Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)
Sister Conferences Best Papers Track

Finding the Hardest Formulas for Resolution (Extended Abstract)*

Tomáš Peitl[1†], **Stefan Szeider**[2]
[1]Friedrich Schiller University Jena
[2]TU Wien
tomas.peitl@uni-jena.de, sz@ac.tuwien.ac.at

Abstract

A CNF formula is harder than another CNF formula with the same number of clauses if it requires a longer resolution proof. We introduce *resolution hardness numbers*; they give for $m = 1, 2, \ldots$ the length of a shortest proof of a hardest formula on m clauses. We compute the first ten resolution hardness numbers, along with the corresponding hardest formulas. To achieve this, we devise a candidate filtering and symmetry breaking search scheme for limiting the number of potential candidates for hardest formulas, and an efficient SAT encoding for computing a shortest resolution proof of a given candidate formula.

1 Introduction

Resolution is a fundamental proof system that can be used to certify the unsatisfiability of a propositional formula in conjunctive normal form (CNF) [Davis and Putnam, 1960]. What makes resolution particularly interesting is that the length of a shortest resolution proof of a given CNF formula (called the *resolution complexity* of the formula) provides an unconditional lower bound on the running time of modern SAT solvers [Pipatsrisawat and Darwiche, 2009].[1] Since we know that there are classes of unsatisfiable CNF formulas (such as the formulas based on the Pigeonhole Principle) with exponential resolution complexity [Haken, 1985], we have an exponential lower bound on the running time. It is a natural question to ask: *which formulas are the hardest for resolution?* i.e., which formulas have the highest resolution complexity? This is an intriguing and surprisingly hard question, which has been approached mainly in an asymptotic way, by propositional *proof complexity* [Urquhart, 1995].

In this paper, following a recent trend in tackling combinatorial problems using SAT and CSP methods [Heule *et al.*, 2016; Heule, 2018; Codish *et al.*, 2019], we consider the question of resolution complexity in exact, rather than asymptotic terms. We define the *resolution hardness numbers* $(h_m)_{m=1}^{\infty}$, which, for a given integer m, give the highest resolution complexity of a CNF formula with m clauses. Results from proof complexity, such as Haken's exponential lower bound for the resolution complexity of the pigeonhole formulas [Haken, 1985], translate into statements about the asymptotics of the resolution hardness numbers h_m—in particular, h_m cannot grow polynomially in m. We aim to look beyond asymptotics and compute h_m exactly.

For small values of m, we compute a set of formulas with fewer than m variables and (exactly) m clauses that contain at least one hardest formula for resolution. With these formulas, we can compute the first 10 resolution hardness numbers.

We obtain our results by combining two techniques:

1. A *candidate filtering and symmetry breaking search scheme* to limit the number of potential candidate formulas with m variables and resolution complexity h_m.
2. An *efficient SAT encoding* for computing the resolution complexity of a given candidate formula.

In our search scheme, we reduce the candidate formulas to a certain class of minimally unsatisfiable (MU) formulas that obey additional structural constraints. We model these formulas by suitable graphs, and generate these graphs modulo isomorphisms by an adaptation of the Nauty graph symmetry package [McKay and Piperno, 2014].

This still leaves us with a large number of formulas whose resolution complexity we must determine algorithmically. For this task, we devise an efficient SAT encoding that produces for a given candidate formula F and an integer s a CNF formula $\texttt{short}_s(F)$, which is satisfiable if and only if F admits a resolution proof no longer than s. We determine the resolution complexity of F by feeding $\texttt{short}_s(F)$ to a SAT solver with increasing values of s. While a SAT encoding for this problem has been proposed before [Mencía and Marques-Silva, 2019], and we do take some inspiration from it, we make crucial adaptations tailored towards minimally unsatisfiable formulas. Furthermore, we introduce a symmetry-breaking scheme that fully breaks all symmetries resulting from permuting the clauses in a proof.

