generality, we assume that the incidence graph of $\varphi^\star$ is connected. Otherwise, we can solve the problem by running the algorithm on each of the connected components. We guess a clause $c \in \varphi^\star$, we let $F_1 := \{c\}$, and we let all variables be unmarked initially. We compute F_{i+1} for $1 \leq i \leq k$ by means of the following (nondeterministic) rule:

1. take an unmarked variable $z \in \text{Var}(F_i)$;
2. guess a non-empty subset $F'_z \subseteq F_z$ for $F_z = \{\, c \in \varphi^\star : z \in \text{Var}(c) \,\}$;
3. let $F_{i+1} := F_i \cup F'_z$;
4. mark z.

If at any point all variables in F_i are marked, we stop computing F_{i+1}. For any F_i, if $|F_i| > k$ we fail. For each F_i, we check whether F_i is unsatisfiable. If it is unsatisfiable, we return with $\varphi' = F_i$. If it is satisfiable and if it contains no unmarked variables, we fail. It is easy to see that this algorithm is sound. If some $\varphi' \subseteq \varphi^\star$ is returned, then φ' is unsatisfiable and $|\varphi'| \leq k$. In order to see that the algorithm is complete, assume that there exists some unsatisfiable $\varphi' \subseteq \varphi^\star$ with $|\varphi'| \leq k$. Then, since we know that the incidence graph of F' is connected, we know that F' can be constructed as one of the F_i in the algorithm.

To see that this algorithm witnesses fixed-parameter linearity, we bound its running time. We have to execute the search process at most once for each clause of $\varphi^\star$. At each point in the execution of the algorithm, F_i contains at most k variables. Therefore, there are at most k choices to take an unmarked variable z. Since each variable occurs in at most d clauses, for each F_z used in the rule we know $|F_z| \leq d$. Thus, there are at most 2^d possible guesses for F'_z in each execution of the rule. Since we iterate the rule at most k times, we consider at most $(k2^d)^k$ sets F', each of size $O(k^2)$. Thus each (un)satisfiability check can be done in $O(2^k)$ time. Therefore, the total running time of the algorithm is $O(k^k 2^{dk} n)$, for n the size of the instance.

This algorithm also gives us a direct algorithm that shows that LOCAL-BACKBONE$[\mathrm{VO}_d]$ is fixed-parameter linear.

Theorem 5. LOCAL-BACKBONE$[\mathrm{VO}_d]$ *is fixed-parameter linear.*

Proof. The result follows directly by using the reduction in the proof of Lemma 2 in combination with the above algorithm. □

7 Iterative Local Backbones

We now consider the (parameterized) complexity of finding iterative local backbones. It is easy to see that ITERATIVE-LOCAL-BACKBONE is in para-NP. This is witnessed by a straightforward nondeterministic fpt-algorithm, that guesses a sequence of n witnesses (φ_i, l_i) with $|\varphi_i| \leq k$, and that verifies whether $\varphi_i \subseteq \varphi|_{\{l_1,\ldots,l_{i-1}\}}$ and whether $\varphi_i \models l_i$.

Some of the results we obtained for the problem of finding local backbones can be carried over.

Theorem 6. *Let $\mathcal{C}$ be a class of formulas such that* LOCAL-BACKBONE$[\mathcal{C}]$ *is fixed-parameter tractable and $\mathcal{C}$ is closed under variable instantiation. Then* ITERATIVE-LOCAL-BACKBONE$[\mathcal{C}]$ *is fixed-parameter tractable.*

Proof. We give an algorithm to solve ITERATIVE-LOCAL-BACKBONE$[\mathcal{C}]$ that calls a subroutine to solve instances of SMALL-UNSATISFIABLE-SUBSET$[\mathcal{C}]$. This algorithm is given in the form of pseudo-code as Algorithm 1. By the fact that $\mathcal{C}$ is closed under variable instantiations we are able to apply the reduction in the proof of Lemma 2. Thus, we can assume that the question of whether some $\varphi \in \mathcal{C}$ contains an unsatisfiable subset of size at most k can be solved in $f(k)||\varphi||^c$ time, for some computable function f and some constant c. Then, the entire algorithm runs in $O(f(k)||\varphi||^{c+2})$ time. This proves the claim. □

input : an instance (φ, x, k) of ITERATIVE-LOCAL-BACKBONE
output: yes iff $(\varphi, x, k) \in$ ITERATIVE-LOCAL-BACKBONE

