Trivial, Tractable, Hard. A Not So Sudden Complexity Jump in Neighborhood Restricted CNF Formulas

Dominik Scheder* ** ***

Aarhus University

Abstract. For a CNF formula F we define its 1-conflict graph as follows: Two clauses $C, D \in F$ are connected by an edge if they have a nontrivial resolvent – that is, if there is a unique literal $u \in C$ for which $\bar{u} \in D$. Let $\mathrm{lc}_1(F)$ denote the maximum degree of this graph.
A k-CNF formula is a CNF formula in which each clause has exactly k distinct literals. We show that (1) a k-CNF formula F with $\mathrm{lc}_1(F) \leq k-1$ is satisfiable; (2) there are unsatisfiable k-CNF formulas F with $\mathrm{lc}_1(F) = k$; (3) there is a polynomial time algorithm deciding whether a k-CNF formula F with $\mathrm{lc}_1(F) = k$ is satisfiable; (4) satisfiability of k-CNF formulas F with $\mathrm{lc}_1(F) \leq k+1$ is NP-hard.
Furthermore, we show that if F is a k-CNF formula and $\mathrm{lc}_1(F) \leq k$, then we can find in polynomial time a satisfying assignment (if F is satisfiable) or a treelike resolution refutation with at most $|F|$ leaves (if F is unsatisfiable). Here, $|F|$ is the number of clauses of F.

1 Introduction

There are several parameters to measure the structural complexity of CNF formulas, and they influence the computational complexity of their associated satisfiability decision problem. Some of them yield a fixed-parameter tractable problem – for example the treewidth of formulas (Allender, Chen, Lou, Papakonstantinou, and Tang [1]). For other parameters we are hit by the full power of NP-completeness once the parameter is large enough. Think of k, the maximum clause width of a formula: For $k = 2$ we know polynomial algorithms, for $k \geq 3$ the problem is NP-complete. In this paper we define in a natural way a graph on the clauses of the formula and investigate the complexity of the satisfiability problem depending on the maximum degree of this graph. We connect two

* The author acknowledges support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, and from the CFEM research center (supported by the Danish Strategic Research Council), within which this work was performed.

** Research was supported by the SNF Grant 200021-118001/1.

*** The author acknowledges support from the Simons Institute for the Theory of Computing, UC Berkeley.

clauses C, D of our formula with an edge if those clauses have a non-trivial resolvent. That is, if there is exactly one literal $u \in C$ for which $\bar{u} \in D$. We call this the 1-conflict graph. Thus, the degree of a clause C in this graph is the number of potential resolution partners in the formula. This graph is similar to the one defined by Ostrowski, Grégoire, Mazure, and Sais [2].[1] We show that a k-CNF formula is satisfiable if its 1-conflict graph has maximum degree at most $k-1$; the satisfiability problem is NP-hard if we allow a maximum degree of $k+1$; in between, for k-CNF formulas graphs of maximum degree k, there is a nontrivial algorithm that runs in polynomial time. If the formula is satisfiable, the algorithm returns a satisfying assignment. If it is unsatisfiable, it returns a treelike resolution refutation of size at most $2m-1$, where m is the number of clauses.

1.1 Notions of Degree, Neighborhood, and Conflict

A k-CNF formula in which every variable appears in at most $2^k/(ek)$ clauses is satisfiable. This is a direct consequence of the Lovász Local Lemma [3] and was first observed by Kratochvíl, Savický, and Tuza [4]. There is no reason to believe that $2^k/(ek)$ is tight. This motivates the following definition: Let $f(k)$ be the largest integer d such that every k-CNF formula F with $\Delta(F) \leq d$ is satisfiable. Here, Δ is the "maximum degree" of a formula: The maximum number of clauses in which a variable appears. The above result shows that $f(k) \geq 2^k/(ek)$. Proving matching upper bounds, i.e., constructing unsatisfiable k-CNF formulas of low maximum variable degree, turned out to be not trivial at all. The upper bound has been improved in several papers, to $O\left(2^k/k^{0.26}\right)$ by Savický and Sgall [5] and to $O\left(2^k \log k/k\right)$ by Hoory and Szeider [6]. Gebauer [7] improved it to $O\left(2^k/k\right)$, which is tight up to a constant factor, and finally Gebauer, Szabó, Tardos [8] proved that $f(k) = (1 \pm o(1))2^{k+1}/ek$, i.e., they even found the right constant factor.

How difficult is satisfiability of k-CNF formulas of bounded degree? Let (k, d)-SAT denote the problem of deciding whether a given k-CNF formula F of maximum degree $\Delta(F) \leq d$ is satisfiable. Clearly, $(k, f(k))$-SAT is trivial: All instances are satisfiable. Kratochvíl, Savický, and Tuza [4] showed that (k, d)-SAT exhibits a complexity jump: When the number of permitted occurrences per variable increases from $f(k)$ to $f(k)+1$, the complexity of the decision problem jumps from trivial (all instances are satisfiable) to NP-complete. It seems surprising that one can prove such a result without knowing the value of $f(k)$.