In addition to the resolution hardness numbers themselves, we can draw a detailed map of the hardest formulas with a particular number of variables and clauses. Our results re-

*The full version appeared in JAIR and at CP 2020 (preliminary).
†Contact Author
[1]More precisely, resolution complexity provides a lower bound on the running time of *conflict-driven clause learning (CDCL)*, on which modern SAT solvers are based. Supplemental solving techniques may prevail against the resolution lower bound.

veal the significance of *regular saturated minimally unsatisfiable (RSMU)* formulas, which are unsatisfiable formulas that (i) become satisfiable by adding any further literal to any clause, and (ii) where each literal appears in at least two clauses. As a by-product of our computations, we obtain a catalog of RSMU formulas with a small number of variables and clauses, which may be of independent interest in the research on minimal unsatisfiability. For instance, the computed formulas' structure could possibly be used to come up with infinite sequences of hard formulas, inspiring general proof-complexity results.

An alternative but not very interesting object of study would be ν_n, the highest resolution complexity among formulas with n *variables*. It is not hard to see by simple induction on the number of variables that every unsatisfiable formula on n variables has a resolution proof of length $\leq 2^{n+1}-1$ and that indeed $\nu_n = 2^{n+1}-1$, witnessed by the formula which contains all possible clauses of width n.

2 Preliminaries

We assume familiarity with standard notions of graph theory, including those of *(un)directed graphs*, *connectivity*, *acyclicity*, *vertex degrees*, etc. Any graphs we refer to, directed or not, contain no loops or multiedges.

Formulas

We consider propositional formulas in conjunctive normal form (CNF) represented as sets of clauses. We assume an infinite set *var* of (propositional) *variables*. A *literal* ℓ is a variable x or a negated variable $\neg x$; we write $\mathit{lit} := \{x, \neg x \mid x \in \mathit{var}\}$. For a literal ℓ we put $\overline{\ell} := \neg x$ if $\ell = x$, and $\overline{\ell} := x$ if $\ell = \neg x$. For a set of C literals we put $\overline{C} := \{\overline{\ell} \mid \ell \in C\}$. C is *tautological* if $C \cap \overline{C} \neq \emptyset$. A finite non-tautological set of literals is a *clause*; a finite set of clauses is a (CNF) *formula*. The empty clause is denoted by $\Box$. We write $\mathrm{CNF}(n,m)$ for the class of all CNF formulas on n variables and m clauses, and $\mathrm{CNF}(m) = \bigcup_{n=0}^{\infty} \mathrm{CNF}(n,m)$. For a clause C, we put $\mathit{var}(C) = \{\mathit{var}(\ell) \mid \ell \in C\}$, and for a formula F, $\mathit{var}(F) = \bigcup_{C \in F} \mathit{var}(C)$. Similarly, we put $\mathit{lit}(F) := \mathit{var}(F) \cup \overline{\mathit{var}(F)}$. An *assignment* is a mapping $\tau : \mathit{var}(F) \to \{0,1\}$. A formula F is *satisfiable* if there is a *satisfying assignment*, i.e., a mapping $\tau : \mathit{var}(F) \to \{0,1\}$ such that every clause of F contains either a literal x with $\tau(x) = 1$ or a literal $\neg x$ with $\tau(x) = 0$; otherwise it is *unsatisfiable*. A formula is *minimally unsatisfiable (MU)* if it is unsatisfiable, but every proper subset is satisfiable.

Resolution

If $C_1 \cap \overline{C_2} = \{\ell\}$ for clauses C_1, C_2 and a literal ℓ, then the *resolution rule* allows the derivation of the clause $D = (C_1 \cup C_2) \setminus \{\ell, \overline{\ell}\}$; D is the *resolvent* of the *premises* C_1 and C_2, and we say that D is obtained by *resolving on* ℓ. Let F be a formula and C a clause. A sequence $P = L_1, \ldots, L_s$ of clauses (proof *lines*) is a *resolution derivation of* L_s *from* F if for each $i \in \{1, \ldots, s\}$ at least one of the following holds.

1. $L_i \in F$ ("L_i is an axiom");
2. L_i is the resolvent of L_j and $L_{j'}$ for some $1 \leq j, j' < i$.

We write $|P| := s$ and call s the *length* of P. If L_s is the empty clause, then P is a *resolution refutation* or *resolution proof* of F. A line L_i in a resolution derivation may have different possible "histories;" i.e., L_i may be the resolvent of more than one pair of clauses preceding L_i, or L_i may be both an axiom and obtained from preceding clauses by resolution, etc. In the sequel, however, we assume that an arbitrary but fixed history is associated with each considered resolution derivation. Thus, with a proof P we can associate the directed acyclic graph (DAG) $G(P)$ whose vertices are the proof lines, and which has an arc from L_i to L_j if there is L_k, $i, k < j$, such that L_j is the resolvent of L_i and L_k.

