Hardness of Random Reordered Encodings of Parity for Resolution and CDCL

Leroy Chew[1], Alexis de Colnet[1], Friedrich Slivovsky[2], Stefan Szeider[1]

[1] Algorithms and Complexity Group, TU Wien, Vienna, Austria
[2] Department of Computer Science, University of Liverpool, Liverpool, UK
{lchew,decolnet,sz}@ac.tuwien.ac.at, f.slivovsky@liverpool.ac.uk

Abstract

Parity reasoning is challenging for Conflict-Driven Clause Learning (CDCL) SAT solvers. This has been observed even for simple formulas encoding two contradictory parity constraints with different variable orders (Chew and Heule 2020). We provide an analytical explanation for their hardness by showing that they require exponential resolution refutations with high probability when the variable order is chosen at random. We obtain this result by proving that these formulas, which are known to be Tseitin formulas, have Tseitin graphs of linear treewidth with high probability. Since such Tseitin formulas require exponential resolution proofs, our result follows. We generalize this argument to a new class of formulas that capture a basic form of parity reasoning involving a sum of two random parity constraints with random orders. Even when the variable order for the sum is chosen favorably, these formulas remain hard for resolution. In contrast, we prove that they have short DRAT refutations. We show experimentally that the running time of CDCL SAT solvers on both classes of formulas grows exponentially with their treewidth.

Introduction

SAT solvers, including Conflict-Driven Clause-Learning (CDCL) solvers, can solve practical problems with millions of variables (Marques-Silva, Lynce, and Malik 2009; Fichte et al. 2023), but on the other hand, can struggle with basic mathematical principles. The Handbook of Satisfiability (Biere et al. 2021, Section 9.6.1) lists one such example: the problem of XOR (exclusive-or) constraints, which is equivalent to the parity problem of summation modulo 2. XOR-constraints serve practical purposes, particularly around modern cryptographical and cryptonanalytical problems. Provably hard XOR problems are usually constructed over complex structures such as expander graphs (Urquhart 1987; Ben-Sasson and Wigderson 2001), but much simpler problems involving only two constraints were found experimentally hard for CDCL (Chew and Heule 2020) and, up until this paper, were not matched with a corresponding lower bound in resolution.

Resolution is particularly important here because the relationship with CDCL solving is two-way; CDCL runs of unsatisfiable instances can be output as resolution proofs, but also every resolution refutation can be followed completely by a CDCL algorithm (with a few non-deterministic choices) to return UNSAT (Pipatsrisawat and Darwiche 2011). Lower bounds on the length of resolution proof of unsatisfiability have been shown for pure XOR problems structured by graphs and represented in CNF formulas called Tseitin formulas (Tseitin 1983; Urquhart 1987). Finding a complete characterization of hard Tseitin formulas for resolution is still an open problem. However, exponential lower bounds for resolution are known under the suitable condition: that their underlying graphs have high linear *treewidth* (a graph invariant that measures how close a graph is to being a tree). The relationship between treewidth and proof length has been extensively studied for Tseitin formulas already (Ben-Sasson and Wigderson 2001; Galesi, Talebanfard, and Torán 2020; Itsykson, Riazanov, and Smirnov 2022).

Here, we look at some XOR-constraint problems that are strikingly simple to define and whose hardness for resolution was observed empirically but yet to be understood theoretically (Chew and Heule 2020). We prove that they are, in fact, families of Tseitin formulas and that linear treewidth emerges for almost all of them, thus showing asymptotic exponential lower bounds for resolution. Furthermore, our experiments suggest that this is not just theoretical and asymptotic for proof systems; treewidth indeed correlates to the solving time of CDCL solvers on these families. In the rest of this section, we present the problems and our results.

Problem 1: Reordered Parity

The standard linear CNF encoding of an XOR-constraint over n propositional variables splits the constraint into a sequence of XORs of size 3 according to an ordering of its variables (Biere et al. 2021, Section 2.2.5). The encoding uses one auxiliary variable for every $k < n$ to store the parity of the first k variables. The simplest form of the XOR-constraint problem starts with two opposite XOR-constraints and their standard linear CNF encodings where the n variables appear in a different order given by the permutation σ. The two CNF are saying the sum of the n variables is both odd and even, so their conjunction—denoted by $\mathsf{rPar}(n, \sigma)$—is a contradiction a SAT solver should be able to recognize.

Chew and Heule (2020) showed that these problems could be proven false by $O(n \log n)$-size DRAT proofs even without new variables (Buss and Thapen 2019). We show here that resolution proofs on their own are often unable to handle even these restricted examples. For some σ, such as the identity mapping, resolution proofs are short; in fact, the identity mapping gives the `Dubois` family in the SAT library. However, the easiness is not seen with other permutations. Chew and Heule (2020) conducted experiments that showed that CDCL solvers struggle and time-out around $n = 50$ for uniformly selected permutations, although a theoretical lower bound was never proved. We show that as n increases, a random permutation σ yields, with high probability, a formula $\mathsf{rPar}(n, \sigma)$ whose resolution proof requires exponentially many clauses.

Theorem 1. *There is a constant $\alpha > 0$ such that, with probability tending to 1 as n increases, the length of a smallest resolution refutation of the unsatisfiable formula $\mathsf{rPar}(n, \sigma)$, where σ is chosen uniformly at random, is at least $2^{\alpha n}$.*

A key observation here is that the rPar formulas are Tseitin formulas. The fact that the rPar formulas come from standard CNF encodings of very simple XOR-constraints problems makes them natural examples of Tseitin formulas that are more likely to occur in practice than those that appear in proofs of hardness that are often constructed from arbitrary expander graphs (Urquhart 1987). Theorem 1 proves the context assumed by Chew and Heule (2020) that a powerful proof system such as DRAT$^-$ was necessary for the short proofs of rPar, as we now have exponential resolution lower bounds. Along with the evidence shown by our experiments, we now conclude that high treewidth is the reason for the hardness in the experiments of Chew and Heule (2020). We can also take this as clarification that order matters in the encoding of parity constraints in general.

Problem 2: Random Parity Addition

There are several effective strategies for dealing with XOR-constraints in practice. One method that has succeeded is to employ Gaussian elimination (Han and Jiang 2012; Soos 2012) techniques to simplify the problem. Two contradictory parity constraints fall short of representing what happens in an average step of Gaussian elimination. Instead, Gaussian elimination involves many steps using the bitwise addition of two XOR-constraints to produce a new constraint. In order to study such a step as an instance for resolution, we have to write it as a contradiction. So here we modify rPar to use three XOR-constraints, with the third containing the variables in the symmetric difference of the first two constraints and then flip some literals to create a contradiction. The input XOR-constraints are encoded in CNF formulas a and b using the standard linear encoding. We then define a CNF encoding $\mathsf{rAddPar}(a, b)$ of the contradiction similar to $\mathsf{rPar}(n, \sigma)$, and we show their hardness for resolution.

