Small Unsatisfiable k-CNFs with Bounded Literal Occurrence

Tianwei Zhang
Algorithms and Complexity Group, TU Wien, Austria

Tomáš Peitl
Algorithms and Complexity Group, TU Wien, Austria

Stefan Szeider
Algorithms and Complexity Group, TU Wien, Austria

Abstract

We obtain the smallest unsatisfiable formulas in subclasses of k-CNF (exactly k distinct literals per clause) with bounded variable or literal occurrences. Smaller unsatisfiable formulas of this type translate into stronger inapproximability results for MaxSAT in the considered formula class. Our results cover subclasses of 3-CNF and 4-CNF; in all subclasses of 3-CNF we considered we were able to determine the smallest size of an unsatisfiable formula; in the case of 4-CNF with at most 5 occurrences per variable we decreased the size of the smallest known unsatisfiable formula. Our methods combine theoretical arguments and symmetry-breaking exhaustive search based on SAT Modulo Symmetries (SMS), a recent framework for isomorph-free SAT-based graph generation. To this end, and as a standalone result of independent interest, we show how to encode formulas as graphs efficiently for SMS.

2012 ACM Subject Classification Mathematics of computing → Solvers; Mathematics of computing → Graph enumeration; Theory of computation → Automated reasoning; Hardware → Theorem proving and SAT solving

Keywords and phrases k-CNF, (k, s)-SAT, minimally unsatisfiable formulas, symmetry breaking

Digital Object Identifier 10.4230/LIPIcs.SAT.2024.31

Related Version *Full Version*: `https://arxiv.org/abs/2405.16149` [28]

Supplementary Material *Software*: `https://doi.org/10.5281/zenodo.11282310`

Funding The project leading to this publication has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101034440 , and was supported by the Austrian Science Fund (FWF) within the project 10.55776/P36688.

1 Introduction

A (k, s)-formula is a propositional CNF formula in which each clause has exactly k distinct literals and each variable occurs (positively or negatively) in at most s clauses. Since Tovey [27] initiated the study of (k, s)-CNF formulas in 1984, they have been the subject of intensive investigation [3, 4, 7, 9, 10, 11, 12, 15, 21, 25]. Using Hall's Marriage Theorem, Tovey showed that all $(3, 3)$-CNF formulas are satisfiable, but allowing a fourth occurrence per variable yields a class of formulas for which the satisfiability problem is NP-complete. Kratochvíl, et al. [21] generalized this result and showed that for each $k \geq 3$, there exists a threshold $s = f(k)$ such that all $(k, f(k))$-formulas are satisfiable and checking the satisfiability of $(k, f(k) + 1)$-formulas (the (k, s)-*SAT problem*) is NP-complete. Therefore, determining whether (k, s)-SAT is NP-hard boils down to identifying an unsatisfiable (k, s)-formula. While tight asymptotic bounds for the threshold have been obtained [10], exact values are

only known for $k \leq 4$ [11]: $f(3) = 3$, $f(4) = 4$; $f(5) \in [5, 7]$. No decision procedure is known for determining the threshold $f(k)$, since no upper bound on the size of a smallest unsatisfiable (k, s)-formula is known for $k > 4$.

It is an intriguing question of extremal combinatorics to determine the size of the smallest unsatisfiable (k, s)-formulas (with $s > f(k)$), and this paper sets out to address this question for various values of the parameters. This requires proving lower and upper bounds – an upper bound typically consists in exhibiting a formula with suitable parameters, while a lower bound requires a proof that no such formulas of a particular size exist. For $k = 3$, the unsatisfiable $(3, 4)$-formula constructed by Tovey [27] has 40 clauses, this was later improved to 19 and then 16 by Berman, et al. [3, 4].

A simple counting argument shows that such a formula must contain at least 8 clauses, hence there is a significant gap between the known lower and upper bounds. For $k = 4$, the gap is even larger. Stříbrná [26] constructed an unsatisfiable (4,5)-CNF formula with 449 clauses, which Knuth [20, p. 588] improved to 257. The same counting argument shows that such a formula must contain at least 16 clauses.

All constructions above have the following in common. A satisfiable (k, s)-formula with a backbone variable x that must be false in all satisfying truth assignments is first constructed. Such a formula is called a (k, s)-*enforcer*. Then one combines k copies of this formula together with a k-clause with the k backbone variables to obtain an unsatisfiable (k, s)-CNF formula. Aside from providing an upper bound for the size of a smallest unsatisfiable (k, s)-formula, the size of unsatisfiable (k, s)-enforcers has a direct effect on inapproximability results for certain NP-hard Max-SAT problems [3]. We also address the problem of the smallest size of a $(3, s)$ or $(3, p, q)$-enforcer in this paper.

Contribution

In this paper, we develop a general approach to computing small unsatisfiable (k, s)-formulas and (k, s)-enforcers. We also consider the more fine-grained setting of (k, p, q)-formulas, where p and q bound the number of positive and negative occurrences per variable, respectively. We observe a similar threshold phenomenon in the complexity of (k, p, q)-SAT as in the case of (k, s)-SAT (Lemma 4).

Our approach rests on utilizing the SAT Modulo Symmetries (SMS) framework [18, 16] for isomorph-free generation of unsatisfiable (k, s) and (k, p, q)-formulas for a fixed number n of variables and number m of clauses with parameters k, s, p, q.

The basic setting without any techniques for speed-up and divide-and-conquer scales up to about $m = 13$. We then use further theoretical arguments together with techniques for speed-up to determine the size of smallest formulas in various setting.

Smallest unsatisfiable formulas. We determined the size of smallest unsatisfiable $(3, s)$ and $(3, p, q)$-formulas for all possible s, p and q the values are given in Table 1 on the left for $(3, s)$-formulas and on the right for $(3, p, q)$-formulas.

In particular, we identified a smallest unsatisfiable $(3, 1, 3)$-formula with 22 clauses, which implies that $(3, 1, 3)$-SAT is NP-complete. Hence, we now have a direct and streamlined proof for the dichotomy of $(3, p, q)$-SAT (Theorem 5).

Smallest enforcers. Table 2 summarizes our results on the size of smallest $(3, s)$-enforcers and $(3, p, q)$-enforcers.

Table 1 Size of smallest unsatisfiable $(3, s)$-formulas (left) and size of smallest unsatisfiable $(3, p, q)$-formulas (right). ∞ indicates that for these parameters no unsatisfiable formula exists.

$s \leq 3$	$s = 4$	$s = 5$	$s \geq 6$
∞	16	11	8

	$q = 1$	$q = 2$	$q = 3$	$q = 4$	$q \geq 5$
$p = 1$	∞	∞	22	19	16
$p = 2$	–	20	11	10	10
$p \geq 3$	–	–	8	8	8

Table 2 Size of smallest $(3, s)$-enforcers (left) and smallest $(3, p, q)$-enforcers (right).

$s \leq 3$	$s = 4$	$s \geq 5$
∞	5	4

	$q = 1$	$q = 2$	$q = 3$	$q = 4$	$q \geq 5$
$p = 1$	∞	∞	7	6	5
$p = 2$	–	10	5	5	5
$p \geq 3$	–	–	5	5	5

A smaller unsatisfiable $(4, 5)$-formula. Recall that $f(4) = 5$. The smallest known unsatisfiable $(4, 5)$-formula is due to Knuth and has 257 clauses. We improve this by exhibiting an unsatisfiable $(4, 5)$-formula with 235 clauses. We obtain this by first computing an auxiliary formula with SMS and then constructing from it an unsatisfiable $(4, 5)$-formula by disjunctive splitting.

Structure of the paper. After this introduction, and preliminaries, the paper is organized into two main parts. In Section 3, we lay out the principle encoding that provides the basis for all our results. In Section 4, we dive into the more complicated, technical aspects that are necessary to rule out the existence of larger formulas and obtain better lower bounds. Lemmas and Theorems marked with $\star$ have proofs in the full version [28], where some arguments have been streamlined.

2 Preliminaries

For positive integers $k < \ell$, we write $[k] = \{1, 2, \ldots, k\}$, $[-k] = \{-k, -(k-1), \ldots, -1\}$, and $[k, \ell] = \{k, \ldots, \ell\}$. We assume familiarity with fundamental notions of propositional logic [19]. In this paper we will talk about *(minimally) unsatisfiable* propositional formulas represented as graphs, specified by properties expressed as *quantified Boolean formulas*, and about *isomorphisms (symmetries)* of these formulas. We review the relevant basics below.

CNF formulas. A *literal* is a (propositional) variable x or a negated variable $\overline{x}$, whereby $\overline{\overline{x}} = x$. We write $\mathsf{var}(x) := \mathsf{var}(\overline{x}) := x$ for the variable belonging to a literal. A set S of literals is *tautological* if $S \cap \overline{S} \neq \emptyset$, where $\overline{S} = \{\, \overline{x} : x \in S \,\}$. A *clause* is a finite non-tautological set of literals. A k-clause is a clause that contains exactly k literals. The 0-clause is denoted by $\square$. A *(CNF) formula* is a finite set of clauses. For $k \geq 1$, a *k-CNF formula* is a formula in which all clauses are k-clauses (please note that some authors allow a k-CNF formula to contain clauses with fewer than k literals, but it is significant in our context that the number is exactly k) and a *$(\leq k)$-CNF formula* is a formula in which all clauses contain at most k literals. A variable x *occurs positively* in a clause C if $x \in C$, it *occurs negatively* in C if $\overline{x} \in C$, and it *occurs* in C if it occurs in C positively or negatively. For a literal x, $F[x]$ denotes the set of clauses in F in which $\mathsf{var}(x)$ occurs. We will often write a $\leq k$-CNF with m

clauses as a $k \times m$ matrix whose columns are the clauses, and entries are literal occurrences. When a clause has $r < k$ literals, we write $\times$ in the last $k - r$ rows in the corresponding column of the matrix.

Counting occurrences. For a clause C, we write $\mathsf{var}(C)$ for the set of variables that occur in C, and for a CNF formula F we write $\mathsf{var}(F) = \bigcup_{C \in F} \mathsf{var}(C)$. The *degree* of a variable in a formula F is defined as $\deg_F^{\mathsf{var}}(x) := |F[x]|$. For literals, we put $\deg_F^{\mathsf{var}}(\overline{x}) := \deg_F^{\mathsf{var}}(x)$. The *degree* of a literal x, denoted by $\deg_F(x)$, is the number of clauses in which the literal occurs. We may omit the subscript F when it is clear from the context.

For $k, s \geq 1$, a (k, s)-*formula* is a k-formula in which each variable occurs in at most s clauses, and a $(\leq k, s)$-*formula* is a $(\leq k)$-formula in which each variable occurs in at most s clauses. For $k, p, q \geq 1$, a (k, p, q)-*formula* is a k-formula in which each variable occurs in at most p clauses positively and in at most q clauses negatively, and a $(\leq k, p, q)$-*formula* is a $(\leq k)$-formula with the same constraint. Without loss of generality, we will assume $p \leq q$ for (k, p, q)-formulas as we can always swap positive and negative literals.