It is well known that resolution is a complete proof system for unsatisfiable formulas; i.e., a formula F is unsatisfiable if and only if there exists a resolution refutation of it. The *resolution complexity* or *resolution hardness* $h(F)$ of an unsatisfiable formula F is the length of a shortest resolution refutation of F. For a nonempty set $\mathcal{C}$ of formulas, we put $h(\mathcal{C}) = \max_{F \in \mathcal{C}} h(F)$.

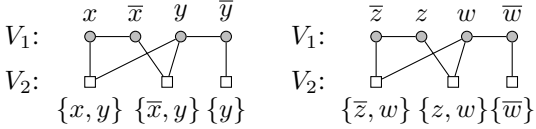
Isomorphisms of Formulas

Two formulas F and F' are *isomorphic* if there exists a bijection $\varphi : \mathit{lit}(F) \to \mathit{lit}(F')$ such that for each literal $\ell \in \mathit{lit}(F)$ we have $\overline{\varphi(\ell)} = \varphi(\overline{\ell})$ and for each $C \subseteq \mathit{lit}(F)$ we have $C \in F$ if and only if $\varphi(C) \in F'$. For instance the formulas $F = \{\{x, \overline{y}\}, \{\overline{x}, y\}, \{\overline{y}\}\}$, and $F' = \{\{\overline{z}, w\}, \{z, w\}, \{\overline{w}\}\}$ are isomorphic.

A *2-graph* is an undirected graph $G = (V, E)$ together with a partition of its vertex set into two subsets $V = V_1 \uplus V_2$. 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 $\varphi : V_1 \uplus V_2 \to V_1' \uplus V_2'$ such that $v \in V_i$ if and only if $\varphi(v) \in V_i'$, $i = 1, 2$, and $\{u, v\} \in E$ if and only if $\{\varphi(u), \varphi(v)\} \in E'$.

The *clause-literal graph* of a formula F is the 2-graph $G(F) = (V_1 \uplus V_2, E)$ with $V_1 = \mathit{lit}(F)$, $V_2 = F$, and $E = \{\{x, \overline{x}\} \mid x \in \mathit{var}(F)\} \cup \{\{C, \ell\} \mid C \in F, \ell \in C\}$. We refer to the edges $\{x, \overline{x}\}$ as *variable* edges.

For instance, the formulas F and F' mentioned above give rise to the following two isomorphic clause-literal graphs.



Proposition 1. *Two formulas are isomorphic if and only if their clause-literal graphs are isomorphic (as 2-graphs).*

3 Theoretical Framework

We define the *m-th resolution hardness number* as the highest resolution complexity among formulas with m clauses:

$$h_m = \max_{F \in \mathrm{CNF}(m)} h(F) = h(\mathrm{CNF}(m)).$$

We also let $H(m) := \{F \in \mathrm{CNF}(m) : h(F) = h_m\}$.

In this section, we describe on a high level our approach to computing h_m. Due to space constraints, we refer the reader to the full version for proofs of the results [Peitl and Szeider, 2020; Peitl and Szeider, 2021a].

We begin by establishing the unsurprising basic fact that resolution hardness numbers form an increasing sequence.

Lemma 1. $h_{m+1} \geq h_m + 2$.

Our strategy for computing h_m is to find and automatically generate a set $\chi(m) \subseteq \text{CNF}(m)$ with the following desired properties:

1. $\chi(m)$ contains at least one formula with hardness h_m;
2. $\chi(m)$ is as small and easy to enumerate as possible.

Once we find a suitable $\chi(m)$, we simply generate all its formulas and for each compute its hardness; the largest value gives the hardness number h_m.

3.1 Finding $\chi(m)$

The central role is played by *saturated minimally unsatisfiable (SMU)* formulas—minimally unsatisfiable formulas that are rendered satisfiable by adding any literal to any clause. Lemma 2 tells us that $\chi(m)$ can be limited to SMU formulas.

Lemma 2. *All formulas in $H(m)$ are minimally unsatisfiable; moreover, at least one is saturated.*

A simple lower bound for the hardness of minimally unsatisfiable formulas follows from the need to 'connect' all axioms in the proof, reading each clause at least once.