Theorem 2. *With high probability, for any two random parity constraints over n variables encoded randomly and independently in CNF formulas a and b using the standard linear encoding, the length of the shortest resolution refutation of $\mathsf{rAddPar}(a, b)$ is exponential in n, the length of a shortest resolution refutation of $\mathsf{rAddPar}(a, b)$ is exponential in the number of variables.*

The rAddPar formulas turn out to also be Tseitin formulas, so this again provides a new intuitive family that demonstrates the hardness of Tseitin formulas—and yet again shows that order matters when encoding parity constraints.

Adding Gaussian elimination to SAT-solving or preprocessing presents several technical challenges. An example is verification—unsatisfiable instances in CDCL SAT solvers can be readily verified in resolution proofs and thus verified in the more powerful checking format standard DRAT (Järvisalo, Heule, and Biere 2012). It was therefore pertinent to show that Gaussian elimination techniques could also be verified efficiently in DRAT (Philipp and Rebola-Pardo 2016). For the specific family of $\mathsf{rPar}(n, \sigma)$, it was shown to have DRAT$^-$ refutations in $O(n \log n)$ many lines using a tool from Chew and Heule (2020). Recently, a BDD-based SAT solver augmented with pseudo-Boolean constraints (Bryant, Biere, and Heule 2022) was shown to have improved the result experimentally. We can generalize Chew and Heule's upper-bound results to $\mathsf{rAddPar}(a, b)$.

Theorem 3. *For any two random parity constraints over n variables encoded randomly and independently in CNF formulas a and b using the standard linear encoding, there are DRAT$^-$ refutations of $\mathsf{rAddPar}(a, b)$ with $O(n \log n)$ many lines.*

$O(n \log n)$ is already a good upper bound, and this can potentially be used in verification. One of the advantages of DRAT$^-$ is that no extension variables are added.

Preliminaries

Boolean *variables* take value in $\{0, 1\}$. A *literal* is either a variable x or its negation $\bar{x}$. *Clauses* are disjunctions of literals and *CNF formulas* are conjunctions of clauses. The negation of clause C can be labelled $\bar{C}$ and is a CNF of clauses each containing one literal. The symbols $\vee$, $\wedge$ denote disjunction and conjunction and we use $\oplus$ for exclusive disjunction, that is, $x \oplus y = x + y \bmod 2$. The *canonical CNF representation* of a *parity constraint* $x_1 \oplus \cdots \oplus x_k = 0$ is the CNF formula $\mathsf{xor}(x_1, \dots, x_k)$ composed of all 2^{k-1} clauses of size k that contain an odd (resp. even) number of positive literals when k is odd (resp. even). For instance $\mathsf{xor}(p, q, r) := (\bar{p} \vee \bar{q} \vee \bar{r}) \wedge (\bar{p} \vee q \vee r) \wedge (p \vee \bar{q} \vee r) \wedge (p \vee q \vee \bar{r})$. The canonical representation of $x_1 \oplus \cdots \oplus x_k = 1$ is just $\mathsf{xor}(x_1, \dots, x_k)$ where we flip all literals for an arbitrary variable, for instance $\mathsf{xor}(\bar{x}_1, x_2, \dots, x_k)$.

Proofs and Refutations

Resolution. *Resolution* is a refutational proof system that works by adding clauses based on a single binary rule—the resolution rule (Robinson 1963). The resolution rule's new clause is a logical implication. Adding it to the formula preserves not only satisfiability but also the models. A resolution proof that derives the empty clause shows that the original formula is unsatisfiable.

$$\frac{C_1 \vee x \qquad C_2 \vee \neg x}{C_1 \vee C_2} \text{ (Resolution)}$$

Extended Resolution. *Extended resolution* adds an extension rule, it creates extension clauses that introduce a new variable with clauses that force that new variables to follow a definition. If we treat the extension variables as new, the extension rule does not change the satisfying assignments when considering only the original variables. Therefore, when we reach the empty clause, we know that the original formula must have been unsatisfiable.

Example 1. *We can add the following extension clauses that state that extension variable n is the exclusive or of x with y: $(\bar{x} \vee y \vee n), (x \vee \bar{y} \vee n), (x \vee y \vee \bar{n}), (\bar{x} \vee \bar{y} \vee \bar{n})$.*

DRAT. Unit propagation is an incomplete, model preserving, and polynomial-time process.

Definition 1. *A unit clause is a clause of one literal.* Unit propagation *takes any unit clause (a) and resolves it with every clause which has an $\bar{a}$ in it (possibly creating another unit clause). After all resolvents are found, the clause (a) is removed, and we repeat the process for another unit clause until no unit clauses remain. We also terminate if we reach the empty clause, and we can write $F \vdash_1 \bot$ to denote that unit propagation of F reaches the empty clause.*

While unit propagation itself is incomplete, it terminates in polynomial time. It therefore is a convenient tool for checking implication, we can use this in the concept of an asymmetric tautology, which is a clause that must be true assuming a CNF because its negation would cause a conflict via unit propagation.

Definition 2 (Järvisalo, Heule, and Biere 2012)**.** *Let F be a CNF formula. A clause C is an* asymmetric tautology *(AT) w.r.t. F if $F \wedge \bar{C} \vdash_1 \bot$.*

A clause being an asymmetric tautology in F is a generalization of being a resolvent of some pair of clauses in F. We also want to be able to generalize the creation of extension clauses. To do this, we first generalize extension clauses to blocked clauses. Blocked clauses are clauses that have a literal that cannot be resolved without causing a tautology, and so they are non-threatening to the satisfiability of the formula. We generalize blocked clauses to RAT clauses, where we widen the condition of tautology to asymmetric tautology.