We define $\mu(k, s)$ to be the number of clauses of a smallest unsatisfiable (k, s)-formula, and $\mu(k, p, q)$ denotes that of the smallest unsatisfiable (k, p, q)-formula.

Bounded literal occurrence SAT. A *truth assignment* for a set X of variables is a mapping $\tau : X \to \{0, 1\}$. In order to define τ on literals, we set $\tau(\overline{x}) = 1 - \tau(x)$. A truth assignment τ *satisfies* a clause C if C contains at least one literal x with $\tau(x) = 1$, and τ *satisfies* a formula F if it satisfies every clause of F. In the latter case, we call F *satisfiable*. The Satisfiability problem (SAT) is to decide whether a given formula is satisfiable. (k, s)-SAT is SAT restricted to (k, s)-formulas, and (k, p, q)-SAT is SAT restricted to (k, p, q)-formulas.

Enforcers. A (k, s)-*enforcer* is a satisfiable (k, s)-formula F with a variable x with $\deg_F^{\mathsf{var}}(x) < s$ that is set to the same value in every satisfying assignment. A (k, p, q)-*enforcer* is a satisfiable (k, p, q) formula F with a variable x which is either set to true in every satisfying assignment, and then $\deg_F(x) < p$, or it is set to false in every satisfying assignment, and then $\deg_F(\overline{x}) < q$. We say the literal of x that is set to true in every satisfying assignment is *enforced*. An enforcer can be completed into an unsatisfiable $(\leq k, s)$ or $(\leq k, p, q)$-formula by adding the unit clause containing the negation of the enforced literal.

Minimal unsatisfiability. A CNF formula is *minimally unsatisfiable* if it is unsatisfiable but dropping any of its clauses results in a satisfiable formula. Let MU denote the class of all minimally unsatisfiable formulas. The *deficiency* of a CNF formula F is $\delta(F) = |C| - |\mathsf{var}(F)|$. It is known that $\delta(F) > 0$ for any $F \in \mathrm{MU}$ [1]; therefore it is natural to parameterize MU by deficiency and to consider the classes $\mathrm{MU}(d) := \{ F \in \mathrm{MU} : \delta(F) = d \}$ for $d \geq 1$.

Variable elimination. It is well-known that one can eliminate variables of a CNF formula by a process often called *DP-resolution*, after an algorithm of Davis and Putnam [6], as follows. For two clauses C, D with $x \in C, \overline{x} \in D$, the *resolution* rule yields the *resolvent* clause $C \cup D \setminus \{x, \overline{x}\}$. Let F be a CNF and $x \in \mathsf{var}(F)$. We define $F^x := F \setminus F[x] \cup \{C \cup D \setminus \{x, \bar{x}\} \mid C, D \in F; C \cap \overline{D} = \{x\}\}$. In other words, the result of eliminating x from F is the formula that contains all clauses where x does not occur together with all possible non-tautological resolvents on x. It is easy to see that $\exists x\, F$ and F^x are logically equivalent, and in particular, if F is unsatisfiable, so is F^x.

Blocked clauses. A clause C in a CNF F is *blocked in F on the literal $x \in C$* if for every $C' \in F$ with $\overline{x} \in C'$, there exists a variable $y \neq \mathsf{var}(x)$ with $y \in C, \overline{y} \in C'$ or $y \in C', \overline{y} \in C$. A clause is *blocked in F* if it is blocked on at least one of its literals. Blocked clauses are a fundamental SAT preprocessing technique: when C is blocked in F and F is satisfiable, then $F \cup \{C\}$ is also satisfiable [13, 22]; in other words, blocked clauses may be added or removed without impacting satisfiability (notice that this also follows from soundness of variable elimination). We will use the simple corollary that a minimally unsatisfiable formula cannot contain a blocked clause.

QBF. Quantified Boolean formulas generalize propositional logic with quantification. In this paper, we will need only the fragment of *closed prenex 2-QBFs* with one quantifier alternation. A 2-QBF has the form $\exists X \forall Y \Phi(X, Y)$, where X and Y are sets of propositional variables, and Φ is a propositional formula. A 2-QBF is *true* if there exists an assignment $\tau : X \to \{0, 1\}$ such that $\Phi(\tau(X), Y)$ evaluates to true for every assignment to Y, where $\tau(X)$ denotes the substitution of τ values for X into Φ. The formula is *false* if no such assignment τ exists.

Graphs. We only use undirected and simple graphs (i.e., without parallel edges or self-loops). A *graph* G consists of set $V(G)$ of vertices and a set $E(G)$ of edges; we denote the edge between vertices $u, v \in V(G)$ by uv or equivalently vu.

We write $\mathcal{G}_n$ to denote the class of all graphs with $V(G) = [n]$. The *adjacency matrix* A of a graph $G \in \mathcal{G}_n$ is the $n \times n$ $\{0, 1\}$-matrix where the element at row v and column u, denoted by $A(v, u)$, is 1 iff $vu \in E(G)$.

Isomorphisms. For a permutation $\pi : [n] \to [n]$, $\pi(G)$ denotes the graph obtained from $G \in \mathcal{G}_n$ by the permutation π, where $V(\pi(G)) = V(G) = [n]$ and $E(\pi(G)) = \{\, \pi(u)\pi(v) : uv \in E(G) \,\}$. Two graphs $G_1, G_2 \in \mathcal{G}_n$ are *isomorphic* if there is a permutation $\pi : [n] \to [n]$ such that $\pi(G_1) = G_2$; in this case G_2 is an *isomorphic copy* of G_1. A *partially defined graph* [17] is a graph G where $E(G)$ is split into two disjoint sets $D(G)$ and $U(G)$. $D(G)$ contains the *defined* edges, $U(G)$ contains the *undefined* edges. A (*fully defined*) graph is a partially defined graph G with $U(G) = \emptyset$. A partially defined graph G can be *extended* to a graph H if $D(G) \subseteq E(H) \subseteq D(G) \cup U(G)$.

CNF Formulas as graphs. For sets S, S', T, we write $T = S \uplus S'$ if $T = S \cup S'$ and $S \cap S' = \emptyset$. A *2-graph* is an undirected graph $G = (V, E)$ together with a partition of its vertex set into two disjoint *blocks* $V_1 \uplus V_2 = V$. Two 2-graphs $G = (V_1 \uplus V_2, E)$ and $G' = (V_1' \uplus V_2', E')$ are *isomorphic* if there exists a bijection $\phi : V_1 \uplus V_2 \to V_1' \uplus V_2'$ such that $v \in V_i$ if and only if $\phi(v) \in V_i'$, $i = 1, 2$, and $\{u, v\} \in E$ if and only if $\{\phi(u), \phi(v)\} \in E'$. The *clause-literal graph* of a CNF formula F is the 2-graph $G(F) = (V_1 \uplus V_2, E)$ with $V_1 = \mathsf{lit}(F)$, $V_2 = F$, and $E = \{\, \{x, \overline{x}\} : x \in \mathsf{var}(F) \,\} \cup \{\, \{C, \ell\} : C \in F, \ell \in C \,\}$. We refer to the edges $\{\, x, \overline{x} \,\}$ as *variable* edges. It is easy to verify that any two CNF formulas are isomorphic if and only if their clause-literal graphs are isomorphic (as 2-graphs). Note that we can safely assume that the first $|V_1|$ rows of the adjacency matrix of a clause-literal graph correspond to the vertices in V_1, and we will thoroughly do so throughout this paper.

SAT Modulo Symmetries (SMS). SMS [18] is a framework that augments a CDCL (conflict-driven clause learning) SAT solver [8, 23] with a custom propagator that can reason about symmetries, allowing to search modulo isomorphism for graphs in $\mathcal{G}_n$ satisfying a property specified in (quantified) propositional logic.

During search, the SMS propagator can trigger additional conflicts on top of ordinary CDCL and consequently learn *symmetry-breaking clauses*, which exclude isomorphic copies of graphs. More precisely, only those copies are kept which are lexicographically minimal (*canonical*) when considering the rows of the adjacency matrix concatenated into a single vector. A key component is a minimality check, which decides whether a partially defined graph can be extended to a minimal graph; if it cannot, a corresponding clause is learned. For a full description of SMS, we refer to the original work where the framework was introduced [18]. In SMS, it is possible to specify a partition of the vertex set and restrict the symmetry breaking to those permutations that preserve the partition. In Section 3.1, we explain how we can specify partitions to efficiently generate formulas (represented by 2-graphs) modulo isomorphism with SMS.

3 Basic encoding

In this section we explain the methodology all of our investigations build upon, and review results already obtainable with it. In the next section, we will delve into technical details and improvements that are necessary to scale up this basic approach.

The idea is to first list all possible number of clauses a smallest (k, s) or (k, p, q)-formula can have, and then decide whether an unsatisfiable formula exists with increasing m. For each m, we split the decision problem further by specifying number of variables n the sought formula has. For fixed m and n, we reduce this task to deciding the satisfiability of a suitable quantified Boolean formula and give it to QBF-enabled SMS. Because we gradually increase m, the first time we hit a satisfiable instance we know the formula thus produced is smallest possible. The desired QBF should have its models correspond to unsatisfiable (k, s)-formulas with n variables and m clauses. Since unsatisfiability is coNP-complete, we cannot hope to obtain a polynomial-size propositional encoding (unless NP = coNP), and instead we use a 2-QBF of the form $\exists X \forall Y \Phi^{n,m}_{k,s}(X) \wedge \neg\Sigma^{n,m}(X, Y)$ (or $\exists X \forall Y \Phi^{n,m}_{k,p,q}(X) \wedge \neg\Sigma^{n,m}(X, Y)$), where $\Phi^{n,m}_{k,s}(X)$ ($\Phi^{n,m}_{k,p,q}(X)$) expresses that X represents a (k, s)-formula ((k, p, q)-formula) with n variables and m clauses, and $\Sigma^{n,m}(X, Y)$ expresses that the assignment represented by Y satisfies X.

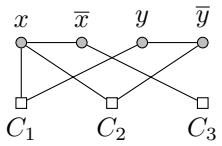
3.1 Hard-coding the first part of the clause-literal graph

Before we delve into the details of the encoding, we observe the following fact about the lexicographically minimal matrix of a clause-literal graph.

Let $\mathrm{pos}(i) = r$ if the rth row/column of the adjacency matrix represents the positive literal of the ith variable, and similarly for $\mathrm{neg}(i)$. The first block of a clause-literal graph contains the vertices corresponding to literals, i.e., we have $\mathrm{pos}(i), \mathrm{neg}(i) \in [2n]$ for all $i \in [n]$.