$\psi \leftarrow \varphi$; $\quad \mathsf{conseq} \leftarrow \emptyset$;
for $i \leftarrow 1$ **to** $|\mathrm{Lit}(\varphi)|$ **do**
  foreach *literal* $l \in \mathrm{Lit}(\psi)$ **do**
    if $(\psi|_{\bar{l}}, k) \in$ SMALL-UNSATISFIABLE-SUBSET **then**
      $\mathsf{conseq} \leftarrow \mathsf{conseq} \cup \{l\}$;
  $\psi \leftarrow \psi|_{\mathsf{conseq}}$;
return $\{x, \neg x\} \cap \mathsf{conseq} \neq \emptyset$

Algorithm 1. Deciding ITERATIVE-LOCAL-BACKBONE with a SMALL-UNSATISFIABLE-SUBSET oracle

Corollary 3. ITERATIVE-LOCAL-BACKBONE[NUHORN $\cap$ 3CNF] *is* W[1]*-hard.*

Proof. Observe that the proofs of Lemma 3 and Theorem 3 imply that it is already W[1]-hard to determine whether a formula $\varphi \in$ NUHORN$\cap$3CNF has a subset $\varphi' \subseteq \varphi$ of size exactly k witnessing that any $x \in \mathrm{Var}(\varphi)$ is a k-backbone. From this, it immediately follows that determining whether $(\varphi, x, k) \in$ ITERATIVE-LOCAL-BACKBONE is W[1]-hard as well. □

We identify several tractable cases for ITERATIVE-LOCAL-BACKBONE. The problem of finding iterative local backbones in definite Horn formulas is polynomial time solvable. Similarly, finding iterative local backbones in Krom formulas is solvable in polynomial time as well. Interestingly, for these restrictions the problem of finding (non-iterative) local backbones remains W[1]-hard. In order to show that finding iterative local backbones in definite Horn formulas is tractable, we will use the following observation.

Observation 1 *Let φ be any propositional formula, let l be any literal such that there exists a $\varphi' \subseteq \varphi$ with $|\varphi'| \leq k$ and $\varphi' \models l$, and let $\psi = \varphi|_l$. Then $x \in \mathrm{Var}(\psi)$ is an iterative k-backbone of ψ if and only if it is an iterative k-backbone of φ.*

Theorem 7. ITERATIVE-LOCAL-BACKBONE[DEFHORN] *is in* P.

Proof. We show that for any definite Horn formula φ and any $k \geq 1$ the set of iterative k-backbones of φ coincides with the set of variables $x \in \mathrm{Var}(\varphi)$ such that $\varphi \models x$. The claim then follows, since the entailment relation $\models$ can be decided in linear time for definite Horn formulas [6].

Fix an arbitrary integer $k \geq 1$ and an arbitrary definite Horn formula φ. Since definite Horn formulas cannot entail negative literals, we know that each iterative k-backbone x of φ is also a semantic consequence of φ. Now, let $x \in \mathrm{Var}(\varphi)$ be an arbitrary atom and assume that $\varphi \models x$. So there exist variables $x_1, \ldots, x_m \in \mathrm{Var}(\varphi)$ such that $x_m = x$ and for each x_i we have either (i) $\{x_i\} \in \varphi$ or (ii) $\{\neg x_{i_1}, \ldots, \neg x_{i_l}, x_i\} \in \varphi$ for some $i_1 < \cdots < i_l < i$. We prove by induction on m that each x_i is an iterative k-backbone. Take an arbitrary x_i. By the induction hypothesis, we can assume that every x_j for $j < i$ is an iterative k-backbone of φ. We proceed by case distinction for the justification of

x_i in the sequence. In case (i), we know that $\{x_i\} \in \varphi$. Therefore, it directly follows that x_i is a k-backbone of φ, and thus is an iterative k-backbone too. In case (ii), we know that $\{\neg x_{i_1}, \ldots, \neg x_{i_l}, x_i\} \in \varphi$ for some $i_1 < \cdots < i_l < i$. By the induction hypothesis, we know that each x_{i_j} is an iterative k-backbone of φ. By assumption, we have that $\varphi \models x_{i_j}$ for each x_{i_j}. By Observation 1, we get that x_i is an iterative k-backbone of φ if and only if it is an iterative k-backbone of $\varphi_{\{x_{i_1},\ldots,x_{i_l}\}}$. It holds that $\{x_i\} \in \varphi_{\{x_{i_1},\ldots,x_{i_l}\}}$. Thus, x_i is an iterative k-backbone of φ. □

Theorem 8. ITERATIVE-LOCAL-BACKBONE[KROM] *is in* P.