Lemma 3. *Let F be a minimally unsatisfiable formula with m clauses. Then $h(F) \geq 2m - 1$.*

SMU formulas have two key properties that make them suitable for our purpose: they are the hardest of MU formulas; and they have a well studied structure we can exploit.

In particular, SMU allows us to treat singular and regular formulas separately. A formula is called *singular* if it contains a *singular literal*—one that only appears once—and *regular* otherwise. Singular SMU (SSMU) formulas are well behaved with respect to *Davis-Putnam (DP) reduction* (variable elimination by exhaustive resolution, [Kullmann and Zhao, 2013, Lemma 12]), which allows us to obtain all SSMU formulas by a simple reverse process, which we term *DP-lifting*, from smaller SMU formulas. Thanks to that, we can split the generation phase in two, generating only regular formulas first. In the first, expensive phase we get a speedup from pruning via regularity—DP-lifting for SSMU formulas afterwards turns out to be cheap.

Another important notion in the context of minimal unsatisfiability is the *deficiency* $\delta(F)$ of a formula F, defined as $\delta(F) = |F| - |var(F)|$. By a lemma attributed to Tarsi [Aharoni and Linial, 1986], $\delta(F)$ is positive if F is MU.

Lemma 4 ([Aharoni and Linial, 1986])**.** *Let F be a minimally unsatisfiable formula. Then* $\log_2 |F| \leq |var(F)| < |F|$.

We denote by $\text{MU}(n,m)(\text{SMU}(n,m))$ the class of (saturated) minimally unsatisfiable formulas with n variables and m clauses; $\text{RSMU}(n,m)$ ($\text{SSMU}(n,m)$) is the subclass of $\text{SMU}(n,m)$ consisting of regular (singular) formulas.

The structure of SMU formulas of deficiencies 1 and 2 is well understood [Kleine Büning and Kullmann, 2009]. It is known that each deficiency-1 SMU formula with at least one variable has a 1-*singular* literal (one whose negation is also a singular literal). It is also known that, up to isomorphism, there is a unique deficiency-2 regular MU formula—we pick $\mathcal{F}_m^2$, which consists of the clauses

$$\{\overline{x_1}, x_2\}, \ldots, \{\overline{x_{m-3}}, x_{m-2}\}, \{\overline{x_{m-2}}, x_1\},$$
$$\{x_1, \ldots, x_{m-2}\}, \{\overline{x_1}, \ldots, \overline{x_{m-2}}\}.$$

This allows us to obtain resolution hardness of $\text{SMU}(m-1,m)$ and $\text{RSMU}(m-2,m)$ formulas in closed form.

Proposition 2. *For $m \geq 1$, $h(\text{SMU}(m-1,m)) = 2m-1$.*

Proposition 3. *For $m \geq 4$, $h(\mathcal{F}_m^2) = 3m-5$.*

Propositions 2 and 3, together with Lemma 4, give us a refined lower bound for h_m.

Corollary 1. *For $m \leq 3$, $h_m = 2m-1$. For $m \geq 4$, $h_m \geq 3m-5$.*

In order to determine h_m for $m \geq 4$, we will need to generate $\chi(m)$, which, according to the results above, can be restricted to $\text{SMU}(n,m)$ formulas for $n \leq m-2$, ignoring deficiency 1, whose hardness we know. We can further facilitate this step by only generating regular formulas, which we need only for $n \leq m-3$ because we know $\text{RSMU}(m-2,m)$; we get singular formulas afterwards by DP-lifting.

Naturally, we do not need to include two different isomorphic formulas in $\chi(m)$; their hardness properties are identical. Hence, we want to enumerate an isomorph-free subset of $\chi(m)$, which we achieve through the correspondence to 2-graphs (Proposition 1) and a tailor-made adaptation of the `genbg` utility from the graph symmetry package Nauty [McKay and Piperno, 2014] to enumerate such graphs. Nauty allows us to specify constraints on the output graphs, such as connectedness (necessary for MU), triangle-freeness (triangles are tautological clauses), neighbourhood incomparability (no subsumed clauses), or degree bounds (regularity), which can enforce regularity and some necessary conditions for minimal unsatisfiability. However, it is not possible to directly instruct Nauty to generate only SMU formulas, and so we need to generate a superset and filter for saturated minimal unsatisfiability. For that purpose, we proposed a streamlined brute-force SMU testing algorithm, which we omit here due to space constraints.