▶ **Theorem 1.** *The lexicographically minimal matrix of any clause-literal graph is antidiagonal in the upper-left (variables) block, i.e., for all $i, j \leq 2n$, $A(i, j) = 1$ iff $i + j = 2n + 1$. For the ordering of literals, this means that for each i we have $\{\mathrm{pos}(i), \mathrm{neg}(i)\} = \{j, 2n + 1 - j\}$ for some $j \in [n]$.*

Proof. Towards a contradiction, consider a lexicographically minimal adjacency matrix of some clause-literal graph, and the row i, where antidiagonality is first violated, because the 1-entry is in column $2n + 1 - j$ for $j > i$ (and not $j = i$ as it should be; notice that $j < i$ is impossible since the literal $2n + 1 - j$ would have to be adjacent to both j and i). Then, swapping the vertices $2n + 1 - j$ and $2n + 1 - i$ yields a lexicographically smaller matrix. ◀



	$\overline{x}$	$\overline{y}$	y	x	C_1	C_2	C_3
$\overline{x}$	0	0	0	1	0	0	1
$\overline{y}$		0	1	0	0	1	0
y			0	0	1	0	0
x				0	1	1	0
C_1					0	0	0
C_2						0	0
C_3							0

Figure 1 Consider the formula $F = \{C_1, C_2, C_3\}$ with $C_1 = \{x, y\}$, $C_2 = \{x, \overline{y}\}$, and $C_3 = \{\overline{x}\}$. We see the corresponding 2-graph and the upper part of its lexicographically minimal adjacency matrix. Observe how the left part is indeed antidiagonal.

Theorem 1 shows that we can hard-code the top-left $2n \times 2n$ matrix of the adjacency matrix. Doing so has the following benefits. The immediate advantage is that with fewer undecided variables to solve, SMS terminates more quickly. Also, this breaks the symmetries from reordering the literal nodes of the clause-literal matrix. Finally, as we will see more clearly in the next part, fixing the matching between literal nodes reduces the size of the encoding since we no longer need to describe the cardinality constraints on how many times a variable can occur *conditionally* on an undecided matching among literal nodes. Figure 1 shows an example of a clause-literal graph and its corresponding lexicographically minimal adjacency matrix.

3.2 Encodings for formulas

In this part, we describe the detailed construction of the specific 2-QBF we use, whose models correspond to unsatisfiable (k, s)-formulas with n variables and m clauses. We write our encoding in the standard circuit-QBF QCIR format [14].

We express cardinality constraints using cardinality networks [2]. A cardinality network Card_b^a takes b binary inputs and outputs the most significant a inputs ordered from more significant to less. For the purpose of illustration, suppose we want variable z to be true if and only if exactly d variables out of $x_1, \ldots, x_b$ are true. Let $(y_1, \ldots, y_d, y_{d+1}) := \mathrm{Card}_b^{d+1}(x_1, \ldots, x_b)$. Note that this way y_i is true if and only if there are at least i true inputs among $x_1, \ldots, x_b$. So we can define $z := y_d \wedge \neg y_{d+1}$. When the number of inputs and outputs is clear from the context, we just write Card. Let $i, i' \in [n], j < j' \in [m]$, we define $a_{i,i'} := A(\mathrm{pos}(i), \mathrm{pos}(i'))$, $a_{i,-i'} := A(\mathrm{pos}(i), \mathrm{neg}(i'))$ and $a_{-i,-i'} := A(\mathrm{neg}(i), \mathrm{neg}(i'))$ for edges between literal vertices, $e_{i,j} := A(\mathrm{pos}(i), j + 2n)$ and $e_{i,j} := A(\mathrm{neg}(i), j + 2n)$ for edges from a literal node to a clause node, and $c_{j,j'} := A(j + 2n, j' + 2n)$ for edges between clause nodes. Take a sufficiently large x. For all $i \in [n], t \in [n] \cup [-n]$ and $j \in [m]$, define

$$
\begin{aligned}
(\omega_{t,1}, \omega_{t,2}, \ldots, \omega_{t,x}) &:= \mathrm{Card}(e_{t,1}, e_{t,2}, \ldots, e_{t,m}), \\
(\sigma_{i,1}, \sigma_{i,2}, \ldots, \sigma_{i,x}) &:= \mathrm{Card}(e_{i,1}, e_{i,2}, \ldots, e_{i,m}, e_{-i,1}, e_{-i,2}, \ldots, e_{-i,m}), \text{ and} \\
(\tau_{j,1}, \tau_{j,2}, \ldots, \tau_{j,k+1}) &:= \mathrm{Card}(e_{1,j}, e_{2,j}, \ldots, e_{n,j}, e_{-1,j}, e_{-2,j}, \ldots, e_{-n,j}).
\end{aligned}
$$

The following are some useful properties expressed in propositional logic which we use as components in the desired 2-QBF. F_1 expresses that there are no edges between two clause nodes. F_2 expresses that each literal occurs at least once. F_3 expresses that a literal and its negation cannot occur in the same clause. F_4^v expresses that each variable occurs at most s times. F_5 expresses that each clause contains exactly k literals.

$$F_1 := \bigwedge_{0 \le j < j' < m} \neg c_{j,j'}, \quad F_2 := \bigwedge_{i \in [n] \cup [-n]} \bigvee_{j \in [m]} e_{i,j}, \quad F_3 := \bigwedge_{i \in [n], j \in [m]} \neg e_{i,j} \vee \neg e_{-i,j}.$$

$$F_4^v := \bigwedge_{i \in [n]} \neg \sigma_{i,s+1}, \quad F_4^l := \bigwedge_{i \in [n] \cup [-n]} \neg \omega_{i,q+1} \wedge \bigwedge_{i \in [n]} \neg \omega_{i,p+1} \vee \neg \omega_{-i,p+1},$$

$$F_5 := \bigwedge_{j \in [m]} \tau_{j,k} \wedge \neg \tau_{j,k+1}.$$

It is easy to see that a smallest (k, s) or (k, p, q)-formula must be minimally unsatisfiable, and thus we can also require (in $\Phi_{k,s}^{n,m}$) that it contain no blocked clauses. Given $i \in [n] \cup [-n]$ and $j, j' \in [n]$, define $\psi_{i,j,j'}$ and $\varphi_{i,j}$ as follows. Here $\varphi_{i,j}$ expresses that the j-th clause is not blocked on the literal i.

$$\psi_{i,j,j'} := \bigwedge_{\substack{i' \in [n] \cup [-n] \\ i' \neq i, -i}} (\neg e_{i',j} \vee \neg e_{-i',j'}), \quad \varphi_{i,j} := \neg e_{i,j} \vee \bigvee_{\substack{j' \in [m] \\ j' \neq j}} (e_{-i,j'} \wedge \psi_{i,j,j'}).$$

F_6 expresses that the formula contained no blocked clauses.

$$F_6 := \bigwedge_{\substack{i \in [n] \cup [-n], \\ j \in [m]}} \varphi_{i,j}$$

F_7 hard-codes the matching between the literal nodes.

$$F_7 := \bigwedge_{i \in [n]} a_{i,-i} \wedge \bigwedge_{\substack{i \in [n], i' \in [-n] \\ i \neq -i'}} \neg a_{i,i'} \wedge \bigwedge_{i < i' \in [n]} \neg a_{i,i'} \wedge \bigwedge_{i' < i \in [-n]} \neg a_{i,i'}$$

Let $X := \{ A(i,j) : i < j \in [2n + m] \}$ and $Y = \{ \alpha_i : i \in [m] \}$. For $i \in [n]$, we put $\alpha_{-i} := \neg \alpha_i$. Finally, define $\Phi_{k,s}^{n,m}(X) := F_1 \wedge F_2 \wedge F_3 \wedge F_4^v \wedge F_5 \wedge F_6 \wedge F_7$, $\Phi_{k,p,q}^{n,m}(X) := F_1 \wedge F_2 \wedge F_3 \wedge F_4^l \wedge F_5 \wedge F_6 \wedge F_7$ and

$$\Sigma^{n,m}(X, Y) := \bigwedge_{j \in [m]} \bigvee_{i \in [n] \cup [-n]} \alpha_i \wedge e_{i,j}.$$

3.3 Preliminary findings

To determine the exact value of $\mu(3, s)$ and $\mu(3, p, q)$ for various choices of s, p and q, we enumerate all permissible values of (m, n), where m is the number of clauses and n is the number of variables, from smaller to bigger in the lexicographical order. For each pair (m, n), we solve the formula described in Section 3.2 with SMS.[1] We terminate the solver if it cannot answer within 5 days.

We ran the solver on a Sun Grid Engine (SGE) cluster consisting of heterogeneous machines running Ubuntu 18.04.6 LTS.[2]

We begin by observing that since allowing more occurrences yields a larger class of formulas, by definition, $\mu(k, s) \le \mu(k, s - 1)$, $\mu(k, p, q) \le \min\big(\mu(k, p - 1, q), \mu(k, p, q - 1)\big)$, and $\mu(k, p + q) \le \mu(k, p, q)$. The following lemmas provide preliminary bounds on m and n for our enumeration.

[1] `https://sat-modulo-symmetries.readthedocs.io`
`https://github.com/markirch/sat-modulo-symmetries`

[2] The cluster contains nodes with the following architectures: 2× Intel Xeon E5540 with 2.53 GHz Quad Core, 2× Intel Xeon E5649 with 2.53 GHz 6-core, 2× Intel Xeon E5-2630 v2 with 2.60GHz 6-core, 2× Intel Xeon E5-2640 v4 with 2.40GHz 10-core and 2× AMD EPYC 7402 with 2.80GHz 24-core.

Table 3 Preliminary results based only on the method of this section, for $\mu(3,s)$ (left) and $\mu(3,p,q)$ (right). Since the right table is symmetric with respect to the main diagonal, we only give the upper triangle due to the assumption $q \geq p$. All values for $s \geq 6$ and $q \geq p \geq 3$ are 8: less is not possible by Lemma 2. Lemma 2 also rules out a $(3,2,q)$-formula with 9 clauses and $q \geq 5$.

$s=4$	$s=5$	$s \geq 6$
$[14,16]$	11	8

	$q=1$	$q=2$	$q=3$	$q=4$	$q \geq 5$
$p=1$	∞	∞	$[14,\infty]$	$[11,\infty]$	$[8,\infty]$
$p=2$	–	$[8,20]$	11	10	10
$p \geq 3$	–	–	8	8	8

▶ **Lemma 2.** *An unsatisfiable k-CNF formula that contains a variable of type (p,q) has at least $2^k + |q-p|$ clauses.*

Proof. Let F be an unsatisfiable k-CNF with m clauses, n variables, and x a variable of type (p,q). $F|_{\overline{x}} := \{ C \setminus \{x\} \in F : \overline{x} \notin C \}$, obtained from F by setting x to false, is unsatisfiable, has $n-1$ variables, p clauses of size $k-1$, and $m-p-q$ clauses of size k. Since a clause of size r is falsified by 2^{n-1-r} assignments, and each assignment falsifies some clause, we have $p2^{n-k} + (m-p-q)2^{n-1-k} \geq 2^{n-1}$, and solving for m completes the proof. ◀

▶ **Lemma 3.** *A minimally unsatisfiable (k,s)-formula with m clauses has between $\lceil \frac{k \cdot m}{s} \rceil$ and $m-1$ variables. Similarly, a minimally unsatisfiable (k,p,q)-formula with m clauses has between $\lceil \frac{k \cdot m}{p+q} \rceil$ and $m-1$ variables.*

Proof. With n variables of degree $\leq s$ there are $mk \leq ns$ literal occurrences. For the upper bound, recall that minimally unsatisfiable formulas have positive deficiency. ◀

It is known that an unsatisfiable (3,4)-formula with 16 clauses and an unsatisfiable (3,2,2)-formula with 20 clauses exist [4]. We give the formulas as $E_{3,4}$ and $M_{3,2,2}$ in the appendix. The experimental results combined with this knowledge yield Table 3. One can find a smallest $(3,2,3)$, $(3,2,4)$ and $(3,3,3)$-formula in the appendix as $M_{3,2,3}$, $M_{3,2,4}$, and $M_{3,3,3}$. $M_{3,2,3}$ also serves as a smallest $(3,5)$-formula, and $M_{3,3,3}$ as a smallest $(3,6)$ formula.