Proof. We show that the iterative k-backbones of a Krom formula φ coincide with those backbones of φ that can be identified by iterated application of the following rule: if the implication graph of φ contains a path from a literal $l \in \{x, \neg x\}$ to its complement $\overline{l}$ of length at most k, conclude that x is a backbone and set $\varphi := \varphi|_{\overline{l}}$. Detection of such a path can be done in polynomial time. Also, at most $O(|\mathrm{Var}(\varphi)|)$ iterated applications of this rule suffice to reach a fixpoint. All that remains is to show the correspondence.

The correspondence claim follows from the following property. Let $l \in \mathrm{Lit}(\varphi)$. If $\mathrm{impl}(\varphi)$ contains a path $\overline{l} \to^* l$ that uses at most k clauses and that doubly uses m of these clauses, then there exist literals $l_1, \ldots, l_{m+1} \in \mathrm{Lit}(\varphi)$ such that (i) $l_{m+1} = l$ and (ii) for each $1 \leq i \leq m+1$ the graph $\mathrm{impl}(\varphi|_{\{l_1,\ldots,l_{i-1}\}})$ contains a path $\overline{l_i} \to^* l_i$ that uses at most k clauses and does not doubly use any clause. We prove this claim by induction on m. The case for $m = 0$ is trivial. Consider the case for $m \geq 1$. Since the path $\overline{l} \to^* l$ doubly uses some clause, we know that $\overline{l} \to^* a \to \overline{b} \to^* b \to \overline{a} \to^* l$, for some $a, b \in \mathrm{Lit}(\varphi)$. We can assume without loss of generality that the path $\overline{b} \to^l b$ does not doubly use any clause. If this is not the case, the path $\overline{b} \to^l b$ contains a subpath $\overline{c} \to^* c$ that does not doubly uses any clauses, and we could select c instead of b. Also, we know that $l \leq k$. It is easy to see that $\mathrm{impl}(\varphi|_b)$ contains the path $\overline{l} \to^* a \to \overline{a} \to^* l$, which uses at most k clauses and doubly uses $m-1$ of these clauses. By the induction hypothesis, we obtain that there exist $l'_1, \ldots, l'_m$ such that $l'_m = l$ and for each $1 \leq i \leq m$ the graph $\mathrm{impl}(\varphi|_{\{l'_1,\ldots,l'_{i-1}\}})$ contains a path $\overline{l'_i} \to^* l'_i$ that uses at most k clauses and does not doubly use any clause. Now let $l_1 = b$ and $l_i = l'_{i-1}$ for all $2 \leq i \leq m+1$. It is straightforward to verify that $l_1, \ldots, l_{m+1}$ satisfy the required properties. □

Somewhat related to our mechanism of computing enforced assignments via iterated k-backbones is the mechanism used to define *unit-refutation complete formulas of level* k [13,19]. This mechanism is based on mappings r_k from CNF formulas to CNF formulas. For a nonnegative integer k, the mapping r_k is defined inductively as follows. In the case for $k = 0$, we let $r_0(\varphi) = \{\bot\}$ if $\bot \in \varphi$, and $r_0(\varphi) = \varphi$ otherwise. In the case for $k > 0$, we let $r_k(\varphi) = r_k(\varphi|_l)$ if there exists a literal $l \in \mathrm{Lit}(\varphi)$ such that $r_{k-1}(\varphi|_{\overline{l}}) = \{\bot\}$, and $r_k(\varphi) = \varphi$ otherwise. In particular, the mapping r_1 computes the result of applying unit propagation. Note that the result of $r_k(\varphi)$ is the application of a number of forced assignments to φ, i.e., $r_k(\varphi) = \varphi|_L$ for some $L \subseteq \mathrm{Lit}(\varphi)$ such that for all $l \in L$ we have $\varphi \models l$. We let $L_k^{\mathrm{UC}}(\varphi)$ denote the set of forced literals that

are computed by r_k, i.e., $L_k^{\mathrm{UC}}(\varphi) = L \subseteq \mathrm{Lit}(\varphi)$ such that $r_k(\varphi) = \varphi|_L$. Similarly, we let $L_k^{\mathrm{ILB}}(\varphi)$ denote the set of forced literals that are found by computing iterative k-backbones.