Other structural parameters exhibit complexity jumps, too. For a clause C in a CNF formula F, let $\Gamma_F(C)$ denote the clauses of F (excluding C) that have at least one variable in common with C, regardless of its sign. Let $\Gamma(F) := \max_{C \in F} |\Gamma_F(C)|$. Again by the Lovász Local Lemma, every k-CNF formula F with $\Gamma(F) \leq 2^k/e - 1$ is satisfiable. Gebauer, Moser, Welzl, and myself [9]

[1] Their graph has edges also between clauses with a trivial resolvent, for example $(x \vee y \vee \bar{z}), (u \vee \bar{y} \vee z)$, which is labeled as a trivial edge. Our graph is such the subgraph of all non-trivial edges of the graph of Ostrowski et al.

showed that there is some number $\ell(k)$ such that (1) all k-CNF formulas F with $\Gamma(F) \leq \ell(k)$ are satisfiable; (2) there exists an unsatisfiable k-CNF formula F with $\Gamma(F) \leq \ell(k)+1$; (3) satisfiability of k-CNF formulas F with $\Gamma(F) \geq \max(k+3, \ell(k)+2)$ is NP-hard. Note that $k+3 \leq \ell(k)+2$ for sufficiently large k. This means an "almost sudden" complexity jump, where in the case $\Gamma(F) = \ell(k)+1$ the decision problem is neither known to be in P nor to be NP-complete.

Define $\Gamma'_F(C)$ to be the number of clauses in F with which C has a *conflict*, that is those clauses D for which $u \in C$ and $\bar{u} \in D$ for some literal u. Let $\mathrm{lc}(F) := \max_{C \in F} |\Gamma'_F(C)|$. Here, lc stands for *local conflict*. The *lopsided* Lovász Local Lemma shows that every k-CNF formula F with $\mathrm{lc}(F) \leq 2^k/(ek) - 1$ is satisfiable. In [9] it was proven that this notion of conflict degree exhibits a sudden complexity jump: There is a function $\mathrm{lc}(k)$ such that (1) all k-CNF formulas F with $\mathrm{lc}(F) \leq \mathrm{lc}(k)$ are satisfiable; (2) deciding satisfiability of k-CNF formulas F with $\mathrm{lc}(F) \geq \mathrm{lc}(k)+1$ is NP-hard.

1.2 Our Contribution

Two clauses C, D have a *1-conflict* if there is exactly one literal u such that $u \in C$ and $\bar{u} \in D$. In other words, if C and D have a non-trivial resolvent. For example, the clauses $\{x, y, z\}$ and $\{\bar{x}, y\}$ have a 1-conflict, but $\{x, y, z\}$ and $\{\bar{x}, \bar{z}\}$ do not. We denote by $\Gamma^1_F(C)$ the set of clauses $D \in F$ such that C and D have a 1-conflict, and $\mathrm{lc}_1(F) := \max_{C \in F} |\Gamma^1_F(C)|$. In contrast to $\Delta(F)$, $\Gamma(F)$ and $\mathrm{lc}(F)$, it turns out that we completely understand the complexity of k-SAT when we restrict lc_1:

Theorem 1 (Complexity Jump). *The following three statements hold for all $k \geq 0$:*

1. *Every k-CNF formula F with $\mathrm{lc}_1(F) \leq k-1$ is satisfiable.*
2. *There exists an unsatisfiable k-CNF formula F with $\mathrm{lc}_1(F) = k$.*
3. *Satisfiability of k-CNF formulas F with $\mathrm{lc}_1(F) \leq k$ is in P.*
4. *Deciding satisfiability of k-CNF formulas F with $\mathrm{lc}_1(F) \leq k+1$ is NP-complete, if $k \geq 3$.*

Let us say a word about the proof of this theorem. Point 2 is very simple, we just provide a construction of a k-CNF formula for every $k \in \mathbb{N}$. Point 4, the hardness result, uses a reduction that is very similar to that of Kratochvíl, Savický, and Tuza [4] and Gebauer, Moser, Welzl, and myself [9]. Point 1 uses the concept of *blocked clauses* (Kullmann [10]). These are special clauses that are redundant and can be removed. The proof of Point 3 is the most interesting in our opinion. It consists of two main observations: (1) It is enough to decide satisfiability separately for each connected component of the 1-conflict graphs. (2) If the 1-conflict graph is connected, then splitting on a variable and iteratively deleting blocked clauses drastically reduces the size of the input formula. Blocked clause elimination (Järvisalo, Biere, and Heule [11]; Ostrowski, Grégoire, Mazure, and Sais [2]) is a known preprocessing step in SAT solvers and is quite useful in practice. In theory, however, eliminating blocked clauses can increase the resolution complexity of a formula exponentially: There are examples of formulas with short resolution proofs, but if one removes blocked clauses, every resolution proof of the remaining formula most be of exponential size; see for example Cook [12]. The class of formulas we discuss in Point 3 is thus *not* of this form: Blocked clause elimination is provably beneficial here.

Point 3 shows that there is a provable gap between the trivial and the NP-hard regime of the parameter lc_1. Such a gap is non-existent or not known to exist for the other parameters discussed above. We give a SAT algorithm that is correct in general, and in the special case of k-CNF formulas F with $\mathrm{lc}_1(F) \leq k$ runs in polynomial time. It is a branching algorithm and thus produces a treelike resolution refutation whose size is bounded by the number of recursive calls (this is a well-known fact; for a proof see [13], Theorem 3.2.5, page 35). Therefore, we get the following theorem:

Theorem 2 (Short Resolution Proofs). *If F is an unsatisfiable k-CNF formula and $\mathrm{lc}_1(F) = k$, then there is a treelike resolution refutation of F with at most $|F|$ leaves, where $|F|$ is the number of clauses in F.*

Theorem 3 (Finding the Satisfying Assignment). *Suppose F is a satisfiable k-CNF formula and $\mathrm{lc}_1(F) \leq k$. Then we can find a satisfying assignment in polynomial time.*