A closer look at the time spent on deciding the existence of an unsatisfiable $(3,4)$-formula with different n and m shown in Table 4 reveals that this basic method reaches its limit with formulas of about 13 clauses. Thus, further considerations are called for if we want to determine the precise value for some of the entries in the tables.

Table 4 Time spent deciding the existence of an unsatisfiable $(3,4)$-formula with n variables and m clauses (without/with blocked-clause detection encoded as F_6). Unsolved queries are marked by to, blank areas are out of bounds determined by Lemma 3. All terminated queries were unsatisfiable except the one marked in blue with $n=12$ and $m=16$.

n	$m=8$	$m=9$	$m=10$	$m=11$	$m=12$
6	0.4s/0.6s				
7	0.7s/1.2s	1.9s/2.0s			
8		4.9s/5.9s	7.3s/10.4s		
9			42s/1m	1m/3m	5m/4m
10				12m/7m	43m/17m
11					3h/1.6h

n	$m=13$	$m=14$	$m=15$	$m=16$
10	2h/1h			
11	12h/6h	28h/27h		
12	5d/42h	to	to	to/5d
13		to	to	to
14			to	to
15				to

3.4 A dichotomy theorem for $(3, 1, q)$-SAT

The following extends a result by Kratochvíl, et al. [21, Lemma 2.2].

▶ **Lemma 4** ($\star$). *Let $k \geq 3$ and $p, q \geq 1$ such that $p + q \geq 3$. If there exists an unsatisfiable (k, p, q)-formula, then (k, p, q)-*SAT is NP-hard.

▶ **Theorem 5** (Dichotomy). *For any $p, q \geq 1$, if $p + q < 4$ then $(3, p, q)$-*SAT is solvable in polynomial time, otherwise $(3, p, q)$-SAT is NP-hard.

Proof. $(3, p, q)$-SAT is a special case of $(3, p + q)$-SAT, and $(3, s)$-SAT is in P for $s \leq 3$ [27]. Let $p + q \geq 4$, w.l.o.g., $p \leq q$. If $p = 1$, then $q \geq 3$, if $p \geq 2$, then $q \geq 2$, and unsatisfiable $(3, 1, 3)$-formulas and $(3, 2, 2)$-formulas exist (which are also $(3, p, q)$-formulas for larger p, q, see Table 1). NP-hardness follows from Lemma 4. ◀

The NP-hardness part of Theorem 5 holds even for monotone SAT, where each clause is required to contain only positive or only negative literals [5], with the exception of monotone $(3, 1, 3)$-SAT, for which van Santvliet and de Haan [24] have recently shown that all instances are satisfiable, and monotone $(3, 1, 4)$-SAT, which is still open. Our proof is uniform in the sense that all hardness results rely on Lemma 4.

4 Compound methods

The core method for finding small unsatisfiable (k, s)-CNF and (k, p, q)-CNF formulas is our *SMS encoding* that we have introduced in Section 3.2. This method is quite powerful and lets us produce the smallest formulas and exact lower bounds for formula size. However, since the search space grows very quickly, this method reaches its limits with formulas of about 10–15 clauses, depending on the imposed side constraints. In this section, we show how we improve the results from Table 3 by combining computational search with theoretical analysis, and with several techniques to decrease the size of the search space. The methods for obtaining upper bounds are *disjunctive splitting* and *combining enforcers*. The methods for obtaining lower bounds are *reductions* and *hard-coding part of the adjacency matrix*.

Generating (k, s)-formulas directly is often prohibitively expensive. In such cases we also consider $(\leq k, s)$-formulas that have a few clauses of width smaller than k. Central to the various techniques we employ in this section is the concept of a stairway. A *stairway*, first introduced by Hoory and Szeider [11], is an abstraction of a CNF formula that focuses only on the clauses that are smaller than a given k. More specifically, a stairway $\sigma = (a_1, \ldots, a_r)$ is a finite non-increasing sequence of positive integers. For a fixed integer k, a stairway $\sigma = (a_1, \ldots, a_r)$ represents the set of all CNF formulas $F = \{C_1, \ldots, C_m\}$ where $a_i = k - |C_i|$ for $1 \leq i \leq r$, and $|C_i| = k$ for $r + 1 \leq i \leq m$. Define $\mu(k, s, \sigma)$ to be the number of clauses of a smallest unsatisfiable $(\leq k, s)$-formula with stairway σ, and $\mu(k, p, q, \sigma)$ to be that of the smallest unsatisfiable $(\leq k, p, q)$-formula with stairway σ.

It is straightforward to adapt $\Phi_{k,s}^{n,m}$ to encode, instead of a (k, s)-formula, a $(\leq k, s)$-formula with a given stairway. Instead of requiring every clause to contain exactly k literals, we require every clause to contain at most k literals, and that a certain number of clauses contain less than r literals, for some $1 < r \leq k$. For each $1 < r \leq k$, $j \in [n]$, let N_r be the number of clauses that contains strictly less than r literals. We replace F_5 with F_5' below.

$$F_5' := \bigwedge_{j \in [n]} \neg\tau_{j,k+1} \wedge \bigwedge_{1 < r \leq k} \sum_{j \in [n]} \neg\tau_{j,r} = N_r.$$

4.1 Improved bounds

The following theorems refine the bounds on $\mu(k,s)$ and $\mu(k,p,q)$ using stairways.

▶ **Theorem 6.** $\mu(k,p,q) \geq \min\left(\mu(k,p,q,1^p)+q, \mu(k,p,q-1)\right)$, *where* 1^p *is the stairway of length* p *with each entry being a* 1.

Proof. An unsatisfiable (k,p,q)-formula where at least one literal occurs q times gives an unsatisfiable formula with p $(k-1)$-clauses and shorter in length by q if we set the literal that occurs q times to true. The latter has minimal size $\mu(k,p,q,1^p)$. Taking this into account, we know that $\mu(k,p,q) < \mu(k,p,q-1)$ is only possible if $\mu(k,p,q,1^p)+q < \mu(k,p,q-1)$. ◀

Recall the concept of an enforcer from Section 2. Enforcers can be used to provide upper bounds, as we will show in Theorem 7. When $k=3$, a $(3,s)$-enforcer (or a $(3,p,q)$-enforcer) gives rise, by appending the appropriate unit clause, to an unsatisfiable $(\leq k,s)$-formula (or an unsatisfiable $(\leq k,p,q)$-formula) with stairway (2), and thus by searching for formulas with this stairway we can generate enforcers. In this way, we computed the size of a smallest enforcer in the classes of $(3,s)$-formulas and $(3,p,q)$-formulas, and list them in Table 2. The minimality of the $(3,3,4)$-enforcer was also shown by Jurenka [15] with a theoretical argument. The corresponding formulas can be found in Appendix A.1.

▶ **Theorem 7.** $\mu(k,s) \leq k\cdot(\mu(k,s,(k-1))-1)+1$. *Similarly,* $\mu(k,p,q) \leq k\cdot(\mu(k,p,q,(k-1))-1)+1$.

Proof. Let E be a smallest (k,s)-enforcer, which by definition has size $\mu(k,s,(k-1))-1$ (the (k,p,q) case is analogous). We can obtain an unsatisfiable (k,s)-formula by taking k variable-disjoint copies of E and adding a k-clause containing the negated enforced literals. ◀

Table 5 Smallest size of an unsatisfiable $(\leq k,1,q)$-formula for stairway (1).

	$q=1$	$q=2$	$q=3$	$q=4$	$q\geq 5$
$p=1$	∞	∞	14 or 15	13	11

We compute the table for $\mu(3,1,3,(1))$ and $\mu(3,1,3,(2))$ through a similar exhaustive search as explained in Section 3, and show the results in Table 5. Combining these tables with Theorems 6 and 7, we have the following improvement.

▶ **Corollary 8.** $\mu(3,1,3), \mu(3,1,4) \geq 17$ *and* $\mu(3,1,q) \geq 16$ *for all* $q \geq 5$.

▶ **Corollary 9.** $\mu(3,1,3) \leq 22, \mu(3,1,4) \leq 19$ *and* $\mu(3,1,q) \leq 16$ *for all* $q \geq 5$.

4.2 Disjunctive splitting

We say a CNF formula F is obtained by *disjunctive splitting in* x from CNF formulas F_1, F_2, in symbols $F = F_1 \oplus F_2$, if F can be partitioned into two nonempty sets F_1', F_2' such that the variable x occurs in F_1' positively but not negatively, and appears in F_2' negatively but not positively, and F_i is obtained from F_i' with all occurrences of $x, \overline{x}$ removed. Observe that if F_1, F_2 are unsatisfiable, then also $F = F_1' \cup F_2'$ is unsatisfiable. Hence, when constructing an unsatisfiable (k,s)-CNF or (k,p,q)-CNF formula, we can first try to construct $(\leq k,s)$-formulas or $(\leq k,p,q)$-formulas F_1, F_2 and then combine them to obtain F.

If F is obtained by disjunctive splitting in x from F_1, F_2, and x is added positively to p clauses in F_1 and negatively to q clauses in F_2, we write $F = F_1 \oplus_{p,q} F_2$. Disjunctive splitting can be recursively applied to F_1 and F_2. This allows us to construct a formula from *axioms* which are CNF formulas that we do not further split. For example, the $(2,4)$-CNF formula $\{\{x,y\}, \{\overline{x},y\}, \{\overline{y},z\}, \{\overline{y},\overline{z}\}\}$ can be constructed from the axiom $\{\square\}$.

We can describe the construction by an $\oplus$-*derivation*, an algebraic expression $(\{\square\} \oplus_{1,1} \{\square\}) \oplus_{2,2} (\{\square\} \oplus_{1,1} \{\square\})$. In fact, MU(1) is exactly the class of all formulas that can be constructed by disjunctive splitting from the axiom $\{\square\}$.

This idea was utilized by Hoory and Szeider [11], who proposed an algorithm that decides for given k, s whether $(k,s)\text{-CNF} \cap \text{MU}(1) \neq \emptyset$. This allows us to compute an upper bound on the threshold function $f(k)$. Hoory and Szeider define $\oplus$-derivations to operate on stairways instead of formulas. For $k = 3$, the above $\oplus$-derivation would now read $((3) \oplus_{1,1} (3)) \oplus_{2,2} ((3) \oplus_{1,1} (3))$ and produce the stairway $(1,1,1,1)$. By means of a saturation algorithm, Hoory and Szeider could determine the upper bounds $f(3) \leq 3$, $f(4) \leq 4$, $f(5) \leq 7$, $f(6) \leq 11$, $f(7) \leq 17$, $f(8) \leq 29$, and $f(9) \leq 51$ on the threshold function $f(k)$ (i.e., all $(k, f(k))$-formulas are satisfiable but $(k, f(k)+1)$-SAT is NP-complete), which are still the best known upper bounds.