3.2 Computing Hardness

To compute shortest proofs of formulas, we devise a SAT encoding to answer the following decision problem, which we call $\texttt{SHORT}(F,s)$.

Given a formula F with the clauses (*axioms*) $A_1, \ldots, A_m$ and $var(F) = \{x_1, \ldots, x_n\}$, does there exist a resolution refutation $L_1, \ldots, L_{s'}$ of F of length $s' \leq s$?

It is easy to see that $\texttt{SHORT}(F,s)$ is coNP-hard (s given in binary): since each unsatisfiable formula F with n variables has a resolution refutation of length at most $2^{n+1}-1$, we have $\texttt{UNSAT}(F) = \texttt{SHORT}(F, 2^{n+1}-1)$; using a SAT-based approach is thus justified. On the other hand, membership in NEXPTIME is trivial: guess a refutation of length s and verify that it is correct. The precise complexity of $\texttt{SHORT}(F,s)$ is an open problem.

The basic idea of our encoding, called $\texttt{short}_s(F)$, is to have variables $\text{pos}[i,v]$ and $\text{neg}[i,v]$ that determine whether v and $\overline{v}$ occur in L_i, and variables $\text{arc}[i,j]$, which describe the

proof DAG $G(P)$, together fully determining a candidate resolution proof sequence P. We additionally use auxiliary variables to express certain constraints more succinctly.

We draw inspiration from a similar recently proposed encoding (MSM, [Mencía and Marques-Silva, 2019]), but make improvements focused on MU formulas. One of the strongest points of MSM, enumerating *minimal correction subsets* (MCSes, i.e., inclusion-minimal sets of clauses whose removal makes the formula satisfiable) in preprocessing, becomes trivial for MU formulas: the MCSes are by definition precisely all singletons. Instead, we require that all axioms are used in the proof.

On the other hand, we extend the encoding with symmetry breaking, inspired by an idea of Schidler and Szeider [2020]. The source of symmetries we aim to break are different topological sorts of a proof DAG $G(P)$. We define a *canonical topological sort*, and introduce symmetry breaking constraints to ensure that no other topological sort is accepted. The canonical sort is obtained by the topological sorting procedure where in each step the largest available vertex under an arbitrary fixed total order of vertices is picked—in our case a lexicographic ordering of clauses. Another novelty of our encoding is the capacity to reject a partially constructed proof early based on a counting argument involving the number of times each clause is used in resolution steps, similar to Lemma 3. Space constraints prevent us from explaining these ideas in detail here—we refer the reader to the full version. Here we limit ourselves to a summary in Theorem 1.

Theorem 1. *Let F be a formula on n variables and m clauses, s an integer. It holds that*

1. *the size of $\texttt{short}_s(F)$ is polynomial in $\max(n, m, s)$ (s can be exponential in the input length);*
2. *$\texttt{short}_s(F)$ is satisfiable if and only if F has a resolution refutation P of length s; any model can naturally be interpreted as a canonically topologically sorted resolution refutation of F;*

Theorem 1 gives rise to a simple algorithm. Start with $s = 1$ and increment while $\texttt{short}_s(F)$ is unsatisfiable. When $\texttt{short}_s(F)$ becomes satisfiable, s equals $h(F)$, and a shortest proof can be extracted from a model of $\texttt{short}_s(F)$. For MU formulas, we can start at $s = 2m - 1$, as per Lemma 3. We implemented the encoding and this algorithm in Python in the PySAT framework [Ignatiev *et al.*, 2018].

4 Results

Finally, we review our most important results. See the full version [Peitl and Szeider, 2021a] for a much deeper analysis.