2 Notation

A CNF formula is a conjunction (AND) of clauses: $C_1 \wedge \cdots \wedge C_m$. A clause is a disjunction (OR) of literals: $x \vee \bar{y} \vee z$, where a literal is either a variable or its negation. We typically let n denote the number of variables in a formula, m the number of clauses, and k the size of its clauses: In a k-CNF formula, all clauses have size k. For notational purposes, we view formulas as set of clauses and clauses as sets of literals. So $\{\{x, y\}, \{\bar{x}, \bar{y}\}\}$ is the 2-CNF formula $(x \vee y) \wedge (\bar{x} \vee \bar{y})$ (which by the way is equivalent to $x \oplus y$). By $\mathrm{vbl}(C)$ and $\mathrm{vbl}(F)$ we denote the set of variables in a clause C or formula F, respectively. For a clause $D = \{u_1, \ldots, u_k\}$, we write $\bar{D} := \{\bar{u}_1, \ldots, \bar{u}_k\}$. This is not the negation of D. For a CNF formula F and a variable x, $F^{[x \mapsto 1]}$ is the CNF formula we obtain by replacing x by the constant 1. Thus, every clause containing x is satisfied (and can be removed from F), and every occurrence of $\bar{x}$ is unsatisfied and can be removed. We define $F^{[x \mapsto 0]}$ analogously.

2.1 Resolution

If C and D have a one-conflict, i.e., $C \cap \bar{D} = \{u\}$, we call the clause $E := (C \setminus \{u\}) \cup (D \setminus \bar{u})$ the resolvent of C and D. It is an easy exercise to show that the formulas $C \wedge D$ and $C \wedge D \wedge E$ are equivalent. Let F be a CNF formula. A *resolution derivation* from F is a sequence of clauses $C_1, C_2, \ldots, C_m$ where each C_i is (1) a clause of F or (2) the resolvent of two earlier clauses in the sequence. It is not difficult to see that F implies each clause in the sequence; that is, any assignment satisfying F satisfies $C_1, \ldots, C_m$. If $C_m = \Box$, i.e., the empty clause, which always evaluates to 0, we call $C_1, \ldots, C_m$ a resolution *refutation*, as it shows that F is unsatisfiable. A *treelike resolution derivation* from F is a binary tree T with the following properties: Every vertex u is labeled with a clause C_u; a leaf is labeled with a clause of F; if an inner vertex u has children v and w, then C_u is the resolvent of C_v and C_w. If the root is labeled with the empty clause $\Box$, we call it a treelike resolution refutation of F.

3 Proofs

We prove Point 2 of Theorem 1, which is the simplest of the four points. Take k variables and let F_k be the k-CNF formula containing all 2^k k-clauses over the k variables. F_k is unsatisfiable and $\mathrm{lc}_1(F_k) = k$. For an alternative construction, take $2k-1$ variables and let G_k consist of all $\binom{2k-1}{k}$ completely positive k-clauses and all $\binom{2k-1}{k}$ completely negative k-clauses. Again one checks that G_k is unsatisfiable and $\mathrm{lc}_1(G_k) = k$. For example, for $k = 2$ those two constructions yield

$$\{\{x, y\}, \{\bar{x}, y\}, \{x, \bar{y}\}, \{\bar{x}, \bar{y}\}\} \tag{1}$$

and

$$\{\{x, y\}, \{x, z\}, \{y, z\}, \{\bar{x}, \bar{y}\}, \{\bar{x}, \bar{z}\}, \{\bar{y}, \bar{z}\}\} \quad . \tag{2}$$

Their 1-conflict graphs are a C_4 and a C_6, respectively.

3.1 Basic Properties of the 1-Conflict Graph

Before we attack the remaining three points of the theorem, let us collect some interesting facts about 1-conflicts. Let us start with a simple but surprising observation, which probably is folklore.

Proposition 1. *Every CNF formula F with $\Box \notin F$ and $\mathrm{lc}_1(F) = 0$ is satisfiable.*

Note that without that proposition, the notion "1-conflict" would be misleading. After all, under any reasonable notion of conflict, a formula without conflicts should be satisfiable (extreme cases like $\Box \in F$ excluded). A direct consequence of the above proposition is that a hypergraph in which $|e \cap f| \neq 1$ for all hyperedges e, f is 2-colorable. This is a result of Lovász (Problem 13.33 in [14]).

Proof. A CNF formula F is unsatisfiable if and only if there is a resolution derivation of the empty clause (For a proof use induction over the number of variables or see for example [15], Theorem 4.2.1, page 26). Since F has no 1-conflicts, we cannot build any new resolvents. Since $\Box \notin F$, the formula is satisfiable. $\Box$

Lemma 1. *A CNF formula F is satisfiable if and only if every connect component of its 1-conflict graph is satisfiable. Furthermore, given satisfying assignments $\alpha_1, \ldots, \alpha_t$ for each of its t connected components, we can efficiently find a satisfying assignment α of F.*

Again, this is something we expect from a reasonable notion of conflict.

Proof. One direction is trivial: If F is satisfiable, then all connected components are satisfiable. For the other direction, write $F = F_1 \uplus F_2$ such that there is no 1-conflict between F_1 and F_2. By induction on the number of connected components, both F_1 and F_2 are satisfiable.