In this paper, we generalize Hoory and Szeider's method in the following ways: (i) we consider $\oplus$-derivations with more axioms: any unsatisfiable $(\leq k, s)$-formula can serve as an axiom; (ii) we modify the saturation algorithm so that it gives the size of the smallest unsatisfiable (k,s)-formula derivable with respect to the sizes of the formulas that serve as axioms; (iii) we adapt the algorithm also for searching for unsatisfiable (k,p,q)-formulas. We do this in the hope that, by finding suitable axioms with the SMS encoding, we can incorporate them into the $\oplus$-derivations to obtain smaller unsatisfiable (k,s) or (k,p,q)-formulas.

Table 6 shows the size of smallest unsatisfiable $(3,s)$ and $(3,p,q)$-formulas generated by disjunctive splitting with $\{\square\}$ being the only axiom. The corresponding $\oplus$-derivations can be found in the appendices. We determined $\mu(3,s,1^r)$ for all $1 \leq r < s$ and $\mu(3,p,q,1^r)$ for all $1 \leq r < q$, but this did not yield any new upper bounds. We were more successful with the method of disjunctive splitting in the case of $k = 4$, which we discuss in Section 5.

Table 6 Size of the smallest $(3,s)$-formulas and $(3,p,q)$-formulas in MU(1).

$s \leq 3$	$s = 4$	$s = 5$	$s = 6$	$s = 7$	$s \geq 8$
∞	16	12	10	9	8

	$q = 1$	$q = 2$	$q = 3$	$q = 4$	$q \geq 5$
$p = 1$	∞	∞	22	19	16
$p = 2$	–	∞	12	10	10
$p = 3$	–	–	10	9	9
$p \geq 4$	–	–	–	8	8

4.3 Hard-coding part of the matrix

After we narrowed down the search scope with tighter bounds, this and the next part deal with deciding the missing values in Table 1 and the techniques involved. These techniques allow us to determine the existence of unsatisfiable $(3,4)$-formulas (or $(3,2,2)$,$(3,1,q)$-formulas) whose size is too large to be exhaustively searched by SMS directly.

In this part, we determine the value of $\mu(3,1,3)$ and $\mu(3,1,3)$. The technique here is hard-coding the part of the matrix that corresponds to the occurrences of some variables/literals. The motivation is that with less undecided values in the matrix to solve, SMS terminates more quickly. Suppose we want to fix both positive and negative occurrences of $\mathcal{V}$ variables

$A(i, 2n+j)$	$1\ldots$	$m-9$	$m-6$	$m-3$	m
1 (x_1)					1
2 ($\overline{x_1}$)				1 1 1	
3 (x_2)					1
4 ($\overline{x_2}$)			1 1 1		
5 (x_3)					1
6 ($\overline{x_3}$)		1 1 1			
7 (a_1)				1	
8 (b_1)				1	
9 (a_2)			1		
10 (b_2)			1		
11 (a_3)		1			
12 (b_3)		1			

Figure 2 The first rows of the matrix determined as a result of our choice for $\mathrm{pos}(i)$ and $\mathrm{neg}(i)$. The omitted values are all 0.

$x_1, x_2, \ldots, x_{\mathcal{V}}$ and only the positive occurrences of $\mathcal{L}$ other variables $x_{\mathcal{V}+1}, x_{\mathcal{V}+2}, \ldots, x_{\mathcal{V}+\mathcal{L}}$. To do this in a way that is compatible with the minimality check of SMS, we need to adjust the following two things. First, when generating the encoding, we stipulate that the first rows in the matrix correspond to $x_1, \overline{x}_1, x_2, \overline{x}_2, \ldots, x_{\mathcal{V}}, \overline{x}_{\mathcal{V}}, x_{\mathcal{V}+1}, x_{\mathcal{V}+2}, \ldots x_{\mathcal{V}+\mathcal{L}}$ and adjust pos and neg to the following.

$$\mathrm{pos}(i) := \begin{cases} 2i-1 & \text{if } i \leq \mathcal{V}, \\ i+\mathcal{V} & \text{otherwise;} \end{cases} \qquad \mathrm{neg}(i) := \begin{cases} 2i & \text{if } i \leq \mathcal{V}, \\ 2n+\mathcal{V}-i+1 & \text{otherwise.} \end{cases}$$

Second, we start SMS with the minimality check restricted to the refined partition

$$\{\{1\}, \{2\}, \ldots, \{2\mathcal{V}+\mathcal{L}\}, [2\mathcal{V}+\mathcal{L}+1, 2n], [2n+1, 2n+m]\}$$

of the original $\{[1, 2n], [2n+1, 2n+m]\}$. Given the specific assumption on the variables and literals fixed, the minimality condition will determine the values of the first $2\mathcal{V}+\mathcal{L}$ rows in the matrix. An example to this will be given shortly after.

To determine the value of $\mu(3, 1, 3)$ and $\mu(3, 1, 4)$, we prove the following lemma that argues about the presence of certain "partially known" $\leq k$-CNFs, so we can fix them. As explained in Section 2, we write $\leq k$-CNFs in matrix form. To argue about "partially known" $\leq k$-CNFs, we extend the matrix notation to what is essentially a first-order language of $\leq k$-CNFs. We use lowercase letters $x, y, z, a, b, c, \ldots$ to denote symbols to be interpreted by propositional literals, positive or negative. We use $^-$ to denote negation: if x is interpreted as some literal, then $\overline{x}$ must be interpreted as its negation. Two different literal symbols may be interpreted by different literals, or they may be interpreted by the same literal, or even by the two literals of the same variable. When a position in the matrix is left blank, we leave the corresponding literal unconstrained. We then say a formula F is *of the form* M if the symbols in the matrix M can be interpreted by the literals of F to yield a matrix of F. A clause $C \in F$ is *singular* if $\deg(x) = 1$ for all $x \in C$.

Table 7 Results of the search for unsatisfiable $(3,1,3)$-formulas for different numbers of variables and clauses after hard-coding the first rows as described. All queries were unsatisfiable except the one marked in blue with $m = 22$ and $n = 21$. For each m, the lower bound for the choice of n is by Lemma 3, and the upper bound is $m - 2$ since we know the smallest size of a $(3,1,3)$-formula in MU(1). There are two values for each pair of m and n. The values on the left indicate the time spent on the case where we assume there is a unique singular clause. The values on the right indicate that of the case where we assume there are at least two singular clauses. The cases superscribed with $\star$ and ▲ did not terminate within a timeout of 6 days and were further split into sub-cases and time shown is the sum of time spent on each of the sub-cases. The cases superscribed with $\star$ are split into 1258 cases in terms of where $\overline{a_1}, \overline{b_1}, \overline{a_2}$ and $\overline{a_2}$ occur, modulo symmetries. The ones superscribed with ▲ are split into 136 cases in terms of where $\overline{y_1}, \overline{y_2}$ and $\overline{y_3}$ occur, modulo symmetries. It is worth noting that both sets of case distinctions are generated automatically without symmetry by reformulating them as graph problems and giving the corresponding encoding to SMS. An unsatisfiable formula is found in this case, assuming there exists a unique singular clause. Upon inspection, this formula is revealed to be composed of 3 enforcers in the spirit of Theorem 7.

	$m=17$	$m=18$	$m=19$	$m=20$	$m=21$	$m=22$
$n=13$	5.3s/7.4m					
$n=14$	16s/17m	38s/1.8h				
$n=15$	19m/1.1h	11m/28.7h	12m/43.8h	53m/13.7h*		
$n=16$		7.2m/20h	7.4h/5.8h*	1.7d/1.2d*	1.7d/9.9d*	
$n=17$			3.8d/9.0h*	15.9h$^▲$/1.8d*	2.1d$^▲$/14.3d*	
$n=18$				1.2d$^▲$/2.5d*	3.6d$^▲$/34.3d*	
$n=19$					11.6d$^▲$/56.9d*	
$n=21$						29.2m/–

▶ **Lemma 10** ($\star$). *Let $q \geq 3$ and let F be a smallest unsatisfiable $(3,1,q)$-formula.*

1. *There is a singular clause in F.*
2. *If $\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \in F$ is singular, then $F[x_i] \cap F[x_j] = \left\{ \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \right\}$ for any $i \neq j \in \{1,2,3\}$, and $\deg(\overline{x_1}) = \deg(\overline{x_2}) = \deg(\overline{x_3}) \geq 3$.*
3. *Let $\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \in F$ be a singular clause and let a be a literal such that $\deg(a) = 1$ and $\mathsf{var}(a) \neq \mathsf{var}(x_i)$ for all $i \in \{1,2,3\}$. If $a \in \bigcup F[x_i]$, then $\overline{a} \notin \bigcup F[x_j]$ for any $i \neq j \in \{1,2,3\}$.*
4. *If there is a unique singular $\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \in F$, then $\bigcup_{i=1,2,3} F[x_i]$ is of the form $\begin{pmatrix} x_1 & \overline{x}_1 & \overline{x}_1 & \overline{x}_1 & \overline{x}_2 & \overline{x}_2 & \overline{x}_2 & \overline{x}_3 & \overline{x}_3 & \overline{x}_3 \\ x_2 & a_1 & & & a_2 & & & a_3 & & \\ x_3 & b_1 & & & b_2 & & & b_3 & & \end{pmatrix}$ where $\deg(a_i) = \deg(b_i) = 1$ for all $i \in \{1,2,3\}$.*

When searching for a smallest unsatisfiable $(3,1,3)$-formula, we distinguish the following two cases, depending on whether there is a unique clause whose literals only occur once. If there is, then we hard-code the partial formula $\begin{pmatrix} x_1 & \overline{x}_1 & \overline{x}_1 & \overline{x}_1 & \overline{x}_2 & \overline{x}_2 & \overline{x}_2 & \overline{x}_3 & \overline{x}_3 & \overline{x}_3 \\ x_2 & a_1 & & & a_2 & & & a_3 & & \\ x_3 & b_1 & & & b_2 & & & b_3 & & \end{pmatrix}$ in the matrix by setting the first 12 rows to represent $x_1, \overline{x}_1, x_2, \overline{x}_2, x_3, \overline{x}_3, a_1, b_1, a_2, b_2, a_3$ and b_3, starting with the ordered partition $\{\{1\}, \{2\}, \ldots, \{12\}, [13, 2n], [2n+1, 2n+m]\}$, and hard-coding the first 12 rows of the matrix thereby determined. The fixed rows are shown in Figure 2 as an example. Otherwise, there are more than one singular clause and we hard-code the partial formula $\begin{pmatrix} x_1 & \overline{x}_1 & \overline{x}_1 & \overline{x}_1 & \overline{x}_2 & \overline{x}_2 & \overline{x}_2 & \overline{x}_3 & \overline{x}_3 & \overline{x}_3 & y_1 \\ x_2 & & & & & & & & & & y_2 \\ x_3 & & & & & & & & & & y_3 \end{pmatrix}$. When searching for a smallest unsatisfiable $(3,1,4)$-formula, we follow the same rationale but split both of these cases into 4 cases depending on the size of $\{x_i \mid \deg^{\mathsf{var}}(x_i) = 5, i \in \{1,2,3\}\}$.