The following observations relate the two mechanisms. Let φ be an arbitrary CNF formula. We have that $L_1^{\mathrm{UC}}(\varphi) = L_1^{\mathrm{ILB}}(\varphi)$. In fact, this set contains exactly those enforced literals that can be found by unit propagation. Also, for any $k \geq 2$ we have that $L_k^{\mathrm{ILB}}(\varphi) \subsetneq L_k^{\mathrm{UC}}(\varphi)$. The inclusion follows from the fact that each minimal subset φ' of size at most k that enforces a literal l has at most k literals (which is a direct result of Tarsi's Lemma). Whenever l is identified as an enforced literal in iterative k-backbone computation, it can then also be computed by r_k by first guessing $\overline{l}$, and subsequently obtaining a contradiction for each instantiation of the other variables in $\mathrm{Var}(\varphi')$. In order to see that the inclusion is strict, consider the family of formulas $(\varphi_n)_{n\in\mathbb{N}}$, where $\varphi_n = \{\,\{\neg x_i, x_{i+1}\} : 1 \leq i < n\,\} \cup \{\neg x_n, \neg x_1\}$. For each φ_n, we know that $\varphi_n \models \neg x_1$. Furthermore, we have that $\neg x_1 \in L_2^{\mathrm{UC}}(\varphi_n)$, but x_1 is not an iterative k-backbone of φ_n for any $k < n$.

8 Experimental Results

In order to illustrate the relevance of the concept of local backbones and iterative local backbones, we provide some empirical evidence of the distribution of (iterative) local backbones in instances from different domains. We considered both randomly generated instances (3CNF instances with various variable-clause ratios around the phase transition) and instances originating from planning [15,17], circuit fault analysis [23], inductive inference [23], and bounded model checking [26]. We considered only satisfiable instances. For practical reasons, we used a method that gives us a lower bound on the number of k-backbone variables. By reducing the separate LOCAL-BACKBONE problems to SMALL-UNSATISFIABLE-SUBSET, we can use algorithms computing subset-minimal unsatisfiable subsets to approximate the number of iterative local backbones (we used MUSer2 [2]). In order to get the exact number, we would have to compute cardinality-minimal unsatisfiable subsets, which is difficult in practice.

The experimental results are shown in Figure 3. For each of the instances, we give the percentage of backbones that are of order k (dashed lines) and the percentage of backbones that are of iterative order k (solid lines), as well as the total number of backbones and the total number of clauses. There are instances with several backbones, most of which have relatively small order. This is the case for the instances from the domains of planning (*logistics*), circuit fault analysis (*ssa7552*) and bounded model checking (*bmc-ibm*). It is worth noting that already more than 75 percent of the backbones in all the considered *bmc-ibm* instances are of iterative order 2. We also found instances that have no backbones of small order or of small iterative order. This is the case for the instances from the domain of inductive inference (*ii32*) and the randomly generated instances. Some of these instances do have backbones, while others have no backbones at all. It would be interesting to confirm these findings by a more rigorous experimental investigation.