Choose a pair α_1, α_2 of assignments to $\text{vbl}(F)$ such that α_1 satisfies F_1, α_2 satisfies F_2, and the Hamming distance $d_H(\alpha_1, \alpha_2)$ is minimized. We claim that α_1 satisfies F_2 as well, and therefore F. Suppose for the sake of contradiction that this is not the case. There is a clause $D \in F_2$ such that α_1 does not satisfy D. Since α_2 satisfies D, there is a literal $u \in D$ such that $\alpha_1(u) = 0$ and $\alpha_2(u) = 1$. Define $\alpha_1' := \alpha[u \mapsto 1]$. Clearly $d_H(\alpha_1', \alpha_2) = d_H(\alpha_1, \alpha_2) - 1$. If we can prove that α_1' still satisfies F_1, we have arrived at a contradiction to d_H being minimal, and are done. Consider any $C \in F_1$. By the assumptions of the lemma, there is no 1-conflict between C and D. Hence either $C \cap \bar{D} = \emptyset$ or $|C \cap \bar{D}| \geq 2$. In the first case, $\alpha_1(C) = \alpha_1'(C) = 1$. In the second case, α_1 satisfies at least two literals in C, and therefore, α_1' satisfies at least one literal in C. This shows that α_1' indeed satisfies F_1, contradicting minimality of $d_H(\alpha_1, \alpha_2)$.

As for the algorithmic aspect, suppose we are given assignments α_1 and α_2 satisfying F_1 and F_2, respectively. As above, we we locally modify α_1, reducing the Hamming distance between to α_2, until we arrive at a single assignment α satisfying both F_1 and F_2. This takes only polynomial time. □

3.2 Blocked Literals and Blocked Clauses

It will pay off to introduce some notation. Let F be a CNF formula, C a clause, and $u \in C$ a literal. Define $\Gamma_F^1(C, u) := \{D \in F \mid C \cap \bar{D} = \{u\}\}$, that is, those clauses that have a 1-conflict with C, and this 1-conflict is generated by u. Note that

$$\Gamma_F^1(C) = \bigcup_{u \in C} \Gamma_F^1(C, u) ,$$

and this union is a disjoint one.

Definition 1 (Blocking Literals and Blocked Clauses, Kullmann [10]). *We say u* blocks *C* in *F if $\Gamma_F^1(C, u) = \emptyset$. A clause C is* blocked *in F if some $u \in C$ blocks C in F.*

If the ambient formula is understood, we simply say that u blocks C and C is blocked, not explicitly referring to F. Blocked clauses are redundant, in some way:

Proposition 2 (Kullmann [10]). *Let F be a CNF formula and $C \in F$ some clause. If C is blocked in F, then F is satisfiable if and only if $F \setminus \{C\}$ is. Furthermore, given a satisfying assignment α of $F \setminus \{C\}$, we can efficiently find a satisfying assignment α' of F.*

We can use Proposition 2 to repeatedly remove blocked clauses in a formula, finally arriving at a formula without blocked clauses, which we denote by $\texttt{deleteBlocked}(F)$.

Proposition 3 (Kullmann [10]). *Let F be a CNF formula and let $F' := \texttt{deleteBlocked}(F)$. Then F is satisfiable if and only if F' is, and given a satisfying assignment α' of F', we can efficiently construct a satisfying assignment α of F.*

Proof. This follows from Proposition 2 and induction on the number of clauses.□

Proposition 3 yields another proof of Proposition 1: If $\mathrm{lc}_1(F) = 0$, then every non-empty clause is blocked by one of its literals. The algorithm `deleteBlocked` will remove one by one, finally arriving at the empty formula, which is satisfiable. Here we were using an innocent but crucial fact: If a clause C is blocked with respect to F, then it is also blocked with respect to every subformula $F' \subseteq F$ for which $C \in F'$.

This proves Point 1 of the theorem: If F is a k-CNF formula and $\mathrm{lc}_1(F) \leq k-1$, then every clause contains at least one literal that blocks it. Thus $\texttt{deleteBlocked}(F) = \{\}$, the empty formula, thus it is satisfiable.

3.3 Simple and Tight Formulas

Definition 2. *A CNF formula F is* simple *if $|\Gamma_F(C, u)| \leq 1$ for every $C \in F$ and every $u \in C$. It is* tight *if $|\Gamma_F(C, u)| = 1$ for every $C \in F$ and every $u \in C$.*

For example, the following formula is tight and satisfiable.

$$\{\{\bar{x}_1, x_2\}, \{\bar{x}_2, x_3\}, \dots, \{\bar{x}_{n-1}, x_n\}, \{\bar{x}_n, x_1\}\}$$

As another example, the formulas in (1) and (2) are tight and unsatisfiable.

Proposition 4. *Suppose F is simple. Then $\texttt{deleteBlocked}(F)$ is tight. Suppose F is a k-CNF formula and $\mathrm{lc}_1(F) \leq k$. Then $\texttt{deleteBlocked}(F)$ is tight.*

Proof. Suppose F is simple. Then any subformula $F' \subseteq F$ is simple, too. Thus $F' := \texttt{deleteBlocked}(F) \subseteq F$ is simple. It contains no blocked clauses, so $|\Gamma_{F'}(C, u)| \geq 1$ for all $u \in C \in F'$. But $|\Gamma_{F'}(C, u)| \leq |\Gamma_F(C, u)| \leq 1$, which means they must be exactly 1. In other words, F' is tight.