Combining the computational results in Tables 7 and 8 with the previous lemma, we have the following result.

▶ **Theorem 11.** *$\mu(3,1,3) \geq 22$ and $\mu(3,1,4) \geq 19$.*

Table 8 Results of the search for unsatisfiable $(3, 1, 4)$-formula for different numbers of variables and clauses after hard-coding the first rows as described. The timeout is 11 days. For each m in the table, the lower bound for the choice of n is by Lemma 3, and the upper bound is $m - 2$ since we know that the smallest size of a $(3, 1, 4)$-formula in MU(1). There are eight values for each pair of m and n divided into two rows of four values each. The upper row shows the amount of time spent on the cases where we assume there is a unique singular clause; the lower row shows the cases with at least two singular clauses. The four values in a row correspond to cases based on the number of variables out of x_1, x_2, x_3 that are of degree 5 (0, 1, 2, 3 left-to-right). An unsatisfiable formula is found only for the case where $m = 19$, $n = 18$ and $\deg^{\mathsf{var}}(x_1) = \deg^{\mathsf{var}}(x_2) = \deg^{\mathsf{var}}(x_3) = 5$. Upon inspection, this formula is revealed to be composed of 3 enforces in the way of Theorem 7.

	$m = 17$	$m = 18$	$m = 19$
$n = 11$	(1.1s,1.1s,1.1s,0.7s)	(–,–,1.1s,0.8s)	
	(19s,15s,12s,13s)	(–,–,13s,14s)	
$n = 12$	(1.8s,2.0s,2.0s,2.0s)	(2.2s,2.5s,2.4s,2.7s)	(4.7s,2.7s,2.8s,3.0s)
	(1.1m,1.0m,0.6m,13s)	(1.2m,1.2m,0.5m,0.5m)	(2.0m,1.4m,1.9m,1.6m)
$n = 13$	(15s,7.1s,4.0s,6.4s)	(5.5s,10s,12s,8.2s)	(12s,9.0s,6.0s,14s)
	(4.5m,1.1m,0.5m,1.0m)	(6m,4.9m,0.9m,1.5m)	(11m,5.1m,10m,9.5m)
$n = 14$	(24s,43s,14s,12s)	(25s,56s,26s,32s)	(1.2m,40s,1.2m,34s)
	(9.1m,2.4m,1.2m,1.1m)	(34m,8.4m,11m,3.6m)	(1.7h,3.0h,2.8h,0.9h)
$n = 15$	(1.0m,1.7m,17s,21s)	(17m,2.7m,49s,1.8m)	(2.3m,1.0h,1.3m,9.7m)
	(42m,10m,2.3m,1.9m)	(1.7h,50m,22m,22m)	(33.2h,17.1h,2.6h,1.4h)
$n = 16$		(39m,24m,36m,3.3m)	(14.1h,4.5h,34m,30m)
		(19.7h,47m,24m,40m)	(t.o.,6.4d,3.3h,2.5h)
$n = 17$			(7.2d,12.0h,4.8h,2.6h)
			(t.o.,t.o.,10.1h,2.3h)
$n = 18$			(t.o.,5.6d,6.5h,1.1h)
			(2.4d,t.o.,6.3d,10.4h)

4.4 Reduction

In this section, we describe the final technique that allow us to determine the value of $\mu(3, 4)$ and $\mu(3, 2, 2)$, and thus complete Table 1. The idea is to reduce a $(3, 4)$-formula (or $(3, 2, 2)$-formula) to a smaller $(\leq 3, 4)$-formula (or $(\leq 3, 2, 2)$-formula) that is equisatisfiable, so that the question of the existence of a certain formula is reduced to that of the existence of a certain, smaller formula. We then use SMS to determine the existence of such small formulas.

It is difficult to prove any useful properties about general unsatisfiable $(3, 4)$-formulas (or $(3, 2, 2)$-formulas), but since we exhaustively search from smaller to bigger formulas, we can restrict our search to *minimal* (in terms of the number of clauses) $(3, 4)$-formulas (or $(3, 2, 2)$-formulas). We reduce such an F to a smaller unsatisfiable $(\leq 3, 4)$-formula (or $(\leq 3, 2, 2)$-formula) by replacing all subsets of clauses from F that fit into one of the three forms below with a single 2-clause $\begin{pmatrix} c \\ d \\ \times \end{pmatrix}$. For each replacement operation, the symbols c, d are instantiated separately, i.e., they could be instantiated differently each time. Each replacement is tantamount to a sequence of variable eliminations, and thus is sound (preserves unsatisfiability).

1. $\begin{pmatrix} x & \overline{x} & \overline{x} & w & \overline{w} & \overline{w} & \overline{a} & c \\ a & a & \overline{a} & b & b & \overline{b} & \overline{b} & d \\ z & y & \overline{y} & z & v & \overline{v} & z & \overline{z} \end{pmatrix}$ for some $\deg^{\mathsf{var}}(x) = \deg^{\mathsf{var}}(w) = 3$ and $\deg^{\mathsf{var}}(y) = \deg^{\mathsf{var}}(v) = 2$.

Elimination sequence: $\mathsf{var}(y), \mathsf{var}(v), \mathsf{var}(x), \mathsf{var}(w), \mathsf{var}(a), \mathsf{var}(b), \mathsf{var}(z)$.

2. $\begin{pmatrix} x & \overline{x} & \overline{x} & c \\ a & a & \underline{a} & \overline{a} \\ d & y & \overline{y} & d \end{pmatrix}$ for some $\deg^{\mathsf{var}}(x) = 3$, $\deg^{\mathsf{var}}(y) = 2$ and d. Eliminate: $\mathsf{var}(y), \mathsf{var}(x), \mathsf{var}(a)$.
3. $\begin{pmatrix} y & \overline{y} \\ c & c \\ d & d \end{pmatrix}$ for some $\deg^{\mathsf{var}}(y) = 2$ and $\deg^{\mathsf{var}}(c) = \deg^{\mathsf{var}}(d) = 4$. Eliminate: $\mathsf{var}(y)$.

Given the number of clauses m and the number of variables n of the formula, the following lemma helps us narrow down possibilities in terms of how many subsets there are that fit into each of the forms.

▶ **Lemma 12** ($\star$)**.** *Let F be an unsatisfiable $(3,4)$-formula of minimal size. Let x, y, u, w be literals such that $\mathsf{var}(y) \neq \mathsf{var}(u)$ and $\mathsf{var}(x) \neq \mathsf{var}(w)$, $\deg^{\mathsf{var}}(y) = \deg^{\mathsf{var}}(u) = 2$ and $\deg^{\mathsf{var}}(x) = \deg^{\mathsf{var}}(w) = 3$. We have the following facts:*

1. *If a literal a occurs in F, then $\overline{a}$ also occurs in F.*
2. *If $\begin{pmatrix} a \\ b \end{pmatrix} \in F$, then $\mathsf{var}(a) \neq \mathsf{var}(b)$.*
3. *$F[y]$ is of the form $\begin{pmatrix} y & \overline{y} \\ a & a \\ b & b \end{pmatrix}$.*
4. *$F[x]$ is of the form $\begin{pmatrix} x & \overline{x} & \overline{x} \\ a & a & a \\ b & c & d \end{pmatrix}$.*
5. *$F[y] \cap F[u] = \emptyset$.*
6. *$F[x] \cap F[w] = \emptyset$.*
7. *Either $F[x] \cap F[y] = \emptyset$, or $F[x]$ is of the form $\begin{pmatrix} x & \overline{x} & \overline{x} \\ a & a & \underline{a} \\ z & y & \overline{y} \end{pmatrix}$.*
8. *If $F[x] = \begin{pmatrix} x & \overline{x} & \overline{x} \\ a & a & \underline{a} \\ z & y & \overline{y} \end{pmatrix}$, then $F[a] = \begin{pmatrix} x & \overline{x} & \overline{x} & b \\ a & a & \underline{a} & \overline{a} \\ z & y & \overline{y} & z \end{pmatrix}$, and $\deg^{\mathsf{var}}(b) = \deg^{\mathsf{var}}(z) = 4$.*

▶ **Lemma 13.** *A smallest $(3,2,2)$-formula has even size and no variables of degree 3.*

Proof. It is possible to prove fact 4 from Lemma 12 for a minimal $(3,2,2)$-formula as well (it does not follow automatically, as a minimal $(3,2,2)$-formula is not necessarily a minimal $(3,4)$-formula, but the proof is based on the same idea). A variable x of degree 3 then implies there is a literal a of degree 3; a contradiction in a $(3,2,2)$-formula. So, a minimal $(3,2,2)$-formula contains only variables of type $(1,1)$ and $(2,2)$, and has an even number of literal occurrences. With $k = 3$, the number of clauses must be even. ◀

One caveat to searching for a formula of a reduced profile using SMS is that the reduction can reduce degrees of some variables and literals. This means that some of the variables or literals in the shorter clauses must occur strictly fewer times than the general bound s (or p, q). To accommodate for this, we count each literal that occurs in a 2-clause twice. However, this means that the literals in the 2-clauses whose degree was not reduced by the steps above may exceed their degree cap under this way of counting. To adjust to this, we sum up the number of exceeding counts and call it the *surplus* of the formula. Given a specific reduction, we know the exact value of allowed surplus for the reduced formula, and so we can include it as a part of the constraints. Here as an example, we write out the formula for the case of bounded variable degree. The formula for the literal case can be defined similarly. For all $i \in [n] \cup [-n], j \in [m]$, define $e'_{i,j} := e_{i,j} \wedge \neg\tau_{j,k}$. For all $i \in [m]$ and $t \in [m] \cup [-m]$, define $(\sigma'_{i,1}, \sigma'_{i,2}, \ldots, \sigma'_{i,s+s}) := \mathrm{Card}(\sigma_{i,1}, \sigma_{i,2}, \ldots, \sigma_{i,s}, e'_{i,1}, \ldots, e'_{i,m}, e'_{-i,1}, \ldots, e'_{-i,m})$. Suppose $\mathcal{S}$ is the value of surplus. Then we can define

$$F_{\mathrm{surp}} := \sum_{i \in [n]} \sigma'_{i,s+1} + \cdots + \sigma'_{i,s+s} = \mathcal{S}.$$

We call the combination of the number of variables n, the number of clauses m, a stairway σ and the number of surplus $\mathcal{S}$ a *profile*. For further speed-up, we also hard-code a