Definition 3 (Järvisalo, Heule, and Biere 2012)**.** *Let F be a CNF formula. A clause C is a* resolution asymmetry tautology *(RAT) w.r.t. F if there exists a literal $l \in C$ such that for every clause $\bar{l} \vee D \in F$ it holds that $F \wedge \bar{D} \wedge \bar{C} \vdash_1 \bot$.*

DRAT is a generalized and application friendly version of extended resolution. Each rule modifies a formula by either adding (removing) a clause while preserving satisfiability (unsatisfiability), respectively. Unlike resolution, clauses can be added (removed) with or without preserving the exact set of satisfying models of a formula. The first set of DRAT rules show us how we can add or remove clauses while preserving models by using asymmetric tautologies when C is AT w.r.t. F:

$$\frac{F}{F \wedge C} \text{ (ATA)} \qquad \frac{F \wedge C}{F} \text{ (ATE)}$$

The second set of rules use resolution asymmetric tautologies (C is RAT w.r.t. F) and do not preserve models:

$$\frac{F}{F \wedge C} \text{ (RATA)} \qquad \frac{F \wedge C}{F} \text{ (RATE)}$$

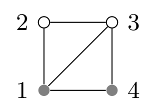
In all clause additions, we can add a new variable as long as it works with the side conditions of the rules. However, an excess of new variables can cause a proof checker to slow down, so there is a version of DRAT that forbids new variables known as DRAT^{-}.

Tseitin Formulas

A *Tseitin formula* is a CNF formula that represents a system of parity constraints where every variable appears in exactly two constraints. Such a formula is determined by a graph G: each edge e corresponds to a unique Boolean variable x_e and each vertex v defines a constraint $\bigoplus_{e \in E(v)} x_e = c(v)$, where $E(v)$ is the set of edges incident to v in G and $c : V(G) \to \{0, 1\}$ is the *charge function*. The Tseitin formula $T(G, c)$ is the conjunction of the xor representations for the constraints for every $v \in V(G)$. We call G the *Tseitin graph* of the formula. It is often assumed that the maximum degree of all vertices in G is bounded by a constant, so that the size of $T(G, c)$ is linear in $|var(T(G, c))| = |E(G)|$.

Example 2. *Let G be the following graph with $V(G) = \{1, 2, 3, 4\}$. Let $c : V(G) \to \{0, 1\}$ such that gray vertices have charge 0 and white vertices have charge 1.*

$x_{12} \oplus x_{13} \oplus x_{14} = 0$
$x_{12} \oplus x_{23} = 1$
$x_{13} \oplus x_{23} \oplus x_{34} = 1$
$x_{14} \oplus x_{34} = 0$



The Tseitin formula for this graph and this charge c is

$$T(G, c) = \mathsf{xor}(x_{12}, x_{13}, x_{14}) \wedge \mathsf{xor}(\bar{x}_{12}, x_{23}) \wedge \mathsf{xor}(\bar{x}_{13}, x_{23}, x_{34}) \wedge \mathsf{xor}(x_{14}, x_{34}).$$

Tseitin formulas were introduced by Tseitin (1968, 1983) in the 1960s as hard instances for proof systems, despite an easy criterion to decide their satisfiability (Urquhart 1987, Lemma 4.1). Urquhart (1987) later showed that when G belongs to the family of bounded-degree expander graphs (whose definition we omit), all resolution refutations of $T(G, c)$ require exponentially many clauses. This was generalized by Ben-Sasson and Widgerson who used their width-length relations on refutation proofs to derive exponential lower bounds parameterized on the edge expansion of G (Ben-Sasson and Wigderson 2001). Beyond expansion, the key parameter to characterize the hardness of Tseitin formulas for resolution could be the treewidth of the graph. Treewidth is a very well-known graph parameter whose definition we omit (see (Bodlaender 1998)). Intuitively the treewidth of G, denoted by $tw(G)$, is an integer between 0 and $|V(G)|$ that measures how close G is to a tree (trees having treewidth 1). On the one hand, it was shown (Alekhnovich and Razborov 2011) that unsatisfiable Tseitin formulas have resolution refutations of length at most $2^{O(tw(G))}|E(G)|^{O(1)}$, thus a logarithmic treewidth guarantees short refutations. On the other hand, combining the width-length relation with Corollaries 8 and 16 of Galesi et al.'s (2020) yields the following:

Theorem 4. *Let G be an n-vertex graph whose maximum degree is bounded by a constant, if $tw(G) = \Omega(n)$, then the length of a shortest resolution refutation of an unsatisfiable Tseitin formula $T(G, c)$ is at least $2^{\Omega(n)}$.*

Note that there is still a gray area for $tw(G)$ less than linear, but more than logarithmic in n. Note also that Tseitin formulas are easily refutable in proof systems different from resolution, regardless of high treewidth (Itsykson et al. 2020; Bonacina, Bonet, and Levy 2023).

Parity Problems

For a constraint $x_1 \oplus \cdots \oplus x_n = c$, the number of clauses in the canonical representation is exponential in n. We can use the xor notation to build larger parity constraints if we include auxiliary variables called Tseitin variables:

Definition 4. *Let σ be a permutation of n elements and $X = \{ x_i \mid 1 \leq i \leq n \}$ be an ordered set of literals. We define* $\mathsf{Parity}(X, T, \sigma) = \mathsf{xor}(x_{\sigma(1)}, x_{\sigma(2)}, t_1) \wedge \bigwedge_{j=1}^{n-4} \mathsf{xor}(t_j, x_{\sigma(j+2)}, t_{j+1}) \wedge \mathsf{xor}(t_{n-3}, x_{\sigma(n-1)}, x_{\sigma(n)})$, *where $T = \{ t_i \mid i \leq n-3 \}$ are Tseitin variables.*

Here $\mathsf{Parity}(X, T, \sigma)$ is satisfiable if and only if the total parity of X is 0. If we wanted a constraint which is satisfiable if and only if the parity was 1, again we simply flip a literal. In our particular Tseitin encoding, we structure the $\oplus$ linearly so that the formula for $n = 5, \sigma = id$ looks like $((((x_1 \oplus x_2) \oplus x_3) \oplus x_4) \oplus x_5)$. Here the formula depth is linear, however the structure does not affect the satisfiability as $\oplus$ is associative. Furthermore the actual permutation σ does not affect the satisfiability because $\oplus$ is commutative.

Problem 1: Reordered Parity

In our first problem we simply take two parity constraints that are in contradiction. We simultaneously state that the variables of X have 0 parity and 1 parity. This is obviously a contradiction. However in order to make it difficult we use two different permutations to obscure the conflict. We define

$$\mathsf{rPar}(n, \sigma) = \mathsf{Parity}(X, S, id) \wedge \mathsf{Parity}(X', T, \sigma)$$

where $X = \{ x_i \mid 1 \leq i \leq n \}$, $X' = \{ x_i \mid 1 \leq i < n \} \cup \{\bar{x}_n\}$, σ is a permutation of n elements, and *id* is the identity map. S, T, and X are disjoint sets of variables. Note that $\mathsf{rPar}(n, \sigma)$ is a Tseitin formula where each xor constraint corresponds to a vertex of the underlying graph. This is because every variable appears in exactly two xor constraints. For n fixed, its Tseitin graph depends only on σ and its vertices all have charge 0, except for the vertex corresponding to $\mathsf{xor}(t_{n-3}, x_{\sigma(n-1)}, \bar{x}_{\sigma_n})$.