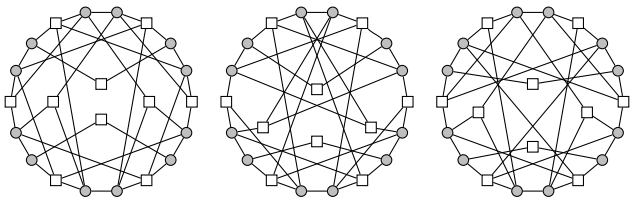
Table 1 lists the maximum hardness of an SMU(n, m) formula, and, by taking the maximum in each column (bold), also values of h_m. In particular, we obtain the first ten resolution hardness numbers:

$$(h_m)_{m=1}^{\infty} = 1, 3, 5, 7, 10, 13, 16, 19, 22, 26, \ldots$$

It is known that every MU(n, m) formula F has a proof of length at most $2^{\delta(F)-1}n + m$ [Kleine Büning and Kullmann, 2009, Section 11.3]; along with the existence of formulas with super-polynomial hardness, this implies that maximum hardness cannot forever be attained by formulas of bounded deficiency. We discover $m = 10$ to be the first point where no formula of deficiency 2 is among the hardest: up to isomorphism, there are exactly three hardest 10-clause SMU formulas, all of deficiency 4, shown as clause-literal graphs in Figure 1. Notice the graphs are drawn vertically symmetrically. Apart from $\mathcal{F}_m^2$, these are the only hardest formulas that can be drawn in such a way—symmetrically and with a cycle containing all variable edges drawn as a circle. We calculated these drawings (and non-existence for other graphs) with MiniZinc [Nethercote *et al.*, 2007; Stuckey *et al.*, 2014].

$n\vert^{m}_{\rightarrow}$	1	2	3	4	5	6	7	8	9	10
0	**1**$^{(1)}$	-	-	-	-	-	-	-	-	-
1	-	**3**$^{(1)}$	-	-	-	-	-	-	-	-
2	-	-	**5**$^{(1)}$	**7**$^{(1)}$	-	-	-	-	-	-
3	-	-	-	**7**$^{(2)}$	**10**$^{(1)}$	11$^{(3)}$	13$^{(1)}$	15$^{(1)}$	-	-
4	-	-	-	-	9$^{(3)}$	**13**$^{(1)}$	15$^{(1)}$	**19**$^{(1)}$	20$^{(1)}$	21$^{(5)}$
5	-	-	-	-	-	11$^{(6)}$	**16**$^{(1)}$	18$^{(3)}$	**22**$^{(1)}$	25$^{(1)}$
6	-	-	-	-	-	-	13$^{(11)}$	**19**$^{(1)}$	**22**$^{(3)}$	**26**$^{(3)}$
7	-	-	-	-	-	-	-	15$^{(23)}$	**22**$^{(1)}$	25$^{(24)}$
8	-	-	-	-	-	-	-	-	17$^{(46)}$	25$^{(1)}$
9	-	-	-	-	-	-	-	-	-	19$^{(98)}$

Table 1: Values of $h(\text{SMU}(n, m))$, and in parentheses the number of hardest formulas in SMU(n, m). For all $3 \leq n \leq 9$ and $n + 2 \leq m \leq 10$, we found that all hardest SMU(n, m) formulas are regular, except for SMU$(7, 10)$, which contains 5 regular and 19 singular hardest formulas. Formulas are counted modulo isomorphism.



$h(F_{6,10,1}) = 26$ $h(F_{6,10,2}) = 26$ $h(F_{6,10,3}) = 26$

Figure 1: Clause-literal graphs of the hardest 10-clause formulas.

Our encoding [Peitl and Szeider, 2021c] and our catalog of SMU formulas [Peitl and Szeider, 2021b], including the data from our computation, are publicly available.

5 Conclusion

We conducted an extensive computational investigation into resolution hardness, computing the first ten resolution hardness numbers along with a catalog of the corresponding hardest formulas. Our results indicate that regular saturated minimally unsatisfiable formulas achieve the highest hardness. It remains as an interesting theoretical question whether the hardest formulas are always regular.

Acknowledgments

We thank Brendan McKay for adapting Nauty `genbg` for us.

We further acknowledge support by the FWF (projects P32441 and J-4361) and by the WWTF (project ICT19-065).

References

[Aharoni and Linial, 1986] Ron Aharoni and Nathan Linial. Minimal non-two-colorable hypergraphs and minimal unsatisfiable formulas. *J. Combin. Theory Ser. A*, 43:196–204, 1986.

[Codish *et al.*, 2019] Michael Codish, Alice Miller, Patrick Prosser, and Peter J. Stuckey. Constraints for symmetry breaking in graph representation. *Constraints An Int. J.*, 24(1):1–24, 2019.