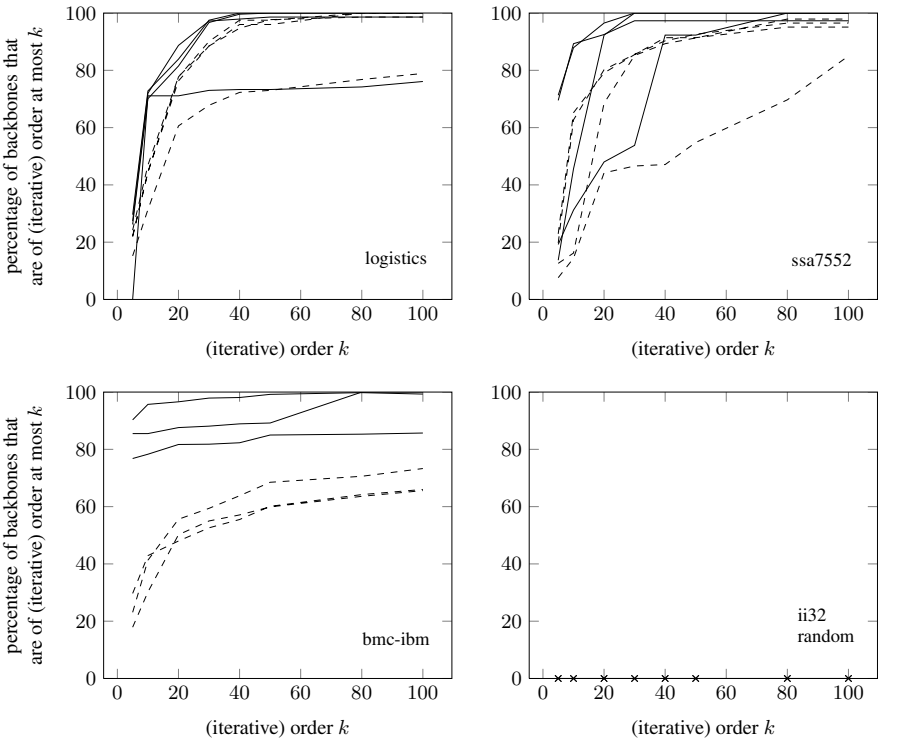


Fig. 3. Percentage of backbones that are of order at most k (dashed) and of iterative order at most k (solid), for SAT instances from planning (*logistics.[a–d]*, 828–4713 variables, 6718–21991 clauses, 437–838 backbones), circuit fault analysis (*ssa7552-[038,158–160]*, 1363–1501 variables, 3032–3575 clauses, 405–838 backbones), bounded model checking (*bmc-ibm-[2,5,7]*, 2810–9396 variables, 11683–41207 clauses, 405–557 backbones), inductive inference (*ii32[b–e][1–3]*, 222–824 variables, 1186–20862 clauses, 0–208 backbones) and random 3SAT instances (*random*, 200 variables, 820–900 clauses, 1–131 backbones).

9 Conclusions

We have drawn a detailed complexity map of the problem of finding local backbones and iterative local backbones, in general and for formulas from restricted classes. Additionally, we have provided some first empirical results on the distribution of (iterative) local backbones in some benchmark SAT instances. We found that in structured instances from different domains backbones are of quite low (iterative) order. This suggests that the notions of local backbones and iterative local backbones can be used to identify structure in SAT instances.

Some of our findings are somewhat surprising. (1) Finding local backbones in Horn and Krom formulas is fixed-parameter intractable, whereas backbones for these classes of formulas can be found in polynomial time. (2) In certain cases finding iterative local backbones is computationally easier than finding (non-iterative) local backbones.

(3) Local backbones and iterative local backbones seem to be a better indicator of structure than backbones. Random instances do have backbones, but these are of high order and iterative order.

Backbones and local backbones are implied unit clauses. It might be interesting to extend our investigation to implied clauses of larger fixed size, binary clauses in particular.