Fact 1. $\mathsf{rPar}(n, \sigma)$ *is an unsatisfiable Tseitin formula.*

The version where the identity map is used for σ is the most natural, and is easy to solve. However the random version can still appear from equivalences (which themselves are xors) obfuscating the random parity. For example: A system of equations containing $x_1 \oplus x_2 \oplus x_3 \oplus x_4 = 0$ and $x_5 \oplus x_6 \oplus x_7 \oplus x_8 = 1$ and other binary clauses implying: $x_5 \leftrightarrow x_4$ and $x_6 \leftrightarrow x_1$ and $x_7 \leftrightarrow x_2$ and $x_8 \leftrightarrow x_3$. A standard CNF encoding using the natural variable ordering $x_1 < x_2 < x_3 < x_4 < x_5 < x_6 < x_7 < x_8$ (as done in the Dubois benchmark family) yields a non-trivial random parity problem after removing binary clauses, namely: $x_1 \oplus x_2 \oplus x_3 \oplus x_4 = 0$ encoded in CNF using the ordering $x_1 < x_2 < x_3 < x_4$, and $x_4 \oplus x_1 \oplus x_2 \oplus x_3 = 1$ encoded in CNF using the ordering $x_4 < x_1 < x_2 < x_3$.

Problem 2: Random Parity Addition

Problem 1 is a simple special case. In general, solvers and preprocessors want to deal with XOR-constraints by Gaussian elimination. In Gaussian elimination we use multiple steps involving adding two parity constraints together to get a third parity constraint. Since the parity constraints may have a large number of input variables we would have to use Tseitin variables, even in the third constraint. We model the difficulty of an addition step by taking the three parity constraints: the two summands and the negation of the sum, in conjunction to get a contradiction.

Let $a = \mathsf{Parity}(A, S, \sigma_a)$ and $b = \mathsf{Parity}(B, T, \sigma_b)$, where A and B are subsets of $X = \{ x_i \mid 1 \leq i \leq n \}$ and S and T are disjoint sets of Tseitin variables. We define

$$\mathsf{rAddPar}(a, b, \sigma_c) = \mathsf{Parity}(A, S, \sigma_a) \wedge \mathsf{Parity}(B, T, \sigma_b) \wedge \mathsf{Parity}(C, U, \sigma_c)$$

where U is disjoint from S, T and X. Here C is the symmetric difference of A and B but with a literal flipped. In a first scenario, the variable ordering σ_c for the sum constraint is independent of that used in a and b. This is modeled fixing σ_c to be the identity *id*. For convenience we write $\mathsf{rAddPar}(a, b) = \mathsf{rAddPar}(a, b, id)$. A more clever approach is to choose σ_c favorably for σ_a and σ_b. We will precise what we mean by "choosing σ_c favorably" later in the paper.

Again $\mathsf{rAddPar}(a, b, \sigma_c)$ is a Tseitin formula since every variable appears in two xor constraints. Every variable appearing in A and B does not appear in the third constraint, and every variable in the symmetric difference of A and B appears a second time the third constraint. The Tseitin variables are disjoint so they also appear in exactly two xor.

Fact 2. $\mathsf{rAddPar}(a, b, \sigma_c)$ *is an unsatisfiable Tseitin formula.*

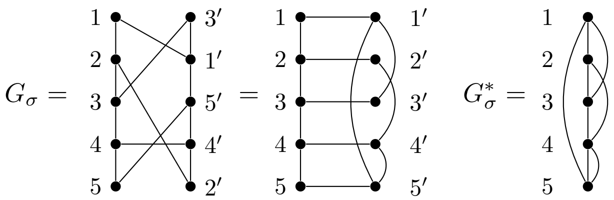
The Graph Model and Lower Bounds

In this section we present our lower bounds on the length of resolution refutations for rPar and rAddPar when they are constructed in a random fashion. The complete proofs of some results are deferred to the full version of the paper. The current paper should contain enough proof sketches and intuition for the reader to navigate through our results.

Lower Bounds for Reordered Parity

In the following, we denote the set $\{1, \ldots, n\}$ by $[n]$ and we call $\mathfrak{S}_n$ the set of permutations of $[n]$. Consider $2n$ vertices labeled $1, \ldots, n, 1', \ldots, n'$, a permutation $\sigma \in \mathfrak{S}_n$, and let G_σ be the graph over these vertices whose edge set is $\{ (i, i+1) \mid i < n \} \cup \{ (\sigma(i)', \sigma(i+1)') \mid i < n \} \cup \{ (i, i') \mid i \leq n \}$. Let G^*_σ be the multigraph obtained by contracting the edges (i, i') for all $i \in [n]$. That is, $V(G^*_\sigma) = [n]$ and, for every edge (i, j), (i', j), (i, j') or (i', j') in $E(G)$, we add an edge (i, j) to $E(G^*_\sigma)$.

Example 3. *Let $n = 5$ and $\sigma(1) = 3$, $\sigma(2) = 1$, $\sigma(3) = 5$, $\sigma(4) = 4$, $\sigma(5) = 2$. The graph G_σ and G^*_σ are:*



The maximum degree of a vertex of G_σ (resp. G^*_σ) is 3 (resp. 4). Since G^*_σ is a minor of G_σ after merging of the parallel edges, we have that $tw(G^*_\sigma) \leq tw(G_\sigma)$. We also have a bound in the other direction which may be useful when $tw(G_\sigma)$ is harder to compute than $tw(G^*_\sigma)$ in practice.

Lemma 1. *We have that $\frac{1}{2}tw(G_\sigma) \leq tw(G^*_\sigma) \leq tw(G_\sigma)$.*

The Tseitin graph of $\mathsf{rPar}(n, \sigma)$ is not exactly G_σ. The two graphs would be the same if we were to slightly modify the first and last constraints of $\mathsf{Parity}(X, S, id)$ and $\mathsf{Parity}(X', T, \sigma)$ by replacing $\mathsf{xor}(x_1, x_2, s_1)$ by $\mathsf{xor}(x_1, \bar{s}_0) \wedge \mathsf{xor}(s_0, x_2, s_1)$ etc.