Table 9 Number $\mathcal{V}$ of hard-coded variables of degree 3 we chose and time spent determining the existence of a formula with each profile from reduction for previously unsolved cases with n variables and m clauses from Table 4. Naturally, in each case $\mathcal{V}$ is no greater than the total number of variables of degree 3 after reduction. In the table on the right, the number of hard-coded variables of degree 3 is always 0 because by Corollary 13 no variable of degree 3 exists.

n	m	n'	m'	#2-cl	$\mathcal{S}$	$\mathcal{V}$	time
		9	11	1	1	3	0.2s
		10	12	2	0	2	2.0s
12	14	8	10	2	1	0	32s
		5	7	1	2	0	0.3s
		6	8	2	2	0	1.5s
		9	11	3	0	0	62s
		12	15	0	0	3	6.5s
12	15	11	14	1	0	1	2.1h
		9	12	1	1	0	41m
		11	13	2	0	3	0.7s
		9	11	2	1	2	1.3s
13	15	6	8	1	2	0	1.0s
		7	9	2	2	0	9.4s
		10	13	3	0	1	1.6h
		8	10	3	1	0	33s

n	m	n'	m'	#2-cl	$\mathcal{S}$	time
12	16	12	16	0	0	5d
13	16	11	14	2	0	7m
14	16	10	12	4	0	5.2s
14	18	13	17	1	0	21d
15	18	12	15	3	0	25m
16	18	11	13	5	0	4.5s

number of variables of degree 3 in the same way as described in the previous part. Let F be a smallest unsatisfiable $(3,4)$-formula and let $x_1, x_2, \ldots, x_{\mathcal{V}}$ be $\mathcal{V}$ degree 3 variables in F whose occurrences we want to fix. By Lemma 12, the formula is of the following form. We fix the first $3\mathcal{V}$ rows of the matrix thereby determined.

$$\begin{pmatrix} x_1 & \overline{x}_1 & \overline{x}_1 \\ a_1 & a_1 & a_1 \end{pmatrix} \begin{pmatrix} x_2 & \overline{x}_2 & \overline{x}_2 \\ a_2 & a_2 & a_2 \end{pmatrix} \ldots \begin{pmatrix} x_{\mathcal{V}} & \overline{x}_{\mathcal{V}} & \overline{x}_{\mathcal{V}} \\ a_{\mathcal{V}} & a_{\mathcal{V}} & a_{\mathcal{V}} \end{pmatrix} \ldots$$

▶ **Lemma 14.** *A smallest unsatisfiable $(3,4)$-formula (or $(3,2,2)$-formula) with m variables and n clauses exists only if a formula of one of the profiles in Table 9, left (right) exists.*

Proof. We enumerate all possible numbers of variables of degree 2 and 3 in the (unreduced) formula, and list all possibilities of how the clauses of variables of degree 2 and 3 overlap according to Lemma 12. We then perform the reduction to each possibility and obtain a list of profiles that the unsolved cases from Table 9 reduce to. ◀

We tested the existence of each of the profiles from Table 9 with SMS and all of them returned negative. Combining these experiment results with Lemma 14, we obtain the following theorem.

▶ **Theorem 15.** *$\mu(3,4) > 15$ and $\mu(3,2,2) > 18$.*

5 A smaller unsatisfiable $(4,5)$-formula

We now turn our attention to unsatisfiable CNF formulas with clauses of length 4. As mentioned in the introduction, $f(4) = 4$, i.e., $s = 5$ is the smallest value so that an unsatisfiable $(4,s)$-formula exists. We found an unsatisfiable $(4,s)$-formula with 235 clauses,

improving upon the formulas provided by Stříbrná [26] (449 clauses) and Knuth [20, p. 588] (257 clauses). Knuth's formula $K_{4,5}$ can be described by the $\oplus$-derivation from the axiom $\{\square\}$ depicted as a directed acyclic graph in Figure 3.

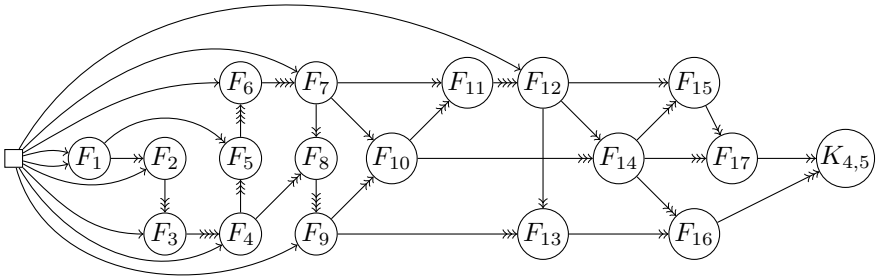


Figure 3 The $\oplus$-derivation of Knuth's formula $K_{4,5}$. Each node has two incoming arcs; the number of arrowheads denotes the values p, q in $\oplus_{p,q}$.

This formula has 257 clauses and 256 variables. However, 22 of these variables occur twice (introduced in F_1) and 31 variables occur in three clauses (introduced in F_2). Hence we can identify each of the variables with 2 occurrences with one variable with 3 occurrences, still keeping the number of occurrences of the new identified variable within the bound 5. By this method, we can save 22 variables, and indeed, Knuth states his formula to have $256 - 22 = 234$ variables. With our modified saturation algorithm we could show that $K_{4,5}$ is a smallest unsatisfiable $(4, 5)$-formula in MU(1). Thus, for finding a smaller formula, one needs to search outside MU(1), and outside the class of formulas that can be obtained from an MU(1) formula by identifying pairs of low-occurrence variables.

Finding an entire unsatisfiable $(4, 5)$-formula with SMS does not seem feasible. However, we can search for an unsatisfiable $(\leq 4, 5)$-formula F that represents a stairway σ with fewer clauses than any formula in MU(1) that represents the same stairway σ. Thus way we can use F as an additional axiom in $\oplus$-derivations and this way possibly find a smaller unsatisfiable $(4, 5)$-formula. We considered all stairways $\sigma \in \{(3), (2), (3, 2), (2, 2), (2, 2, 2), (1), (3, 1), (2, 1), (2, 2, 1), (1, 1), (3, 1, 1), (2, 1, 1), (2, 2, 1, 1), (1, 1, 1), (3, 1, 1, 1), (2, 1, 1, 1), (1, 1, 1, 1)\}$ and run our SMS encoding to find an unsatisfiable formula F with at most 13 clauses and fewer clauses than a smallest MU(1)-formula for σ.

The search resulted in two such formulas within a timeout of five days. One formula has 7 clauses for the stairway $(3, 1, 1, 1)$ and the other has 8 clauses for the stairway $(3, 2)$; shortest MU(1) formulas for these stairways have 8 and 9 clauses, respectively. Using the first of these two formulas as axiom indeed reduces the size of the unsatisfiable $(4, 5)$-formula from 257 to 235, since the axiom is used several times; the second formula can be derived by an $\oplus$ operation from the new axiom and axiom $\{\square\}$. Our smaller unsatisfiable $(4, 5)$-formula can be obtained by replacing F_6 with F_6' in the $\oplus$-derivation from Figure 3, and using F_6' as an additional axiom, where the two formulas are as follows (rows are variables, columns are clauses, $+/-$ indicates positive/negative occurrence; c.f. the appendix).

$$F_6' = \begin{pmatrix} - & & & & & & + \\ & - & - & - & - & + & \\ - & & - & + & + & & - \\ - & - & + & & + & & - \\ - & + & & - & + & & - \end{pmatrix} \qquad F_6 = \begin{pmatrix} & & & - & - & - & - & + \\ & & & & - & - & - & + & + \\ - & - & - & - & - & + & & & \\ - & - & - & - & + & & & & \\ - & - & - & + & & & & & \\ - & + & & & & & - & + & \end{pmatrix}$$

6 Conclusion

We have identified the smallest unsatisfiable $(3, s)$ and $(3, p, q)$-formulas for a comprehensive range of values, and brought an improvement in the known minimal size for an unsatisfiable $(4, 5)$-CNF formula. Our work also contributed a uniform proof of the dichotomy for $(3, p, q)$-SAT. The core methodology, a fusion of theoretical insights and an innovative application of the SMS framework with the methods of disjunctive splitting and reductions has not only led to the discovery of new smallest unsatisfiable formulas but also demonstrated the practical utility of SMS in exploring the combinatorial landscape of CNF formulas.

Our findings have illuminated several challenging and open avenues for future work. For instance, extending the scope to identify smallest unsatisfiable formulas for $k \geq 4$ remains a significant challenge. While the methods developed here provide a solid foundation, both novel techniques and theoretical advances are necessary to tackle the increased complexity of larger k values. An extension of our methods may lead to determining the exact value of the threshold $f(5)$, currently only known to be in the interval $[5, 7]$. Moreover, the interplay between the size of unsatisfiable formulas and their implications for inapproximability results in MaxSAT problems [3] invites deeper investigation. Finally, since we found out that, for all $q \geq 3$, the size of the smallest unsatisfiable $(3, 1, q)$-formula coincide with that of the smallest unsatisfiable $(3, 1, q)$-formula in MU(1), we conjecture that this is true for all $k \geq 3$ and $q \geq f(k) + 1$.

References

1 Ron Aharoni and Nathan Linial. Minimal non-two-colorable hypergraphs and minimal unsatisfiable formulas. *J. Combin. Theory Ser. A*, 43:196–204, 1986. doi:10.1016/0097-3165(86)90060-9.

2 Roberto Asín, Robert Nieuwenhuis, Albert Oliveras, and Enric Rodríguez-Carbonell. Cardinality networks and their applications. In Oliver Kullmann, editor, *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*, volume 5584 of *Lecture Notes in Computer Science*, pages 167–180. Springer, 2009. doi:10.1007/978-3-642-02777-2_18.

3 Piotr Berman, Marek Karpinski, and Alex D. Scott. Approximation hardness and satisfiability of bounded occurrence instances of SAT. Technical Report TR03-022, *Electronic Colloquium on Computational Complexity* (ECCC), 2003. URL: https://eccc.weizmann.ac.il/report/2003/022/.

4 Piotr Berman, Marek Karpinski, and Alexander Scott. Approximation hardness of short symmetric instances of max-3sat. Technical Report TR03–049, *Electronic Colloquium on Computational Complexity* (ECCC), 2003. URL: https://eccc.weizmann.ac.il/report/2003/049/.

5 Andreas Darmann and Janosch Döcker. On simplified np-complete variants of monotone3-sat. *Discret. Appl. Math.*, 292:45–58, 2021. doi:10.1016/J.DAM.2020.12.010.

6 M. Davis and H. Putnam. A computing procedure for quantification theory. *J. of the ACM*, 7(3):201–215, 1960. doi:10.1145/321033.321034.

7 Olivier Dubois. On the r, s-SAT satisfiability problem and a conjecture of Tovey. *Discr. Appl. Math.*, 26(1):51–60, 1990. doi:10.1016/0166-218X(90)90020-D.

8 Johannes K. Fichte, Markus Hecher, Daniel Le Berre, and Stefan Szeider. The silent (r)evolution of SAT. *Communications of the ACM*, 66(6):64–72, June 2023. doi:10.1145/3560469.