For the second statement, suppose F is a k-CNF formula and $\mathrm{lc}_1(F) \leq k$. Then this statement is true for $F' := \texttt{deleteBlocked}(F)$, too. Since F' contains no blocked clause, $|\Gamma_{F'}(C, u)| \geq 1$ for all $u \in C \in F'$. Thus $k \leq \sum_{u \in C} |\Gamma_{F'}(C, u)| = |\Gamma_{F'}(C)| \leq \mathrm{lc}_1(F') = k$, so equality holds throughout, meaning $|\Gamma_{F'}(C, u)| = 1$, and F' is tight. □

Proposition 5. *Suppose F is simple, and x is a variable. Then $F^{[x\mapsto 1]}$ is simple, and so is $F^{[x\mapsto 0]}$.*

Proof. Suppose $F' := F^{[x\mapsto 0]}$ is not simple. We will show that F is not simple. By assumption on F', there is a clause $C' \in F'$ and a literal $u \in C'$ such that $|\Gamma_{F'}(C', u)| \geq 2$. This means there are clauses D_1', D_2' such that $C' \cap \bar{D}_1' = C' \cap \bar{D}_2' = \{u\}$. Since $F' = F^{[x\mapsto 0]}$, this means that F contains clauses C, D_1, D_2 such that either $C = C'$ or $C = C' \vee x$; either $D_1 = D_1'$ or $D_1 = D' \vee x$; either $D_2 = D_2'$ or $D_2 = D' \vee x$. None of those clauses contains $\bar{x}$, though. Therefore $C \cap \bar{D}_1 = C' \cap \bar{D}_1' = \{u\}$, and similarly $C \cap \bar{D}_2 = C' \cap \bar{D}_2 = \{u\}$. Thus, $D_1, D_2 \in \Gamma_F^1(C, u)$, and F is not simple, either. $\square$

3.4 An Efficient Algorithm

We will now use the above notions of blocked clauses and simple and tight formulas to prove the main result of this paper, i.e., Point 3 of Theorem 1. We give an algorithm that efficiently decides satisfiability of k-CNF formulas F with $\mathrm{lc}_1(F) \leq k$. See Algorithm `simpleSAT` below. To see the correctness `simpleSAT`,

Algorithm 1.1. `simpleSAT`(CNF formula F)

1: $F \leftarrow$ `deleteBlocked`(F)
2: **if** $\square \in F$ **then**
3:    **return** `false`
4: **else if** $F = \{\}$ **then**
5:    **return** `true`
6: **else if** $F = F_1 \uplus F_2$ for some $F_1, F_2 \neq \{\}$ and $|C \cap \bar{D}| \neq 1$ for all $C \in F_1, D \in F_2$ **then**
7:    **return** `simpleSAT`(F_1) $\wedge$ `simpleSAT`(F_2)
8: **else**
9:    $x \leftarrow \mathrm{vbl}(F)$
10:    $G_1 :=$ `deleteBlocked`$(F^{[x\mapsto 1]})$
11:    $G_0 :=$ `deleteBlocked`$(F^{[x\mapsto 0]})$
12:    **return** `simpleSAT`(G_1) $\vee$ `simpleSAT`(G_0)
13: **end if**

consider lines 1.1 and 1.1. The algorithm recurses on F_1 and F_2 and returns `true` if both calls return `true`. By Lemma 1, F is satisfiable if and only if F_1 and F_2 are both satisfiable individually. The challenging part is to argue that its running time is polynomial in our case.

Lemma 2. *If F is simple, then* `simpleSAT`(F) *runs in polynomial time. More precisely, let m be the number of clauses in F. The total number of calls to* `simpleSAT`(F) *during its execution is $2m - 1$ if $m \geq 1$ and 1 otherwise.*

Proof. If $m = 0$, then $F = \{\}$ and the algorithm just returns `true`. So the claim holds for $m = 0$. After the first line, F is tight, which follows from Proposition 4.

If $m = 1$, then $F = \{\Box\}$ or $F = \{\}$ after the first line, so there is no further recursive call either. So the claim holds for $m = 1$, too.

Otherwise, suppose $m \geq 2$, i.e., F has at least two clauses. Then `simpleSAT` either recurses on two subformulas F_1, F_2 (line 1.1) or on G_0, G_1 (line 1.1). Suppose `simpleSAT` recurses on F_1 and F_2. Note that both F_1 and F_2 have at least one clause. We apply induction to F_1 and F_2 and see that the total number of calls is at most $1 + (2|F_1| - 1) + (2|F_2| - 1) = 2(|F_1| + |F_2|) - 1 = 2m - 1$. If `simpleSAT` recurses on G_0 and G_1, things are more complicated. This is the only point where we need that F is tight:

Proposition 6. *Suppose F is tight, $x \in \mathrm{vbl}(F)$, and let $G_0 :=$ `deleteBlocked`$(F^{[x\mapsto 0]})$ and $G_1 :=$ `deleteBlocked`$(F^{[x\mapsto 1]})$. Then $|G_0| + |G_1| \leq |F|$.*

With this proposition, we apply induction to G_0 and G_1. If both G_0 and G_1 contain at least one clause, then the total number of calls is $1 + (2|G_0| - 1) + (2|G_1| - 1) \leq 2|F| - 1$.

At this point we are almost done, but have to deal with the annoying special case that G_0 or G_1 might be empty. If G_0 contains no clause but G_1 does, then we apply induction on G_1 and see that the number of calls is $1+1+(2|G_1|-1) = 2|G_1| + 1 \leq 2|F| - 1$, since $|G_1| < |F|$. If $G_0 = G_1 = \{\}$, then there is a total of 3 calls. Since F has $m \geq 2$ clauses, this completes the proof of the lemma. $\Box$.

Proof (of Proposition 6). Let $C \in F$ be a clause. We argue that C may make its way into either G_0 or G_1, but not both. Thus $|G_0| + |G_1| \leq F$.