Lemma 2. *The Tseitin graph of $\mathsf{rPar}(n, \sigma)$ is obtained by contracting four edges of G_σ.*

Proof sketch. Contract the edges $(1, 2)$, $(n - 1, n)$, $(\sigma(1)', \sigma(2)')$ and $(\sigma(n-1)', \sigma(n)')$ of G_σ. □

Since an edge contraction can only decrease the treewidth by one, it follows that the treewidth of the Tseitin graph of $\mathsf{rPar}(n, \sigma)$ is at least $tw(G_\sigma) - 4$. But then we show that when σ is sampled uniformly at random from $\mathfrak{S}_n$, with high probability (i.e., with probability tending to 1 as n increases) both G_σ and $\mathsf{rPar}(n, \sigma)$ have linear treewidth.

Lemma 3. *There is a constant $\alpha > 0$ such that $\Pr(tw(G_\sigma) < \alpha n)$ vanishes to 0 as n increases when σ is chosen uniformly at random in $\mathfrak{S}_n$.*

Proof. Kim and Wormald (2001) have studied the graph distribution $\mathcal{H}_n \oplus \mathcal{H}_n$ where each graph over n vertices is the superposition of two independent hamiltonian cycles over these vertices (merging parallel edges). They call $\mathcal{G}_{4,n}$ the uniform distribution over all 4-regular graphs over n vertices. They show that any sequence of events is true asymptotically almost surely (a.a.s.) in $\mathcal{H}_n \oplus \mathcal{H}_n$ if and only if it is true a.a.s. in $\mathcal{G}_{4,n}$ (Kim and Wormald 2001, Theorem 2).

The treewidth of a random 4-regular graph from $\mathcal{G}_{4,n}$ is linear in n with high probability (Chandran and Subramanian 2003). On the one hand, Chandran and Subramanian (2003) have shown that the treewidth of a d-regular graph G is at least $\lfloor \frac{3n}{4} \frac{(d - \lambda_2(G))}{(3d - 2\lambda_2(G))} \rfloor - 1$ where $\lambda_2(G)$ is the second largest eigenvalue of the adjacency matrix of G. On the other hand, Friedman has shown that, for any fixed $\varepsilon > 0$ and any $d \geq 2$, $|\lambda_2(G)| \leq 2\sqrt{d-1} + \varepsilon$ holds with high probability when $G \in \mathcal{G}_{d,n}$ (Friedman 2003, Corollary 1.4)[1]. The combination of the two results yields that, with high probability when $G \in \mathcal{G}_{4,n}$, $tw(G) \geq \Omega(n)$.

Thus, with high probability, a random graph $G \in \mathcal{H}_n \oplus \mathcal{H}_n$ has linear treewidth. Now G^*_σ is not the superposition of two independent hamiltonian cycles but the superposition of two independent paths. But if we close both paths before superposition, then we obtain a graph in $\mathcal{H}_n \oplus \mathcal{H}_n$ whose treewidth is at least $tw(G^*_\sigma) - 2$. So with high probability $tw(G_\sigma) \geq tw(G^*_\sigma) \geq \Omega(n)$ holds. □

$\mathsf{rPar}(n, \sigma)$'s graph has degree at most 4 and linear treewidth with high probability so, by Theorem 4 we immediately have that the formula is hard for resolution.

Theorem 1. *There is a constant $\alpha > 0$ such that, with probability tending to 1 as n increases, the length of a shortest resolution refutation of $\mathsf{rPar}(n, \sigma)$ where σ is chosen uniformly at random in $\mathfrak{S}_n$, is least $2^{\alpha n}$.*

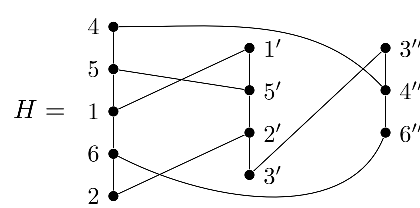
Lower Bounds for Random Parity Addition

The symmetric difference of two subsets A and B of $X = \{x_1, \ldots, x_n\}$ is denoted by $A \triangle B := (A \cup B) \setminus (A \cap B)$. Recall that for $a = \mathsf{Parity}(A, S, \sigma_a)$ and $b = \mathsf{Parity}(B, T, \sigma_b)$, the formula $\mathsf{rAddPar}(a, b) = a \wedge b \wedge \mathsf{Parity}(C, U, id)$ is a Tseitin formula (where $U \cap S = U \cap T = S \cap T = \emptyset$ and C is $A \triangle B$ with one literal flipped). Let us describe $\mathsf{rAddPar}(a, b)$'s Tseitin graph. We call H the graph whose vertices are split into three sets $V = \{ i \mid x_i \in A \}$, $V' = \{ i' \mid x_i \in B \}$ and $V'' = \{ i'' \mid x_i \in A \triangle B \}$. The edge set of H contains (i, i') for all $x_i \in A \cap B$, and (i, i'') for all $x_i \in A \cap (A \triangle B)$, and (i', i'') for all $x_i \in B \cap (A \triangle B)$. The vertices in V (resp. V' and V'') are also connected in a path following the order σ_a (resp. σ_b and id). H is not exactly the Tseitin graph of $\mathsf{rAddPar}$, but is close enough and easier to analyze.

Lemma 4. *For $a = \mathsf{Parity}(A, S, \sigma_a)$ and $b = \mathsf{Parity}(B, T, \sigma_b)$ the Tseitin graph of $\mathsf{rAddPar}(a, b)$ is obtained by contracting six edges of the graph H.*

At this point it is worth giving an example of such a graph H.

Example 4. *Let $n = 6$, $A = \{x_1, x_2, x_4, x_5, x_6\}$ and $B = \{x_1, x_2, x_3, x_5\}$. So $A \triangle B = \{x_3, x_4, x_6\}$. The constraints encoded in CNF with* Parity *are $x_1 \oplus x_2 \oplus x_4 \oplus x_5 \oplus x_6 = 0$, $x_1 \oplus x_2 \oplus x_3 \oplus x_5 = 0$ and $x_3 \oplus x_4 \oplus x_6 = 1$. Let $\sigma_a(1) = 4$, $\sigma_a(2) = 5$, $\sigma_a(4) = 1$, $\sigma_a(5) = 6$, $\sigma_a(6) = 2$ and $\sigma_b(1) = 1$, $\sigma_b(2) = 5$, $\sigma_b(3) = 2$, $\sigma_b(5) = 3$. Then the graph H is:*



The Tseitin graph of $\mathsf{rAddPar}(a, b)$ is the graph H above after contraction of the edges $(4, 5)$, $(6, 2)$, $(1', 5')$, $(2', 3')$, $(3'', 4'')$ and $(4'', 6'')$.