[Davis and Putnam, 1960] M. Davis and H. Putnam. A computing procedure for quantification theory. *J. of the ACM*, 7(3):201–215, 1960.

[Haken, 1985] Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.

[Heule *et al.*, 2016] Marijn J. H. Heule, Oliver Kullmann, and Victor W. Marek. Solving and verifying the Boolean Pythagorean Triples problem via cube-and-conquer. In Nadia Creignou and Daniel Le Berre, editors, *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings*, volume 9710 of *Lecture Notes in Computer Science*, pages 228–245. Springer Verlag, 2016.

[Heule, 2018] Marijn J. H. Heule. Schur number five. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 6598–6606, 2018.

[Ignatiev *et al.*, 2018] Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. PySAT: A Python toolkit for prototyping with SAT oracles. In *SAT*, pages 428–437, 2018.

[Kleine Büning and Kullmann, 2009] Hans Kleine Büning and Oliver Kullmann. Minimal unsatisfiability and autarkies. In A. Biere, M. J. H. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 11, pages 339–401. IOS Press, 2009.

[Kullmann and Zhao, 2013] Oliver Kullmann and Xishun Zhao. On Davis-Putnam reductions for minimally unsatisfiable clause-sets. *Theoretical Computer Science*, 492:70–87, 2013.

[McKay and Piperno, 2014] Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, {II}. *Journal of Symbolic Computation*, 60(0):94 – 112, 2014.

[Mencía and Marques-Silva, 2019] Carlos Mencía and João Marques-Silva. Computing shortest resolution proofs. In Paulo Moura Oliveira, Paulo Novais, and Luís Paulo Reis, editors, *Progress in Artificial Intelligence, 19th EPIA Conference on Artificial Intelligence, EPIA 2019, Vila Real, Portugal, September 3-6, 2019, Proceedings, Part II*, volume 11805 of *Lecture Notes in Computer Science*, pages 539–551. Springer, 2019.

[Nethercote *et al.*, 2007] Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack. Minizinc: Towards a standard cp modelling language. In Christian Bessière, editor, *Principles and Practice of Constraint Programming – CP 2007*, pages 529–543, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[Peitl and Szeider, 2020] Tomáš Peitl and Stefan Szeider. Finding the hardest formulas for resolution. In Helmut Simonis, editor, *Proceedings of CP 2020, the 26th International Conference on Principles and Practice of Constraint Programming*, volume 12333 of *Lecture Notes in Computer Science*, pages 514–530. Springer Verlag, 2020. Best Paper Award.

[Peitl and Szeider, 2021a] Tomáš Peitl and Stefan Szeider. Finding the hardest formulas for resolution. *Journal of Artificial Intelligence Research*, 2021.

[Peitl and Szeider, 2021b] Tomáš Peitl and Stefan Szeider. Saturated minimally unsatisfiable formulas on up to ten clauses. https://doi.org/10.5281/zenodo.3951545, 2021. Accessed: 2021-01-14.

[Peitl and Szeider, 2021c] Tomáš Peitl and Stefan Szeider. short.py: Encoding for the shortest resolution proof. https://doi.org/10.5281/zenodo.3951549, 2021. Accessed: 2021-01-13.

[Pipatsrisawat and Darwiche, 2009] Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning SAT solvers with restarts. In Ian P. Gent, editor, *Principles and Practice of Constraint Programming - CP 2009, 15th International Conference, CP 2009, Lisbon, Portugal, September 20-24, 2009, Proceedings*, volume 5732 of *Lecture Notes in Computer Science*, pages 654–668. Springer Verlag, 2009.

[Schidler and Szeider, 2020] André Schidler and Stefan Szeider. Computing optimal hypertree decompositions. In Guy Blelloch and Irene Finocchi, editors, *Proceedings of ALENEX 2020, the 22nd Workshop on Algorithm Engineering and Experiments*, pages 1–11. SIAM, 2020.

[Stuckey *et al.*, 2014] Peter Stuckey, Thibaut Feydy, Andreas Schutt, Guido Tack, and Julien Fischer. The minizinc challenge 2008-2013. *AI Magazine*, 35:55–60, 06 2014.

[Urquhart, 1995] Alasdair Urquhart. The complexity of propositional proofs. *Bull. of Symbolic Logic*, 1(4):425–467, December 1995.