9 Heidi Gebauer. Disproof of the neighborhood conjecture with implications to SAT. *Combinatorica*, 32(5):573–587, 2012. doi:10.1007/S00493-012-2679-Y.

10 Heidi Gebauer, Tibor Szabó, and Gábor Tardos. The local lemma is asymptotically tight for SAT. *J. of the ACM*, 63(5):Art. 43, 32, 2016. doi:10.1145/2975386.

11 Shlomo Hoory and Stefan Szeider. Computing unsatisfiable k-SAT instances with few occurrences per variable. *Theoretical Computer Science*, 337(1-3):347–359, 2005. doi:10.1016/j.tcs.2005.02.004.

12 Shlomo Hoory and Stefan Szeider. A note on of unsatisfiable k-CNF formulas with few occurrences per variable. *SIAM J. Discrete Math.*, 20(2):523–528, 2006. doi:10.1137/S0895480104445745.

13 Matti Järvisalo, Armin Biere, and Marijn Heule. Blocked clause elimination. In Javier Esparza and Rupak Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, volume 6015 of *Lecture Notes in Computer Science*, pages 129–144. Springer, 2010. doi:10.1007/978-3-642-12002-2_10.

14 Charles Jordan, Will Klieber, and Martina Seidl. Non-cnf QBF solving with QCIR. In Adnan Darwiche, editor, *Beyond NP, Papers from the 2016 AAAI Workshop.*, volume WS-16-05 of *AAAI Workshops*. AAAI Press, 2016. URL: https://aaai.org/papers/aaaiw-ws0186-16-12601/.

15 David Jurenka. Upper bounds for (k, s)-SAT. Bachelor's Thesis, Charles University in Prague, Faculty of Mathematics and Physics, 2011. URL: http://hdl.handle.net/20.500.11956/50583.

16 Markus Kirchweger, Tomáš Peitl, and Stefan Szeider. Co-certificate learning with SAT modulo symmetries. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 1944–1953. ijcai.org, 2023. Main Track. doi:10.24963/IJCAI.2023/216.

17 Markus Kirchweger and Stefan Szeider. SAT modulo symmetries for graph generation. In *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*, LIPIcs, pages 39:1–39:17. Dagstuhl, 2021. doi:10.4230/LIPIcs.CP.2021.34.

18 Markus Kirchweger and Stefan Szeider. SAT modulo symmetries for graph generation and enumeration. *ACM Transactions on Computational Logic*, 2024. Full and extended version of [17], to appear.

19 Hans Kleine Büning and Theodor Lettman. *Propositional logic: deduction and algorithms*. Cambridge University Press, Cambridge, 1999.

20 Donald E. Knuth. *The art of computer programming. Vol. 4B. Combinatorial algorithms. Part 2*. Addison-Wesley, Upper Saddle River, NJ, 2023.

21 Jan Kratochvíl, Petr Savický, and Zsolt Tuza. One more occurrence of variables make satisfiability jump from trivial to NP-complete. *SIAM J. Comput.*, 30:397–403, 1993. doi:10.1137/0222015.

22 O. Kullmann. On a generalization of extended resolution. *Discrete Appl. Math.*, 96–97(1):149–176, October 1999. doi:10.1016/S0166-218X(99)00037-2.

23 João P. Marques-Silva, Inês Lynce, and Sharad Malik. Conflict-driven clause learning SAT solvers. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, pages 131–153. IOS Press, 2009.

24 Hannah Van Santvliet and Ronald de Haan. All instances of monotone 3-sat-(3,1) are satisfiable, 2023. arXiv:2311.06563.

25 P. Savický and Jiří Sgall. DNF tautologies with a limited number of occurrences of every variable. *Theoretical Computer Science*, 238(1-2):495–498, 2000. doi:10.1016/S0304-3975(00)00036-0.

26 J. Stříbrná. Between combinatorics and formal logic. Master's thesis, Charles University, Prague, 1994.

27 Craig A. Tovey. A simplified NP-complete satisfiability problem. *Discr. Appl. Math.*, 8(1):85–89, 1984. doi:10.1016/0166-218X(84)90081-7.

28 Tianwei Zhang, Tomáš Peitl, and Stefan Szeider. Small unsatisfiable k-cnfs with bounded literal occurrence, 2024. arXiv:2405.16149.

A Formulas

In the appendix we list formulas in tabular form, where rows represent variables, columns represent clauses, and the $+$ and $-$ signs indicate positive and negative occurrence, respectively. Blank cells mean the variable does not occur in the clause.

A.1 Smallest enforcers

In the following, we give examples of smallest $(3, s)$-enforcers $E_{3,s}$ and smallest $(3, p, q)$-enforcers $E_{3,p,q}$ for all entries in Table 2. Note that $E_{3,4}$ also acts as a smallest $(3, 2, 3)$-enforcer.

$$E_{3,5} = \begin{pmatrix} - & - & - & - \\ - & - & + & + \\ - & + & - & + \end{pmatrix} \qquad E_{3,4} = \begin{pmatrix} & - & - & \\ - & & - & + & + \\ - & - & + & & + \\ - & + & & - & + \end{pmatrix} \qquad E_{3,1,5} = \begin{pmatrix} - & - & - & - & - \\ - & & & - & + \\ & - & - & & + \\ - & & + & & \\ & - & + & & \end{pmatrix}$$

$$E_{3,1,4} = \begin{pmatrix} - & & & - & + \\ & - & - & - & + \\ - & & & - & - & - \\ - & & & + & & \\ & - & - & + & & \\ & - & + & & & \end{pmatrix} \qquad E_{3,1,3} = \begin{pmatrix} & & & + & + & + \\ - & & & - & & - & + \\ & - & - & & - & & + \\ - & & & - & & + & \\ & - & - & & + & & \\ - & & & + & & & \\ & - & + & & & & \end{pmatrix} \qquad E_{3,2,2} = \begin{pmatrix} - & - & & & & & \\ & - & & & - & & + & + \\ & & & - & & - & + & \\ - & & & - & & + & & + \\ & & & & - & + & - & + \\ & - & - & & + & & + & \\ & & & - & + & - & & + \\ - & - & + & + & & & & \end{pmatrix}$$

A.2 Smallest MU(1) formulas

In the following, we give $\oplus$-derivations of the smallest $(3, s)$-formula $M^1_{3,s}$ and the smallest $(3, p, q)$-formula $M^1_{3,p,q}$ restricted to MU(1) for each entry from Table 6. For all the derivations, we have $F_1 = \{\square\} \oplus_{1,1} \{\square\}$, $F_2 = \{\square\} \oplus_{1,2} F_1$ and $F_{2'} = F_1 \oplus_{2,2} F_1$; other F_i symbols are defined locally in each derivation.

$M^1_{3,4} = F_4 \oplus_{2,2} F_4$	$F_3 = \{\square\} \oplus_{1,3} F_2$	$F_4 = F_3 \oplus_{2,2} F_3$		
$M^1_{3,5} = F_4 \oplus_{3,2} F_5$	$F_3 = \{\square\} \oplus_{1,3} F_2$	$F_4 = F_1 \oplus_{2,3} F_2$	$F_5 = F_2 \oplus_{3,2} F_3$	
$M^1_{3,6} = F_{2'} \oplus_{4,2} F_3$	$F_3 = F_1 \oplus_{2,4} F_{2'}$			
$M^1_{3,7} = F_{2'} \oplus_{4,3} F_3$	$F_3 = F_1 \oplus_{2,3} F_2$			
$M^1_{3,8} = F_{2'} \oplus_{4,4} F_{2'}$				
$M^1_{3,1,3} = F_6 \oplus_{1,3} F_4$	$F_3 = \{\square\} \oplus_{1,3} F_2$	$F_4 = F_3 \oplus_{1,3} F_2$	$F_5 = \{\square\} \oplus_{1,3} F_4$	$F_6 = F_5 \oplus_{1,3} F_4$
$M^1_{3,1,4} = F_5 \oplus_{1,3} F_3$	$F_3 = F_2 \oplus_{1,3} F_2$	$F_4 = \{\square\} \oplus_{1,4} F_3$	$F_5 = F_4 \oplus_{1,3} F_3$	
$M^1_{3,1,5} = F_5 \oplus_{1,5} F_3$	$F_3 = F_1 \oplus_{1,3} F_2$	$F_4 = \{\square\} \oplus_{1,5} F_3$	$F_5 = F_4 \oplus_{1,5} F_3$	
$M^1_{3,2,3} = F_5 \oplus_{2,3} F_4$	$F_3 = \{\square\} \oplus_{1,3} F_2$	$F_4 = F_1 \oplus_{2,3} F_2$	$F_5 = F_3 \oplus_{2,3} F_2$	
$M^1_{3,2,4} = F_3 \oplus_{2,4} F_{2'}$	$F_3 = F_1 \oplus_{2,4} F_{2'}$			
$M^1_{3,3,3} = F_3 \oplus_{3,3} F_3$	$F_3 = F_1 \oplus_{2,3} F_2$			
$M^1_{3,3,4} = F_3 \oplus_{3,4} F_{2'}$	$F_3 = F_1 \oplus_{2,3} F_2$			
$M^1_{3,4,4} = F_{2'} \oplus_{4,4} F_{2'}$				

A.3 Smallest $(3, s)$ and $(3, p, q)$-formulas

In the following, we give examples of smallest $(3, s)$-formulas $M_{3,s}$ and smallest $(3, p, q)$-formulas $M_{3,p,q}$ for all entries in Table 1. We take $M_{3,4} = M^1_{3,4}$ and $M_{3,1,q} = M^1_{3,1,q}$ for all applicable q.

$$M_{3,6} = M_{3,3,3} = \begin{pmatrix} - & & - & - & + & + & + \\ - & - & & + & - & + & + \\ - & - & + & + & & - & + \\ - & + & - & + & - & + & \end{pmatrix} \qquad M_{3,5} = M_{3,2,3} = \begin{pmatrix} & - & & - & & - & + & + \\ - & & - & & - & & + & + \\ - & & & - & & + & - & + \\ - & - & & - & & + & & + \\ - & & & & - & + & + & - \\ & & - & & - & + & & \\ & - & - & + & + & & & \end{pmatrix}$$

$$M_{3,2,4} = \begin{pmatrix} & & & - & & & + \\ & & - & & & + & \\ & - & & & + & & \\ - & & & + & & & \\ - & & - & + & - & & - & + \\ & - & - & & + & - & - & + \\ - & & + & & - & & + & \\ & - & + & & & - & + & \end{pmatrix}$$

$$M_{3,2,2} = \begin{pmatrix} + & & & + & & & - & & & - & & \\ & + & + & & & - & - & & & & & \\ & & & & & & & + & + & & - & - \\ & & & & & & & & & - & + & + & - \\ & & & & & & & & & + & - & + & - \\ & & & & & & & & & & + & - & + & - \\ & & & & & & - & + & + & - & & & \\ & & & & & & + & - & & + & - & & \\ & & & & & & & + & - & + & - & & \\ - & + & + & - & & & & & & & & & \\ + & - & & + & - & & & & & & & & \\ & + & - & + & - & & & & & & & & \end{pmatrix}$$