There are two cases: Suppose $x \in C$ or $\bar{x} \in C$, without loss of generality $x \in C$. Then setting $x \mapsto 1$ satisfies C, and C does not make it into G_1, but $C^{[x\mapsto 0]}$ may make it into G_0 (provided it survives `deleteBlocked`).

So suppose $x \notin C$ and $\bar{x} \notin C$. After line 1.1, the 1-conflict graph of F is connected. So there is a path $C = C_1, C_2, \ldots, C_{t-1}, C_t$ such that $x \in \mathrm{vbl}(C_t)$ but $x \notin \mathrm{vbl}(C_i)$ for $1 \leq i \leq t - 1$. Without loss of generality, $x \in C_t$. After setting $x \mapsto 1$, the clause C_t is satisfied, and C_{t-1} has one 1-conflict neighbor less. So now C_{t-1} is blocked, and `deleteBlocked`$(F^{[x\mapsto 1]})$ deletes it. Thus C_{t-2} loses a neighbor and becomes blocked, and so on, until finally $C = C_1$ will be removed. See Figure 1 for an illustration. Thus, C does not make its way into G_1. This proves the proposition. $\Box$

Resolution Size – Proof of Theorem 2

The algorithm `simpleSAT` is a branching algorithm, and it is a well-known fact that branching algorithms implicitly produce a treelike resolution refutation when run on an unsatisfiable formula (see e.g. [13], Theorem 3.2.5, page 35). The number of clauses in the refutation is at most the number of recursive calls. Since an unsatisfiable formula has $m \geq 1$ clauses, the number of calls is at most $2m - 1$, by Lemma 2. Thus the resolution tree has at most $2m - 1$ nodes, and therefore at most m leaves. This proves Theorem 2.

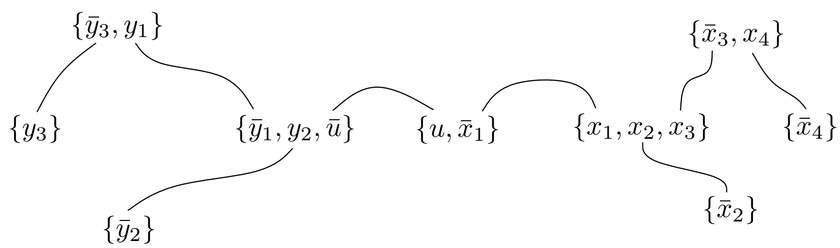

Fig. 1. Illustration of Proposition 6. When we set u to 1, the clause $\{u, \bar{x}_1\}$ disappears. The clause $\{x_1, x_2, x_3\}$ has only one outgoing edge labeled x_1. Once $\{u, \bar{x}_1\}$ disappears, the literal x_1 will block $\{x_1, x_2, x_3\}$, and $\{x_1, x_2, x_3\}$ will be deleted, too. Then $\bar{x}_2$ will block $\{\bar{x}_2\}$ and $\bar{x}_3$ will block $\{\bar{x}_3, x_4\}$, thus these clauses are also deleted, and so on.

Finding the Satisfying Assignment – Proof of Theorem 3

Suppose F is a satisfiable k-CNF formula and $\mathrm{lc}_1(F) \leq k$. We construct a satisfying assignment F by tracking the execution of `simpleSAT`(F). Denote by $F' :=$ `deleteBlocked`(F) the input formula after the first line. By Proposition 4, F' is tight. If we can efficiently find a satisfying of F', then by Proposition 3 we can efficiently find a satisfying assignment of F. If `simpleSAT` recurses in line 1.1 on F_1 and F_2, we assume by induction that we know satisfying assignments α_1 of F_1 and α_2 of F_2. By Lemma 1 we can efficiently combine α_1, α_2 into a single α satisfying of F'. If `simpleSAT` recurses in line 1.1 on G_0 and G_1, suppose without loss of generality that G_1 is satisfiable and let α be a satisfying assignment. Since $G_1 =$ `deleteBlocked`$(F'^{[x \mapsto 1]})$, we can efficiently find a satisfying assignment α' of $F'^{[x \mapsto 1]}$, by Proposition 3. Thus, $\alpha' \cup [x \mapsto 1]$ satisfies F'. Summing up, we can construct a satisfying assignment of F by adding some bookkeeping to `simpleSAT`.

4 Hardness for $\mathrm{lc}_1 \geq k+1$: Proof Sketch

We sketch a reduction from k-SAT to k-SAT with $\mathrm{lc}(F) \leq k+1$, but refer the reader to the appendix for the full details. Let F be a k-CNF formula and let $\deg_F(x)$ denote the number of clauses in F in which x occurs, regardless of its sign. In a first step, we introduce $2\deg_F(x)$ new variables $x_1, x_2, \ldots, x_{2\deg_F(x)}$ for each $x \in \mathrm{vbl}(F)$ and replace the i^{th} occurrence of x by x_{2i}.

In a second step, we add an *equalizer formula* $\mathrm{Eq}(x_1, \ldots, x_{2\deg_F(x)})$ for each $x \in \mathrm{vbl}(F)$. This is a 2-CNF formula which is satisfied if and only if its $2\deg_F(x)$ variables receive the same truth value. The resulting formula is satisfiable if and only if F is. However, it is not a k-CNF formula, because it contains 2-clauses.