[1] Note that our $\mathcal{G}_{d,n}$ is Friedman's $\mathcal{K}_{d,n}$

When the constraints a and b are constructed in a random fashion, the hardness of $\mathsf{rAddPar}(a, b)$ for resolution stems from the hardness of rPar. One proves this by showing that, when the parameters of a and b are sampled uniformly at random, H is a random graph that, w.h.p., admits a graph G_σ as a minor, for σ a permutation over $\Omega(n)$ elements of X. When this minor exists (which is almost always the case), it also follows a uniform distribution, and thus its treewidth is in $\Omega(n)$ by Lemma 3. This then shows that $tw(H) = \Omega(n)$ holds w.h.p., and the lower bounds for resolution follows.

Theorem 2. *There is a constant $\alpha > 0$ such that, with probability tending to 1 as n increases, when A, B, σ_a and σ_b are chosen independently and uniformly at random, the length of a shortest resolution refutation of* $\mathsf{rAddPar}(a, b)$ *is least $2^{\alpha n}$.*

Notice here that σ_a, σ_b and *id* are relative to each other shuffled randomly. This is the most chaotic scenario which is likely to contribute to its difficulty. Let us instead briefly discuss an example that favors shorter proofs. We are given a and b in their random orders but, in an addition step, we may be the ones creating the encoding for the sum constraint and so we can choose the permutation σ_c favorably by ensuring that $\sigma_c(i) < \sigma_c(j)$ if and only if either:

- $x_i \in A \setminus B$ and $x_j \in B \setminus A$
- $x_i, x_j \in A \setminus B$ and $\sigma_a(i) < \sigma_a(j)$
- $x_i, x_j \in B \setminus A$ and $\sigma_b(i) < \sigma_b(j)$

Even in this case, the encoding $\mathsf{rAddPar}(a, b, \sigma_c) = a \wedge b \wedge \mathsf{Parity}(C, U, \sigma_c)$ will be hard w.h.p. when $|A \cap B| = \Omega(n)$ since then we can find the minor G_σ evoked above by only looking in $H[V \cup V']$. The condition $|A \cap B| = \Omega(n)$ is fulfilled almost surely when A and B are chosen uniformly.

Sorting and Upper Bounds

Philipp and Rebola-Pardo (2016) showed that XOR-reasoning such as in Gaussian elimination can have short proofs; a BDD approach can find polynomial-size extended resolution proofs (Sinz and Biere 2006). For these particular formulas we can do even better, reducing the complexity and the number of extra variables needed.

Lemma 5 (Chew and Heule 2020)**.** *Suppose we have a CNF F and two sets of XOR clauses* $\mathsf{xor}(x, y, p)$ *and* $\mathsf{xor}(p, z, q)$, *where variable p appears nowhere in F. We can infer:*

$$\frac{F \wedge \mathsf{xor}(x, y, p) \wedge \mathsf{xor}(p, z, q)}{F \wedge \mathsf{xor}(y, z, p) \wedge \mathsf{xor}(p, x, q)}$$

in 32 of DRAT steps without adding new variables.

This provides the building blocks for the short proofs.

Lemma 6 (Chew and Heule 2020)**.** *Given two permutations σ_1 and σ_2, X, S and T are disjoint sets of variables where both S variables and T variables do not appear in CNF F, $F \wedge$* $\mathsf{Parity}(X, S, \sigma_1)$ *can be transformed into $F \wedge$* $\mathsf{Parity}(X, T, \sigma_2)$ *in $O(n \log n)$ many DRAT steps where $|X| = n$.*

Sketch Proof.
1. In $O(n \log n)$ applications of Lemma 5 we can take the linear structure of the Tseitin variables and reorganize it into a balanced binary tree, using a divide-and-conquer approach.
2. In $O(n \log n)$ applications of Lemma 5 we can make any permutation of the leaf edges. By first swapping any two leaves takes $O(\log n)$ applications of Lemma 5 and then $n-1$ swaps are required (in the worst case) and sufficient to place every variable in place.
3. In $O(n \log n)$ applications of Lemma 5 we can take our balanced binary tree and return it to a linear structure. □

Theorem 3. *For any parity constraints a, b over n input variables X, there are DRAT$^-$ refutations of* $\mathsf{rAddPar}(a, b)$ *that have $O(n \log n)$ many lines.*

Sketch Proof. Using Lemma 6 we can rearrange the the random orderings to an easy case where all variables appear in order in $O(n \log n)$-step. The remaining refutation is a linear-sized resolution refutation. □

Experiments

We ran experiments to confirm that the reordered parity and random parity addition formulas are hard to refute for CDCL solvers, and that their hardness is largely explained by the treewidth of their Tseitin graphs. This is expected given the lower bounds of Theorems 1 and 2, but these results are asymptotic and probabilistic, and it is not certain that they apply to relatively small formulas encountered in practice.

The experiments described here were performed on a cluster with Intel Xeon E5649 processors at 2.53 GHz running 64-bit Linux. An 8 GB memory limit and varying time limits were enforced with RUNSOLVER (Roussel 2011). The benchmarks are available here: (Chew et al. 2023). [2]

Problem 1: Reordered Parity

We generated a benchmark set of $\mathsf{rPar}(n, \sigma)$ for $n = 50$. Experiments for increasing values of n were done by Chew and Heule (2020). To get formulas of varying treewidth, the permutations σ were constructed in several ways:

(a) 5 permutations were drawn from a uniform distribution.
(b) 30 permutations were obtained from a stochastic process following a Mallows distribution whose parameter controls the likelihood of inversions (Mallows 1957).
(c) 30 permutations come from a sequence of random adjacent swaps, with a varying number of swaps.
(d) 15 permutations were constructed by a sequence of random adjacent swaps until an element is a set distance away from its original position in the original order.

An ideal construction would have allowed us to uniformly sample graphs G_σ for a fixed n, with treewidth lying in a fixed range. But we know of no such construction that is also efficient. Hence the three last constructions listed above, that have parameters that intuitively give us some control on the treewidth. The drawback is that the graphs are not sampled uniformly at random, as in the theoretical results.