References

1. Aharoni, R., Linial, N.: Minimal non-two-colorable hypergraphs and minimal unsatisfiable formulas. J. Combin. Theory Ser. A 43, 196–204 (1986)
2. Belov, A., Marques-Silva, J.: MUSer2: An efficient MUS extractor. J. on Satisfiability, Boolean Modeling and Computation 8(1/2), 123–128 (2012)
3. Buresh-Oppenheim, J., Mitchell, D.: Minimum 2CNF resolution refutations in polynomial time. In: Marques-Silva, J., Sakallah, K.A. (eds.) SAT 2007. LNCS, vol. 4501, pp. 300–313. Springer, Heidelberg (2007)
4. Buresh-Oppenheim, J., Mitchell, D.: Minimum witnesses for unsatisfiable 2CNFs. In: Biere, A., Gomes, C.P. (eds.) SAT 2006. LNCS, vol. 4121, pp. 42–47. Springer, Heidelberg (2006)
5. Darwiche, A., Marquis, P.: A knowledge compilation map. J. Artif. Intell. Res. 17, 229–264 (2002)
6. Dowling, W.F., Gallier, J.H.: Linear-time algorithms for testing the satisfiability of propositional horn formulae. J. Logic Programming 1(3), 267–284 (1984)
7. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Monographs in Computer Science. Springer, New York (1999)
8. Dubois, O., Dequen, G.: A backbone-search heuristic for efficient solving of hard 3-SAT formulae. In: Nebel, B. (ed.) Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, pp. 248–253 (2001)
9. Fellows, M.R., Hermelin, D., Rosamond, F.A., Vialette, S.: On the parameterized complexity of multiple-interval graph problems. Theoretical Computer Science 410(1), 53–61 (2009)
10. Fellows, M.R., Szeider, S., Wrightson, G.: On finding short resolution refutations and small unsatisfiable subsets. Theoretical Computer Science 351(3), 351–359 (2006)
11. Flum, J., Grohe, M.: Parameterized Complexity Theory. Texts in Theoretical Computer Science. An EATCS Series, vol. XIV. Springer, Berlin (2006)
12. Gallo, G., Longo, G., Pallotino, S., Nguyen, S.: Directed hypergraphs and applications. Discrete Applied Mathematics 42, 177–201 (1993)
13. Gwynne, M., Kullmann, O.: Generalising and unifying SLUR and unit-refutation completeness. In: van Emde Boas, P., Groen, F.C.A., Italiano, G.F., Nawrocki, J., Sack, H. (eds.) SOFSEM 2013. LNCS, vol. 7741, pp. 220–232. Springer, Heidelberg (2013)
14. Hertli, T., Moser, R.A., Scheder, D.: Improving PPSZ for 3-SAT using critical variables. In: Schwentick, T., Dürr, C. (eds.) Symposium on Theoretical Aspects of Computer Science, vol. 9, pp. 237–248. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2011)
15. Hoos, H.H., Stützle, T.: SATLIB: An online resource for research on SAT. In: Gent, I., van Maaren, H., Walsh, T. (eds.) SAT 2000: Highlights of Satisfiability Research in the year 2000. Frontiers in Artificial Intelligence and Applications, pp. 283–292. Kluwer Academic (2000)
16. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? J. of Computer and System Sciences 63(4), 512–530 (2001)
17. Kautz, H., Selman, B.: Pushing the envelope: planning, propositional logic, and stochastic search. In: Proceedings of the Thirteenth AAAI Conference on Artificial Intelligence, AAAI 1996, pp. 1194–1201. AAAI Press (1996)

18. Kilby, P., Slaney, J.K., Thiébaux, S., Walsh, T.: Backbones and backdoors in satisfiability. In: Proceedings of the Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, AAAI 2005, Pittsburgh, Pennsylvania, USA, July 9-13, pp. 1368–1373 (2005)
19. Kullmann, O.: Investigating a general hierarchy of polynomially decidable classes of cnf's based on short tree-like resolution proofs. Electronic Colloquium on Computational Complexity (ECCC) 6(41) (1999)
20. Kullmann, O.: An application of matroid theory to the SAT problem. In: Fifteenth Annual IEEE Conference on Computational Complexity, pp. 116–124 (2000)
21. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford Lecture Series in Mathematics and its. Applications. Oxford University Press, Oxford (2006)
22. Parkes, A.J.: Clustering at the phase transition. In: Proceedings of the Fourteenth National Conference on Artificial Intelligence, AAAI 1997, pp. 340–345. AAAI Press (1997)
23. Prelotani, D.: Efficiency and stability of hypergraph SAT algorithms. In: Johnson, D.S., Trick, M.A. (eds.) Cliques, Coloring and Satisfiability, pp. 479–498. AMS (1996)
24. Schneider, J., Froschhammer, C., Morgenstern, I., Husslein, T., Singer, J.M.: Searching for backbones – an efficient parallel algorithm for the traveling salesman problem. Computer Physics Communications 96, 173–188 (1996)
25. Slaney, J.K., Walsh, T.: Backbones in optimization and approximation. In: Nebel, B. (ed.) Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, pp. 254–259 (2001)
26. Strichman, O.: Tuning SAT checkers for bounded model checking. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 480–494. Springer, Heidelberg (2000)