In a third step, we "fill up" each 2-clause $\{u, v\}$ to a k-clause, by adding $k-2$ new variables. This is, we replace $\{u, v\}$ by $\{u, v, w_3, \ldots, w_k\}$. Finally, we add a "forcer" for every new variable w_i introduced in the third step. A forcer is a k-CNF formula that is satisfiable, but only if w_i is set to 0. Such a forcer can be built in a rather straightforward manner from an unsatisfiable k-CNF formula G with $\mathrm{lc}_1(G) = k$.

Acknowledgments

I am very thankful to Heidi Gebauer, Robin Moser, and Emo Welzl. The question posed and answered in this paper arose during our work on [9]. Furthermore, I am grateful to Navid Talebanfard, who pointed out Exercise 13.33 from [14].

References

1. Allender, E., Chen, S., Lou, T., Papakonstantinou, P., Tang, B.: Width-parameterized SAT: time-space tradeoffs. ECCC TR12-027 (2012)
2. Ostrowski, R., Grégoire, É., Mazure, B., Sais, L.: Recovering and exploiting structural knowledge from CNF formulas. In Hentenryck, P.V., ed.: CP. Volume 2470 of Lecture Notes in Computer Science., Springer (2002) 185–199
3. Erdős, P., Lovász, L.: Problems and results on 3-chromatic hypergraphs and some related questions. In Hajnal, A., Rado, R., Sós, V.T., eds.: Infinite and Finite Sets (to Paul Erdős on his 60th birthday), Vol. II. North-Holland (1975) 609–627
4. Kratochvíl, J., Savický, P., Tuza, Z.: One more occurrence of variables makes satisfiability jump from trivial to NP-complete. SIAM Journal of Computing **22**(1) (1993) 203–210
5. Savický, P., Sgall, J.: DNF tautologies with a limited number of occurrences of every variable. Theoret. Comput. Sci. **238**(1–2) (2000) 495–498
6. Hoory, S., Szeider, S.: A note on unsatisfiable k-CNF formulas with few occurrences per variable. SIAM Journal on Discrete Mathematics **20**(2) (2006) 523–528
7. Gebauer, H.: Disproof of the neighborhood conjecture with implications to SAT. In Fiat, A., Sanders, P., eds.: 17th Annual European Symposium on Algorithms (ESA 2009). Volume 5757 of Lecture Notes in Computer Science., Springer (2009) 764–775
8. Gebauer, H., Szabó, T., Tardos, G.: The local lemma is tight for SAT. In Randall, D., ed.: SODA, SIAM (2011) 664–674
9. Gebauer, H., Moser, R.A., Scheder, D., Welzl, E.: The Lovász Local Lemma and satisfiability. In Albers, S., Alt, H., Näher, S., eds.: Efficient Algorithms. Volume 5760 of Lecture Notes in Computer Science., Springer (2009) 30–54
10. Kullmann, O.: On a generalization of extended resolution. Discrete Applied Mathematics **96-97** (1999) 149–176
11. Järvisalo, M., Biere, A., Heule, M.: Blocked clause elimination. In Esparza, J., Majumdar, R., eds.: TACAS. Volume 6015 of Lecture Notes in Computer Science., Springer (2010) 129–144
12. Cook, S.A.: A short proof of the pigeon hole principle using extended resolution. SIGACT News **8**(4) (October 1976) 28–32
13. Hoffmann, J.: Resolution proofs and DLL algorithms with clause learning (2007) Diploma Thesis, Ludwig-Maximilians-Universität München.
14. Lovász, L.: Combinatorial problems and exercises (2. ed.). North-Holland (1993)
15. Krajicek, J.: Bounded Arithmetic, Propositional Logic and Complexity Theory. Encyclopedia of Mathematics and its Applications. Cambridge University Press (1995)

A Hardness for $\mathrm{lc}_1 \geq k+1$: Full Proof

Finally, we show point 4 of Theorem 1. The proof is similar to the proof by Gebauer et al. [9] that the number of local conflicts exhibits a complexity jump. We will give a reduction that takes a k-CNF formula F as input and outputs a k-CNF formula F', such that F is satisfiable if and only if F' is, and $\mathrm{lc}_1(F') \leq k+1$, i.e., every clause in F' has a 1-conflict with at most $k+1$ other clauses. In fact, F' will have a stronger property: For every clause $C = \{u_1, \ldots, u_k\} \in F'$, $k-1$ of its literals generate at most one 1-conflict, say $|\Gamma^1_{F'}(C, u_i)| = 1$ for $1 \leq i \leq k-1$, and one literal is may have up to two 1-conflicts: $|\Gamma^1_{F'}(C, u_k)| \leq 2$. For $k \geq 3$ this shows that deciding satisfiability of k-CNF formulas with $\mathrm{lc}_1(F) \leq k+1$ is NP-complete.

For a variable x, denote by $\deg_F(x)$ the number of clauses of F in which x occurs, regardless of its sign. In a first step, for each variable $x \in \mathrm{vbl}(F)$ we set $d := \deg_F(x)$ and introduce $2d$ new variables $x_1, x_2, \ldots, x_{2d}$. We replace the d occurrences of x by the variables $x_2, x_4, \ldots, x_{2d}$. Skipping the odd indices will prove useful soon. We call the new formula F_2. For example, if x appears in three clauses, say

$$F = \{\{x, \bar{y}, \bar{z}\}, \{x, u, v\}, \{\bar{x}, y, \bar{u}\}, \ldots\},$$

then we replace those three occurrences by x_2, x_4, and x_6 and obtain

$$\{\{x_2, \bar{y}, \bar{z}\}, \{x_4, u, v\}, \{\bar{x_6}, y, \bar{u}\}, \ldots\}.$$