The resulting Tseitin graphs have 100 vertices each, so determining their treewidth is challenging. We obtained upper and lower bounds using tools by Tamaki (2022).[3] Within

[2] https://doi.org/10.5281/zenodo.10391790
[3] https://github.com/twalgor/tw

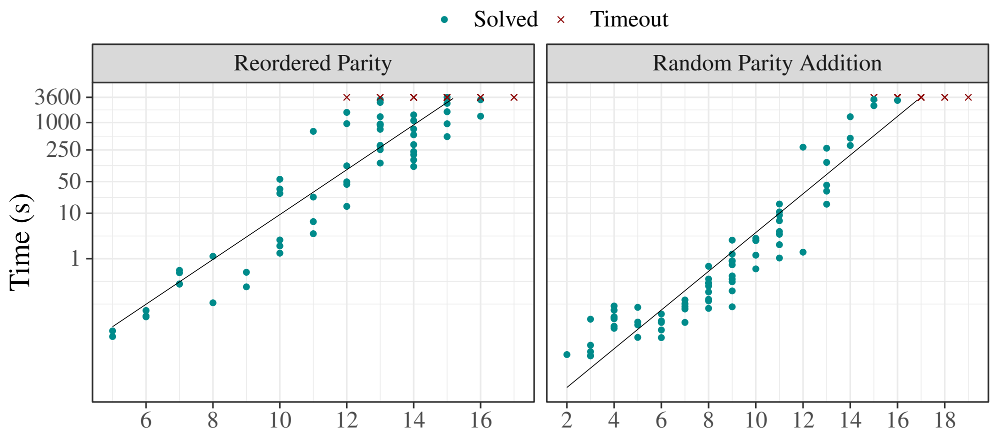


Figure 1: Treewidth (x-axis) vs. solving time (y-axis) for reordered parity (left) and random parity addition (right).

a time limit of 3600 seconds, the treewidth of only 30 graphs could be computed exactly. For another 30 graphs, no non-trivial lower bound was returned. To get lower bounds for such graphs G_σ, we determined the treewidth of the graph G^*_σ obtained by contracting the edges $(i, \sigma(i))$ for $i \in [n]$. Lemma 1 shows that $tw(G^*_\sigma) \le tw(G_\sigma) \le 2tw(G^*_\sigma)$, and since the graphs G^*_σ only contain half as many vertices, matching upper and lower bounds could be computed for all except two instances (curiously, these were graphs for which the treewidths of the original graphs could be determined).

We ran the CDCL solver CaDiCaL (Biere et al. 2020) on each reordered parity formula with default settings and a timeout of 3600 seconds. CaDiCaL generates Reverse Unit Propagation (RUP) proofs (Gelder 2008; Heule, Jr., and Wetzler 2013), which can be converted to resolution proofs with a quadratic overhead (Goldberg and Novikov 2003). Figure 1 (left) plots the solver's running time against the best upper bounds on the treewidth for each instance. The regression line clearly shows that the running time grows exponentially with the treewidth. By contrast, CryptoMiniSat (Soos, Nohl, and Castelluccia 2009), a solver that is capable of reasoning with XORs using Gaussian elimination, was able to solve all instances within a few seconds.

Problem 2: Random Parity Additions

We generated a benchmark set consisting of 95 reordered parity formulas $\mathsf{rAddPar}(a, b, \sigma_c)$. To get formulas of varying treewidth, we altered the value p, which dictates the probability of each input variable being selected to be including in a, and the same probability also for b. We chose p in increments of 0.05 from 0.05 up to 0.95. We scaled the number of variables we drew from based on p to keep the expectation of the number of clauses the same, in our sample the number of clauses ended up being between 360 and 536. σ_c was chosen to be favorable to solving. Figure 1 plots the solver's running time against the best upper bounds on the treewidth for each instance. Once again we see running time grows exponentially with the treewidth.

Conclusion

We present both theoretical and experimental evidence that treewidth explains the hardness of reordered parity, and random parity additions for CDCL/resolution.

Chew and Heule (2020) have left the DRAT$^-$ upper bound for rPar without a proof lower bound for resolution. We have now provided that. In particular, noticing that the instances were Tseitin formulas, we were driven to study the treewidth of their underlying graphs and we have shown that it is, with high probability, linear in the number of variables. And although the relationship between resolution refutations of Tseitin formulas and the graph's treewidth is not fully understood yet, results do exist for linear treewidth that are enough for us to prove exponential lower bounds on the length of the resolution proofs. Previous experiments showed the exponential increase in CaDiCaL proof size as the number of variables increases (Chew and Heule 2020), in this paper we show that same exponential increase (but in solving time), but with the number of variables and clauses controlled and now the treewidth being varied.

We generalize this further to $\mathsf{rAddPar}$ which draws its motivation from Gaussian elimination. $\mathsf{rAddPar}$ provides yet another example of a hard Tseitin formula, and its hardness is confirmed both theoretically and experimentally. Again, treewidth is the important factor in determining its hardness. In both the rPar and $\mathsf{rAddPar}$ case we can draw the conclusion that the variable order matters. Just as in rPar, we can show that we have short DRAT$^-$ proofs for $\mathsf{rAddPar}$. This will be useful for verification, with a hope that it may be generalized to Gaussian elimination.

In future work, it would be interesting to explore the BDD-techniques for dealing with XOR-constraints, in a similar manner to our exploration on CDCL here. On reordered parity EBDDRES (Sinz and Biere 2006) can perform even more poorly than CaDiCaL, but recent work (Bryant, Biere, and Heule 2022) show BDD-solvers can perform even better than Chew and Heule's sorting tool, so the overall picture may be more complicated.

Acknowledgments

The authors acknowledge the support from the FWF (P36420, ESP 197, ESP 235) and the WWTF (ICT19-060, ICT19-065).

References

Alekhnovich, M.; and Razborov, A. A. 2011. Satisfiability, Branch-Width and Tseitin tautologies. *Comput. Complex.*, 20(4): 649–678.

Ben-Sasson, E.; and Wigderson, A. 2001. Short proofs are narrow - resolution made simple. *Journal of the ACM*, 48(2): 149–169.

Biere, A.; Fazekas, K.; Fleury, M.; and Heisinger, M. 2020. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling Entering the SAT Competition 2020. In Balyo, T.; Froleyks, N.; Heule, M.; Iser, M.; Järvisalo, M.; and Suda, M., eds., *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, volume B-2020-1 of *Department of Computer Science Report Series B*, 51–53. University of Helsinki.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2021. *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press. ISBN 978-1-64368-160-3.

Bodlaender, H. L. 1998. A Partial *k*-Arboretum of Graphs with Bounded Treewidth. *Theor. Comput. Sci.*, 209(1-2): 1–45.

Bonacina, I.; Bonet, M. L.; and Levy, J. 2023. Polynomial Calculus for MaxSAT. In Mahajan, M.; and Slivovsky, F., eds., *26th International Conference on Theory and Applications of Satisfiability Testing, SAT 2023, July 4-8, 2023, Alghero, Italy*, volume 271 of *LIPIcs*, 5:1–5:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Bryant, R. E.; Biere, A.; and Heule, M. J. H. 2022. Clausal Proofs for Pseudo-Boolean Reasoning. In Fisman, D.; and Rosu, G., eds., *Tools and Algorithms for the Construction and Analysis of Systems*, 443–461. Cham: Springer International Publishing. ISBN 978-3-030-99524-9.

Buss, S.; and Thapen, N. 2019. DRAT proofs, propagation redundancy, and extended resolution. In *International Conference on Theory and Applications of Satisfiability Testing*, 71–89. Springer.

Chandran, L. S.; and Subramanian, C. R. 2003. A spectral lower bound for the treewidth of a graph and its consequences. *Inf. Process. Lett.*, 87(4): 195–200.

Chew, L.; de Colnet, A.; Slivovsky, F.; and Szeider, S. 2023. Dataset of Random Reordered Encodings of Parity Problems.

Chew, L.; and Heule, M. J. H. 2020. Sorting Parity Encodings by Reusing Variables. In Pulina, L.; and Seidl, M., eds., *Theory and Applications of Satisfiability Testing - SAT 2020 - 23rd International Conference, Alghero, Italy, July 3-10, 2020, Proceedings*, volume 12178 of *Lecture Notes in Computer Science*, 1–10. Springer.

Fichte, J. K.; Berre, D. L.; Hecher, M.; and Szeider, S. 2023. The Silent (R)evolution of SAT. *Communications of the ACM*, 66(6): 64–72.

Friedman, J. 2003. A proof of Alon's second eigenvalue conjecture. In Larmore, L. L.; and Goemans, M. X., eds., *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, 720–724. ACM.

Galesi, N.; Talebanfard, N.; and Torán, J. 2020. Cops-Robber Games and the Resolution of Tseitin Formulas. *ACM Trans. Comput. Theory*, 12(2): 9:1–9:22.

Gelder, A. V. 2008. Verifying RUP Proofs of Propositional Unsatisfiability. In *International Symposium on Artificial Intelligence and Mathematics, ISAIM 2008, Fort Lauderdale, Florida, USA, January 2-4, 2008*.

Goldberg, E. I.; and Novikov, Y. 2003. Verification of Proofs of Unsatisfiability for CNF Formulas. In *2003 Design, Automation and Test in Europe Conference and Exposition (DATE 2003), 3-7 March 2003, Munich, Germany*, 10886–10891. IEEE Computer Society.

Han, C.-S.; and Jiang, J.-H. R. 2012. When Boolean Satisfiability Meets Gaussian Elimination in a Simplex Way. In Madhusudan, P.; and Seshia, S. A., eds., *Computer Aided Verification*, 410–426. Berlin, Heidelberg: Springer Berlin Heidelberg.

Heule, M.; Jr., W. A. H.; and Wetzler, N. 2013. Trimming while checking clausal proofs. In *Formal Methods in Computer-Aided Design, FMCAD 2013, Portland, OR, USA, October 20-23, 2013*, 181–188. IEEE.

Itsykson, D.; Knop, A.; Romashchenko, A. E.; and Sokolov, D. 2020. On OBDD-based Algorithms and Proof Systems that Dynamically Change the order of Variables. *J. Symb. Log.*, 85(2): 632–670.

Itsykson, D.; Riazanov, A.; and Smirnov, P. 2022. Tight Bounds for Tseitin Formulas. In Meel, K. S.; and Strichman, O., eds., *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, August 2-5, 2022, Haifa, Israel*, volume 236 of *LIPIcs*, 6:1–6:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Järvisalo, M.; Heule, M. J. H.; and Biere, A. 2012. Inprocessing Rules. In Gramlich, B.; Miller, D.; and Sattler, U., eds., *Automated Reasoning*, 355–370. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-31365-3.

Kim, J. H.; and Wormald, N. C. 2001. Random Matchings Which Induce Hamilton Cycles and Hamiltonian Decompositions of Random Regular Graphs. *J. Comb. Theory, Ser. B*, 81(1): 20–44.

Mallows, C. L. 1957. Non-null ranking models. I. *Biometrika*, 44(1/2): 114–130.

Marques-Silva, J. P.; Lynce, I.; and Malik, S. 2009. Conflict-Driven Clause Learning SAT Solvers. In *Handbook of Satisfiability*. IOS Press.

Philipp, T.; and Rebola-Pardo, A. 2016. DRAT Proofs for XOR Reasoning. In Michael, L.; and Kakas, A., eds., *Logics in Artificial Intelligence*, 415–429. Cham: Springer International Publishing.

Pipatsrisawat, K.; and Darwiche, A. 2011. On the power of clause-learning SAT solvers as resolution engines. *Artificial Intelligence*, 175(2): 512 – 525.

Robinson, J. A. 1963. Theorem-Proving on the Computer. *Journal of the ACM*, 10(2): 163–174.

Roussel, O. 2011. Controlling a solver execution with the runsolver tool. *Journal on Satisfiability, Boolean Modeling and Computation*, 7(4): 139–144.

Sinz, C.; and Biere, A. 2006. Extended Resolution Proofs for Conjoining BDDs. In Grigoriev, D.; Harrison, J.; and Hirsch, E. A., eds., *Computer Science – Theory and Applications*, 600–611. Berlin, Heidelberg: Springer Berlin Heidelberg.

Soos, M. 2012. Enhanced Gaussian Elimination in DPLL-based SAT Solvers. In Berre, D. L., ed., *POS-10. Pragmatics of SAT*, volume 8 of *EPiC Series in Computing*, 2–14. EasyChair.

Soos, M.; Nohl, K.; and Castelluccia, C. 2009. Extending SAT Solvers to Cryptographic Problems. In Kullmann, O., ed., *Theory and Applications of Satisfiability Testing - SAT 2009*, 244–257. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-02777-2.

Tamaki, H. 2022. Heuristic Computation of Exact Treewidth. In Schulz, C.; and Uçar, B., eds., *20th International Symposium on Experimental Algorithms, SEA 2022, July 25-27, 2022, Heidelberg, Germany*, volume 233 of *LIPIcs*, 17:1–17:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Tseitin, G. 1968. On the Complexity of Derivation in Propositional Calculus. *Studies in Constructive Mathematics and Mathematical Logic*, Part 2: 115–125.

Tseitin, G. S. 1983. *On the Complexity of Derivation in Propositional Calculus*, 466–483. Springer Berlin Heidelberg. ISBN 978-3-642-81955-1.

Urquhart, A. 1987. Hard Examples for Resolution. *J. ACM*, 34(1): 209–219.