We apply the same procedure to y, z, and all other variables. F_2 has no conflicts, since each variable appears in only one clause. It is satisfiable, which is not good, because we want a formula F' such that F is satisfiable if and only if F' is. We introduce an *equalizer* formula for the variables $x_1, x_2, \ldots, x_{2d}$. This is a formula which is satisfied if and only if one assigns the same value to $x_1, x_2, \ldots, x_{2d}$:

$$\mathrm{Eq}(x_1, \ldots, x_{2d}) = \{\{\bar{x}_1, x_2\}, \{\bar{x}_2, x_3\}, \ldots, \{\bar{x}_{2d-1}, x_{2d}\}, \{\bar{x}_{2d}, x_1\}\} .$$

$\mathrm{Eq}(x_1, \ldots, x_{2d})$ has exactly two satisfying assignments: All-1 and All-0. We obtain F_3 by adding an equalizer for every variable $x \in \mathrm{vbl}(F)$:

$$F_3 := F_2 \cup \bigcup_{x \in \mathrm{vbl}(F)} \mathrm{Eq}(x_1, x_2, \ldots, x_{2\deg_F(x)}) .$$

The property of the equalizers implies F is satisfiable if and only if F_3 is. Furthermore, $|\Gamma^1_{F_3}(C)| \leq |C| + 1$ for every $C \in F_3$: Since each occurrence of a variable in F gets replaced by a fresh copy of this variable, there are no conflicts within F_2. Every clause $C \in F_2$ has a 1-conflict with exactly $|C|$ clauses in the equalizer formulas: If $x_i \in C$, then C has a 1-conflict with the clause $\{\bar{x}_i, x_{i+1}\} \in \mathrm{Eq}(x_1, \ldots, x_{\deg_F(x)})$. Similarly, if $\bar{x}_i \in C$, then C has a 1-conflict with $\{\bar{x}_{i-1}, x_i\}$. Therefore, C is simple. Consider a clause $\{\bar{x}_i, x_{i+1}\}$ in an equalizer. Each of the two literals generates one 1-conflict with another equalizer-clause. Additionally, $\bar{x}_i$ (if i is even) or x_{i+1} (if i is odd) might generate a 1-conflict with a clause in F_2. Thus, $|\Gamma^1_{F_3}(C)| \leq |C|$ if $C \in F_2$, and $|\Gamma^1_{F_3}(C)| \leq |C| + 1$ if C is an equalizer-clause.

The formula F_3 fulfills almost all our needs, except that its clauses are too short: We want to output a k-CNF formula. For this reason, we add $k-2$ new variables to each equalizer clause: We replace $\{\bar{x}_i, x_{i+1}\}$ by

$$\{\bar{x}_i, x_{i+1}, u_3, \ldots, u_k\} .$$

We add clauses that force the variables $u_3, \ldots, u_k$ to 0: For each u_j, we construct a formula that is satisfiable if and only if u_j is set to 0. Let $v_1, \ldots, v_k$ be new variables and let $CF(v_1, \ldots, v_k)$ denote the k-CNF formula consisting of all 2^k k-clauses over $v_1, \ldots, v_k$. This formula is unsatisfiable and tight. Pick one clause from this formula, say $\{v_1, \ldots, v_k\}$, and replace it by $\{v_1, \ldots, v_{k-1}, \bar{u}_j\}$. We call this k-CNF formula $G(u_j)$. It is satisfiable, but every satisfying assignment sets u_j to 0. Furthermore, $G(u_j)$ is simple, and $\bar{u}_j$ blocks $\{v_1, \ldots, v_{k-1}, \bar{u}_j\}$ in $G(u_j)$. We denote by F' the k-CNF formula we obtain from F_3 by filling up the equalizer-clauses and adding the formulas $G(u_j)$ to it. By construction F' is satisfiable if and only if F is.

Let us summarize the construction of F'. For every $x \in \mathrm{vbl}(F)$, we add $2\deg_F(x)$ equalizer clauses, each of which we fill up to a k-clause, introducing a total of $(k-2)2\deg_F(x)$ new variables. Finally, we add the formulas $G(u_j)$, consisting of 2^k clauses. That is, for each $x \in \mathrm{vbl}(F)$, we add $2\deg_F(x) + 2(k-2)\deg_F(x)2^k$ clauses. This increases the total size of F by a constant factor.

Let us verify that $\mathrm{lc}_1(F') \leq k+1$. There are three types of clauses in F': First, there are the "original" clauses, those of F_2. These clauses have at most k 1-conflict neighbors in F_3 and also in F'. Second, there are equalizer clauses $\{\bar{x}_i, x_{i+1}, u_3, \ldots, u_k\}$. Here, every literal causes at most one 1-conflict, except possibly x_i (if i is even) or x_{i+1} (if i is odd), which may cause up to two 1-conflicts. Thus this clause has at most $k+1$ many 1-conflicts. Third, there are clauses in $G(u_j)$. Every clause $C \in G(u_j)$ has at most k 1-conflict neighbors in $G(u_j)$, and the literal $\bar{u}_j$ blocks $\{v_1, \ldots, v_{k-1}, \bar{u}_j\}$ with respect to $G(u_j)$. Since u_j occurs in exactly one equalizer clause, this adds exactly one 1-conflict to $\{v_1, \ldots, v_{k-1}, \bar{u}_j\}$. Therefore every $C \in G(u_j)$ has at most k 1-conflict neighbors in F'. This concludes the proof.